The Lift Monad in the Topos of Trees

Eric Giovannini

February 13, 2023

1 Background

Recall that the topos of trees $\mathcal{S}$ is the presheaf category $\mathtt{Set}^{\omega^o}$. An object X is a family $\{X_i\}$ of sets indexed by natural numbers, along with restriction maps $r_i^X \colon X_{i+1} \to X_i$.

A morphism from $\{X_i\}$ to $\{Y_i\}$ is a family of functions $f_i \colon X_i \to Y_i$ that commute with the restriction maps in the obvious way, that is, $f_i \circ r_i^X = r_i^Y \circ f_{i+1}$.

There is an operator $\triangleright$ ("later"), defined on an object X by $(\triangleright X)_0 = 1$ and $(\triangleright X)_{i+1} = X_i$. The restriction maps are given by $r^{\triangleright}{}_0 = !$, where $!$ is the unique map into 1, and $r^{\triangleright}{}_{i+1} = r_i^X$.

For each X, there is a morphism $\mathtt{next}^X \colon X \to \triangleright X$ defined pointwise by $\mathtt{next}_0^X = !$, and $\mathtt{next}_{i+1}^X = r_i^X$.

We also have a universe $\mathcal{U}$ of types, with encodings for operations such as sum types (written as $\widehat{+}$). There is also an operator $\widehat{\triangleright} \colon \triangleright\mathcal{U} \to \mathcal{U}$ such that $\mathsf{El}(\widehat{\triangleright}(\mathtt{next} A)) = \triangleright \mathsf{El}(A)$, where El is the type corresponding to the code A.

2 Guarded Fixpoints

Given a morphism $f \colon \triangleright X \to X$, we want to compute its fixed point $\mathtt{gfix} f : X$. We define $\mathtt{gfix} \colon (\triangleright X \Rightarrow X) \to X$ as a morphism in $\mathcal{S}$, where $(\triangleright X \Rightarrow X)$ is the exponential object, the object of functions from $\triangleright X$ to X.

We first describe $(\triangleright X \Rightarrow X)$. Since $\mathcal{S}$ is a presheaf category, we can use the Yoneda Lemma to describe the exponential objects, as reviewed below.

In general, let X and Y be object in the functor category $\mathtt{Set}^{\mathcal{C}^o}$, i.e., functors X and Y from $\mathcal{C}^o$ to $\mathtt{Set}$. We have by the Yoneda Lemma that for each object a of $\mathcal{C}$, the set $(X \Rightarrow Y)(a)$ is isomorphic to the set of natural transformations $\mathsf{Yo}(a) \to (X \Rightarrow Y)$, i.e., the set of morphisms $\mathsf{Yo}(a) \to (X \Rightarrow Y)$ in the functor category. But by the universal property of the exponential, such a morphism is the same as a morphism (in this case, a natural transformation) $\mathsf{Yo}(a) \times X \to Y$.

So, the set $(X \Rightarrow Y)(a)$ is isomorphic to the set of natural transformations $\mathsf{Yo}(a) \times X \to Y$, that is, $(\mathsf{Hom}(-, a) \times X) \to Y$.

With the above, we can compute the definition of $(\triangleright X \Rightarrow X)$ in $\mathcal{S}$. Since $\mathcal{C} = \omega$ is a preorder, the above result specializes to

$$(\triangleright X \Rightarrow X)_i = \{f^i : \forall j \leq i.(\triangleright X)_j \to X_j\}.$$

(The superscript is included to remind us of which set each function belongs to.)

The naturality condition says that, for $j' \leq j$ the following diagram commutes:

$$\begin{array}{ccc} (\triangleright X)_j & \xrightarrow{f^i\, j} & X_j \\ {\scriptstyle r^{\triangleright X}_{j \leq j'}}\downarrow & & \downarrow{\scriptstyle r^{X}_{j \leq j'}} \\ (\triangleright X)_{j'} & \xrightarrow[f^i\, j']{} & X_{j'} \end{array}$$

Additionally, the restriction maps $r_i^{(\triangleright X \Rightarrow X)} : (\triangleright X \Rightarrow X)_{i+1} \to (\triangleright X \Rightarrow X)_i$ are defined in the obvious way. Namely, if $f : \forall j \leq i + 1.(\triangleright X)_j \to X_j$ then in particular $f : \forall j \leq i.(\triangleright X)_j \to X_j$, so we define

$$r_i^{(\triangleright X \Rightarrow X)}(f) := f.$$

Note that this is just a special case of the action of the functor $X \Rightarrow Y$ discussed above on morphisms.

Now we can proceed to define $\texttt{gfix}$. We define $\texttt{gfix}_i : (\triangleright X \Rightarrow X)_i \to X_i$ by induction on i. We let

$$\texttt{gfix}_0(f) = f\, 0\, 1,$$

since here $f\, 0 : (\triangleright X)_0 \to X_0$ and we have $(\triangleright X)_0 = 1$. For $i \geq 1$, we define

$$\texttt{gfix}_i(f) = f\, i\, (\texttt{gfix}_{i-1}(f\, (i - 1))),$$

which has type X_i as follows: since $f\, (i-1) : ((\triangleright X)_{i-1} \to X_{i-1})$, then $\texttt{gfix}_{i-1}(f\, (i-1))$ has type X_{i-1}, which is equal to $(\triangleright X)_i$, and so it is a valid input to $f\, i$.

This completes the definition of $\texttt{gfix}_i$, and hence of $\texttt{gfix}$. We need to verify that $\texttt{gfix} f = f(\texttt{next}(\texttt{gfix} f))$, where $f : \triangleright X \Rightarrow X$. This follows from the definition of $\texttt{gfix}$ above.

3 The Lift Monad

For an object X, the lift monad $L_{\mho}X$ is defined as follows:

$$\begin{aligned} L_{\mho}X := \\ &\eta\colon X \to L_{\mho}X \\ &\mho\colon L_{\mho}X \\ &\theta\colon \triangleright(L_{\mho}X) \to L_{\mho}X \end{aligned}$$

In the setting of guarded domain theory, the above translates to the following guarded fixpoint:

$$\widehat{L_{\mho}}X = \mathtt{gfix}(\lambda Y\colon \triangleright\mathcal{U}.X \mathbin{\widehat{+}} 1 \mathbin{\widehat{+}} \widehat{\triangleright}Y)$$

With this definition, we can take $L_{\mho}X = \mathsf{El}(\widehat{L_{\mho}}X)$, and we can show that $L_{\mho}X = X+1+\triangleright(L_{\mho}X)$. This follows from the fact that $\mathtt{gfix}f = f(\mathtt{next}(\mathtt{gfix}f))$ and the rule $\mathsf{El}(\widehat{\triangleright}(\mathtt{next}A)) = \triangleright\mathsf{El}(A)$.

We can compute $(L_{\mho}X)_i$ for each i, as follows. Intuitively, we can think of having a potentially-erroring computation that returns values of type X, and $(L_{\mho}X)_i$ represents our view of the behavior of the computation provided we can watch it for $i+1$ steps.

We have

$$(L_{\mho}X)_0 = X_0 + 1 + (\triangleright L_{\mho}X)_0 = X_0 + 1 + 1.$$

This represents the behavior of the computation after one step. The first 1 represents the case where an error occurred, while the second 1 represents the case where the computation took more than one step, and hence we have run out of time to observe it.

For $i \geq 1$, we have

$$\begin{aligned} (L_{\mho}X)_i &= X_i + 1 + (\triangleright L_{\mho}X)_i \\ &= X_i + 1 + (L_{\mho}X)_{i-1}. \end{aligned}$$

In other words, when observing a computation for $i+1$ steps, either (1) we get a value in the first step, or (2) we error in the first step, or (3) the answer is not ready yet and we must wait, in which case we will have i steps left to watch the computation.

For concreteness, when $i = 1$, we have

$$(L_{\mho} X)_1 = X_1 + 1 + (X_0 + 1 + 1).$$

Here, the first 1 represents an error occurring in the first step of the computation, while the second 1 represents an error in the second step, and the third 1 represents having run out of steps to observe the computation.

We now discuss each of the constructors of $L_{\mho} X$. First consider the constructor $\eta^X \colon X \Rightarrow L_{\mho} X$. This is defined component-wise as $\eta_i^X = \lambda x.\mathtt{inl}(\mathtt{inl} x)$ (we assume sums are left-associative). The constructor $\mho^X \colon 1 \Rightarrow L_{\mho} X$ is defined by $\mho_i^X = \lambda().\mathtt{inl}(\mathtt{inr}\,())$. Lastly, the constructor $\theta^X \colon (\triangleright\,(L_{\mho} X) \Rightarrow L_{\mho} X)$ is defined by $\theta_i^X = \lambda \tilde{l}.\mathtt{inr}\,\tilde{l}$.

The restriction maps $r_i^{L_{\mho} X} \colon (L_{\mho} X)_{i+1} \to (L_{\mho} X)_i$ are defined by "truncating" the observation of the computation from $i+2$ steps to $i+1$ steps, as follows. If the computation has the form $\eta_{i+1}^X(x)$, then we return $\eta_i^X(r_i^X(x))$. If the computation has the form $\mho_{i+1}^X$, then we return $\mho_i^X$. Lastly, if the computation has the form $\theta_{i+1}^X(\tilde{l})$, then we return $\theta_0^X()$ if $i = 0$ and $(\theta_i^X(r_{i-1}^{L_{\mho} X}(\tilde{l})))$ if $i \geq 1$.

3.1 The Diverging Computation

Consider the "non-terminating" computation $\mathtt{gfix}(\theta) \colon L_{\mho} X$. This can be described pointwise as follows:

$$(\mathtt{gfix}(\theta))_0 = \mathtt{inr}\,() \colon (L_{\mho} X)_0,$$

and for $i \geq 1$,

$$(\mathtt{gfix}(\theta))_i = \mathtt{inr}((\mathtt{gfix}(\theta))_{i-1}),$$

where the latter type-checks because $(\mathtt{gfix}(\theta))_{i-1} \colon (L_{\mho} X)_{i-1}$.

Unfolding the above, we have

$$(\mathtt{gfix}(\theta))_i = \mathtt{inr}(\mathtt{inr}(\ldots(\mathtt{inr}\,())\ldots)),$$

where there are $i+1$ nested occurrences of $\mathtt{inr}$.

So, no matter how many steps we observe this computation for, it will never return a result within that number of steps. It will instead always say that it needs more time.

4 Weak Bisimilarity

We now define a weak bisimilarity relation "$\approx$" on $L_{\mho}X$. This relation is parameterized by an equivalence relation $\sim$ on X. Intuitively, two computations l and l' if they are equivalent "up to θ". Specifically, their underlying results are related by $\sim$ and they only differ in the number of steps taken to deliver their results.

In the below, we define $\delta \colon L_{\mho}X \to L_{\mho}X$ to be $\theta \circ \texttt{next}$.

We define $\approx$ via guarded fixpoint as follows:

$$
\begin{array}{l}
\approx = \texttt{gfix}(\lambda\ rec\ l\ l'. \\
\quad \text{case } l, l' \text{ of} \\
\qquad \mid \quad \mho, \mho \Rightarrow Unit \\
\qquad \mid \quad \eta(x), \eta(y) \Rightarrow x \sim y \\
\qquad \mid \quad \theta(\tilde{x}), \theta(\tilde{y}) \Rightarrow \widehat{\triangleright}(\lambda t.(rec\ t)(\tilde{x}\,t)(\tilde{y}\,t)) \\
\qquad \mid \quad \theta(\tilde{x}), \mho \Rightarrow \Sigma_{n:\mathbb{N}}.\theta(\tilde{x}) = \delta^n(\mho) \\
\qquad \mid \quad \theta(\tilde{x}), \eta(y) \Rightarrow \Sigma_{n:\mathbb{N}}.\Sigma_{x:X}.\theta(\tilde{x}) = \delta^n(\eta(x)) \times (x \sim y) \\
\qquad \mid \quad \mho, \theta(\tilde{y}) \Rightarrow \Sigma_{n:\mathbb{N}}.\theta(\tilde{y}) = \delta^n(\mho) \\
\qquad \mid \quad \eta(x), \theta(\tilde{y}) \Rightarrow \Sigma_{n:\mathbb{N}}.\Sigma_{y:X}.\theta(\tilde{y}) = \delta^n(\eta(y)) \times (x \sim y) \\
\qquad \mid \quad \text{otherwise} \Rightarrow \bot)
\end{array}
$$

Note that in the case where both variables are θ's, we have used tick abstraction so that we can apply the guarded recursive function rec.

We now consider the claim that for all $l : L_{\mho}X$,

$$l \approx (\texttt{gfix}(\theta)).$$

We claim that this does not hold. Specifically, we claim that

$$\mho \not\approx (\texttt{gfix}(\theta)).$$

In order for these to be related, according to the defintion of the relation, (and recalling that $(\texttt{gfix}(\theta)) = \theta(\texttt{next}(\texttt{gfix}(\theta))))$, we would need that $(\texttt{gfix}(\theta)) = \delta^n(\mho)$ for some n.

Suppose this were the case for some natural number n_0. Consider an index $i > n_0 \in \omega$. The left-hand side above is equal to

$$\texttt{inr}(\texttt{inr}(\ldots(\texttt{inr}\,())\ldots)),$$

(where there are $i+1$ occurrences of $\texttt{inr}$), while the right-hand side will have as its innermost term an $\texttt{inl}(\texttt{inr}\,())$, representing the error case $\mho$. Thus, the two terms cannot be equal for any n. It follows that

$$\mho \not\approx (\texttt{gfix}(\theta)).$$