5 Initial algebras and terminal coalgebras

5.1 Natural numbers

Following Peano we define the natural numbers as inductively generated from $0 : \mathbb{N}$ and $\mathrm{suc} : \mathbb{N} \to \mathbb{N}$. We can define functions like addition $\_+\_ : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ by recursion:

$$\begin{aligned} 0 + m &= m \\ \mathrm{suc}\, n + m &= \mathrm{suc}\,(n + m) \end{aligned}$$

and we can prove properties such that for all $n : \mathbb{N}$ we have that $n + 0 = n$ by induction:

$n = 0$ $0 + 0 = 0$ follows from the definition of $\_ + \_$.

$n = \mathrm{suc}\, m'$ we assume that the statement holds for m:

$$\begin{aligned} n + 0 &= \mathrm{suc}\, m + 0 \\ &= \mathrm{suc}\,(m + 0) && \text{Defn of } \_ + \_ \\ &= \mathrm{suc}\, m && \text{Ind.hypothesis} \\ &= n \end{aligned}$$

To categorify natural numbers we observe that we have two morphisms

$$\begin{aligned} 0 &: 1 \to \mathbb{N} \\ \mathrm{suc} &: \mathbb{N} \to \mathbb{N} \end{aligned}$$

and given a set A and $z : A$ or equivalently $z : 1 \to A$ and $s : A \to A$ we can define $\mathrm{It}_A\, z\, s : \mathbb{N} \to A$ as:

$$\begin{aligned} \mathrm{It}_A\, z\, s\, 0 &= z \\ \mathrm{It}_A\, z\, s\, (\mathrm{suc}\, n) &= s\, (\mathrm{It}_A\, z\, s\, n) \end{aligned}$$

this is the unique morphism that makes the following diagram commute

$$\begin{tikzcd}
 & A \arrow[r, "s"] & A \\
1 \arrow[r, "0"] \arrow[ru, "z"] & \mathbb{N} \arrow[r, "\mathrm{suc}"] \arrow[u, dashed, "\mathrm{It}_A\, z\, s"'] & \mathbb{N} \arrow[u, dashed, "\mathrm{It}_A\, z\, s"']
\end{tikzcd}$$

The uniqueness of $\mathrm{It}_A\, z\, s$ can be shown by induction. We assume that there is another function $h : \mathbb{N} \to A$ that makes the digram commute, i.e.

$$\begin{aligned} h \circ 0 &= z \\ h \circ \mathrm{suc} &= s \circ h \end{aligned}$$

We now show that $h = \mathrm{It}_A\, z\, s$. Using extensionality it is enough to show $h\, n = \mathrm{It}_A\, z\, s\, n$ for all $n : \mathbb{N}$. We show this by induction:

$n = 0$

$$\begin{aligned} h\,0 &= h \circ 0 \\ &= z \\ &= \mathrm{It}_A\, z\, s\, 0 \end{aligned}$$

$n = \mathrm{suc}\, n'$

$$\begin{aligned} h\,(\mathrm{suc}\, n') &= (h \circ \mathrm{suc})\, n' \\ &= (s \circ h)\, n' \\ &= s(h\, n') \\ &= s(\mathrm{It}_A\, z\, s\, n') && \text{by ind.hyp.} \\ &= \mathrm{It}_A\, z\, s\,(\mathrm{suc}\, n') \end{aligned}$$

This leads directly to the definition of a *Natural Number Object (NNO)* in a category $\underline{\mathbf{C}}$ with a terminal object: it is given by an object $\mathbb{N} : |\underline{\mathbf{C}}|$ and two morphisms $0 : \underline{\mathbf{C}}(1, \mathbb{N})$ and $\mathrm{suc} : \underline{\mathbf{C}}(\mathbb{N}, \mathbb{N})$ such for any object A and $z : \underline{\mathbf{C}}(1, A)$ and $s : \underline{\mathbf{C}}(A, A)$ there is a unique morphism $\mathrm{It}_A\, z\, s$ that makes the above diagram commute.

We can derive the addition function exploiting also cartesian closure. We choose $A = \mathbb{N} \to \mathbb{N}$ and

$$\begin{aligned} &z : 1 \to (\mathbb{N} \to \mathbb{N}) \\ &z\,()\, n = n \\ &s : (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N}) \\ &s\, f\, n = \mathrm{suc}\,(f\, n) \end{aligned}$$

We can now define $\_ + \_ = \mathrm{It}_{\mathbb{N} \to \mathbb{N}}\, z\, s : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$.

Exercise 51 Convince yourself that this computes the same function as the previous definition of $\_ + \_$ using pattern maching.

We can prove that $n + 0 = 0$ using the unicity of It instead of induction. We observe that both $\mathrm{id}_{\mathbb{N}}$ and $\_ + 0 : \mathbb{N} \to \mathbb{N}$ make the following diagram commute:

$$\begin{tikzcd}
 & \mathbb{N} \arrow[r, "\mathrm{suc}"] & \mathbb{N} \\
1 \arrow[ur, "0"] \arrow[ur, phantom, "\mathrm{id}_{\mathbb{N}}"'] \arrow[r, "0"] & \mathbb{N} \arrow[u, shift left, "\_+0"'] \arrow[u, shift right] \arrow[r, "\mathrm{suc}"] & \mathbb{N} \arrow[u, shift right, "\mathrm{id}_{\mathbb{N}}"] \arrow[u, shift left, "\_+0"']
\end{tikzcd}$$

And hence by uniqueness $\mathrm{id}_{\mathbb{N}} = \_ + 0$, by applying both sides to $n : \mathbb{N}$ we obtain $n = \mathrm{id}\, n = (\_ + 0)\, n = n + 0$.

Exercise 52 Verify the claim that both functions make the diagram commute.

One basic function on natural numbers is the *predecessor*, i.e. the inverse to the successor. Initially we may think that we should define a function $\mathrm{pred} : \mathbb{N} \to \mathbb{N}$ with $\mathrm{pred}\,(\mathrm{suc}\,n) = n$. But what should $\mathrm{pred}\,0$ be? We could arbitrarily set $\mathrm{pred}\,0 = 0$ but this seems a very unprincipled move. It seems better to say that pred should return an *error* when applies to 0. That is as well known in functional programming, we should use the *Maybe*-monad [14] and use $\mathrm{Maybe}\,A = 1 + A$ as the result type. That is we define $\mathrm{pred} : \mathbb{N} \to 1 + \mathbb{N}$ as

$$\begin{aligned} \mathrm{pred}\,0 &= \mathrm{inj}_1\,() \\ \mathrm{pred}\,(\mathrm{suc}\,n) &= \mathrm{inj}_2\,n \end{aligned}$$

However, we cannot translate this definition directly into an application of It since in the successor case we referred directly to n while It just gives us the recursive result. However, the following alternative definition works:

$$\begin{aligned} \mathrm{pred}\,0 &= \mathrm{inj}_1\,() \\ \mathrm{pred}\,(\mathrm{suc}\,n) &= \begin{bmatrix} \mathrm{inj}_2\,0 \\ \mathrm{inj}_2 \circ \mathrm{suc} \end{bmatrix} \end{aligned}$$

The idea is that the 2nd definition computes the predeccessor by delaying the constructors by one step.

Exercise 53 Verify in detail that both definitions of pred compute the same function.

The second definition can be translated into

$$\begin{aligned} &\mathrm{pred} : \mathbb{N} \to 1 + \mathbb{N} \\ &\mathrm{pred} = \mathrm{It}_{1+\mathbb{N}}\,(\mathrm{inj}_1\,())\begin{bmatrix} \mathrm{inj}_2\,0 \\ \mathrm{inj}_2 \circ \mathrm{suc} \end{bmatrix} \end{aligned}$$

pred is the inverse of the constructors 0 and suc. We can make this precise by packaging 0 and suc into one function

$$\begin{aligned} &\mathrm{in} : 1 + \mathbb{N} \to \mathbb{N} \\ &\mathrm{in} = \begin{bmatrix} 0 \\ \mathrm{suc} \end{bmatrix} \end{aligned}$$

This is an instance of Lambek's lemma which we will verify in the general case later.

Exercise 54 Prove by induction or by using initiality that pred and in are an isomorphism, i.e.

$$\begin{aligned} &\mathrm{pred} \circ \mathrm{in} = \mathrm{id}_{1+\mathbb{N}} \\ &\mathrm{in} \circ \mathrm{pred} = \mathrm{id}_{\mathbb{N}} \end{aligned}$$

[14] Even though we don't actually know yet what a monad is

5.2 Initial algebras

Natural numbers are an instance of the general concept of an initial algebra which are the categorical counterpart of what we call an inductive definition, that is a datatype which is generated by constructors. An initial algebra is specified by an endofunctor T. In the case of natural numbers this was the functor $T_{\mathbb{N}} : \mathbf{Set} \to \mathbf{Set}$ defined as $T_{\mathbb{N}}\,X = 1 + X$.

We give a general definition: Given an endofunctor $T : \underline{\mathbf{C}} \to \underline{\mathbf{C}}$ an initial T-algebra is given by an object $\mu T : |\underline{\mathbf{C}}|$ and a morphism $\mathrm{in}_T :: \underline{\mathbf{C}}(T\,\mu\,T, \mu\,T)$ such that for any T-algebra, that is a pair of $A : |\underline{\mathbf{C}}|$ and a morphism $f : \underline{\mathbf{C}}(T\,A, A)$ there is a unique morphism $\mathrm{It}_T\,f : \underline{\mathbf{C}}\mu T A$ [15] that makes the following diagram commute:

$$\begin{tikzcd}
T\,A \arrow[r, "f"] & A \\
T\,(\mu T) \arrow[u, "T\,(\mathrm{It}_T\,f"] \arrow[r, "\mathrm{in}_T"] & \mu T \arrow[u, dashed, "\mathrm{It}_T\,f"']
\end{tikzcd}$$

Indeed, the initial algebra is just the initial object in the category of T-algebras, whose objects are T-algebras and given T-algebras $f : \underline{\mathbf{C}}(T\,A, A)$ and $g : \underline{\mathbf{C}}(T\,B, B)$ a morphism is given by a morphism $h : \underline{\mathbf{C}}(A, B)$ such that the following diagram commutes:

$$\begin{tikzcd}
T\,B \arrow[r, "g"] & B \\
T\,A \arrow[u, "T\,h"] \arrow[r, "f"] & A \arrow[u, "h"]
\end{tikzcd}$$

Identity and composition is given by identity and composition in $\underline{\mathbf{C}}$, it is easy to see that the corresponding diagrams commute.

Exercise 55 Show that $\mathbb{N}$ is the initial $T_{\mathbb{N}}$ algebra with $T_{\mathbb{N}} : \mathbf{Set} \to \mathbf{Set}$ defined as $T_{\mathbb{N}}\,X = 1 + X$.

We can understand the type of lists in a similar fashion. List are given by two constructors (pronounced nil and cons):

$$\begin{aligned}
[] &: \mathrm{List}\,A \\
\_\!:\!\_ &: A \to \mathrm{List}\,A \to \mathrm{List}\,A
\end{aligned}$$

We can transform this into one constructor function by first uncurrying

$$\begin{aligned}
1 &\to \mathrm{List}\,A \\
A \times \mathrm{List}\,A &\to \mathrm{List}\,A
\end{aligned}$$

and then merging them into one function using coproducts:

$$1 + A \times \mathrm{List}\,A \to \mathrm{List}\,A$$

Hence $\mathrm{List}\,A$ is the initial algebra of the functor $T_{\mathrm{List}\,A} : \underline{\mathbf{Set}} \to \underline{\mathbf{Set}}$ with $T_{\mathrm{List}\,A}\,X = 1 + A \times X$.

[15] I leave the carrier A implicit since it can be inferred from f.

Exercise 56 Define the function $\mathrm{rev}_A : \mathrm{List}\,A \to \mathrm{List}\,A$ using only the iterator. *Hint:* it is useful to define an auxilliary function $\mathrm{snoc} : A \times \mathrm{List}\,A \to A$ that appends a single element at the end of a list using the iterator.

Show that this function is idempotent, that is $\mathrm{rev}_A \circ \mathrm{rev}_A = \mathrm{id}_{\mathrm{List}\,A}$ using only initiality.

Exercise 57 What are the functors defining

1. unlabelled binary trees,
2. binary trees whose leaves are labelled with $A : \mathbf{Set}$,
3. binary trees whose nodes are labelled with $A : \mathbf{Set}$,

Since we already know that List is a functor, what is the initial algebra of the list functor? This has a constructor

$$\mathrm{in}_{\mathrm{List}} : \mathrm{List}\,(\mu\,\mathrm{List}) \to \mu\,\mathrm{List}$$

Indeed, this is the type of finitely branching trees, that is any node has a list of subtrees. We don't need to include leaves explicitely because we can have a node with an empty list of subtrees. This type is also called *rose trees.*

We can also construct infinitely branching trees which are the initial algebra of the functor $T\,X = 1 + \mathbb{N} \to X$. Here a node is given by a function that assigns to every natural number a subtree. This raises the question wether every functor on sets should have an initial algebra. We note that obviously paradoxical cases like $T\,X = X \to X$ are ruled out because they don't give rise to a functor. However, depending on our foundations even positive hence functorial definitions like $T\,X = (X \to \mathrm{Bool}) \to \mathrm{Bool}$ are problematic. If we work in a classical setting then this is obviously paradoxical because it gives us a fixpoint of the double powerset functor and from this we can derive a contradiction using diagonalisation.

Exercise 58 Show that having an initial algebra of $T : \mathbf{Set} \to \mathbf{Set}$ with $T\,X = (X \to \mathrm{Bool}) \to \mathrm{Bool}$ leads to a contradiction if we assume $\mathbf{Prop} = \mathrm{Bool}$ (classical logic). Assume that $\mathrm{in}_T : T(\mu\,T) \to \mu\,T$ is an isomorphism [16]. *Hint:* Derive a retraction, i.e. a function $\phi : (\mu\,T \to \mathrm{Bool}) \to \mu\,T$ that has a left inverse. Show that such a retract cannot exists using Cantor's diagonalisation.

5.3 Lambek's lemma

We return to the predecessor function and show that the constructor morphism $\mathrm{in}_T : \underline{\mathbf{C}}(T\,(\mu\,T), \mu T)$ is an isomorphism for any initial T-algebra. The inverse is given by

$$\begin{aligned} \mathrm{out}_T &: \underline{\mathbf{C}}(\mu T, T\,(\mu\,T)) \\ \mathrm{out}_T &= \mathrm{It}_T\,(T\,\mathrm{in}_T) \end{aligned}$$

[16] We are going to prove this in the next section.

which generalizes our definition of pred. To see that this is an isomorphism, we use the following diagram:

$$\begin{tikzcd}
T\,(\mu T) \arrow[r, "\mathrm{in}_T"] & \mu T \\
T\,(T\,(\mu\,T)) \arrow[u, "T\,\mathrm{in}_T"'] \arrow[r, "T\,\mathrm{in}_T"] & T\,(\mu\,T) \arrow[u, "\mathrm{in}_T"] \\
T\,(\mu T) \arrow[u, "T\,\mathrm{out}_T"'] \arrow[r, "\mathrm{in}_T"] \arrow[uu, bend left, "T\,\mathrm{id}=\mathrm{id}"] & \mu T \arrow[u, "\mathrm{out}_T"] \arrow[uu, bend right, "\mathrm{id}"']
\end{tikzcd}$$

The lower square commutes due to the definition of out_T, the upper square commutes trivially. On the other hand the whole square also commutes with id which on the left equals $T\,\mathrm{id}$. However, since there is at most one algebra morphism, we know that $\mathrm{in}_T \circ \mathrm{out}_T = \mathrm{id}$. Now we can reason that :

$$\begin{aligned}
\mathrm{out}_T \circ \mathrm{in}_T &= T\,\mathrm{in}_T \circ T\,\mathrm{out}_T && \text{lower square}\\
&= T\,(\mathrm{in}_T \circ \mathrm{out}_T)\\
&= T\,\mathrm{id}\\
&= \mathrm{id}
\end{aligned}$$

Hence we have verified that in_T and out_T are an isomorphism.

5.4 Streams

The dual of initial algebras, terminal coalgebras, turn out to be useful as well. An example of a coinductive type is the type of streams $\mathrm{Stream}\,A : \mathbf{Set}$ this are infinite sequences of elements of $A : \mathbf{Set}$, that is

$$[a_0, a_1, a_2 \cdots : \mathrm{Stream}\,A$$

with $a_i : A$. Streams can be understood via destructors, that is for for any stream we can compute its head and its tail:

$$\begin{aligned}
\mathrm{head} &: \mathrm{Stream}\,A \to A\\
\mathrm{tail} &: \mathrm{Stream}\,A \to \mathrm{Stream}\,A
\end{aligned}$$

To construct streams we need a coiterator, that is given $X : \mathbf{Set}$ and functions

$$\begin{aligned}
h &: X \to A\\
t &: X \to X
\end{aligned}$$

we obtain a function:

$$\mathrm{CoIt}\,h\,t : X \to \mathrm{Stream}\,A$$

which is given by the copatterns:

$$\begin{aligned}\mathrm{head}\,(\mathrm{CoIt}\,h\,t\,x) &= h\,x\\ \mathrm{tail}\,(\mathrm{CoIt}\,h\,t\,x) &= \mathrm{CoIt}\,h\,t\,(t\,x)\end{aligned}$$

$\mathrm{CoIt}\,h\,t$ is the unique function that makes the following diagram commute:

$$\begin{tikzcd} & X \arrow[r, "t"] \arrow[dl, "h"'] \arrow[d, dashed, "\mathrm{CoIt}\,h\,t"] & X \arrow[d, dashed, "\mathrm{CoIt}\,h\,t"] \\ A & \mathrm{Stream}\,A \arrow[l, "\mathrm{head}"] \arrow[r, "\mathrm{tail}"] & \mathrm{Stream}|,A \end{tikzcd}$$

As an example we can define the function $\mathrm{from} : \mathbb{N} \to \mathrm{Stream}\,\mathbb{N}$ which generates a stream of natural numbers starting with the input. I.e.

$$\mathrm{from}\,n = [n, n+1, n+2, \ldots$$

We can define from via copatterns

$$\begin{aligned}\mathrm{head}\,(\mathrm{from}\,n) &= n\\ \mathrm{tail}\,(\mathrm{from}\,n) &= \mathrm{from}\,(\mathrm{suc}\,n)\end{aligned}$$

and this can be turned into an application of the coiterator:

$$\mathrm{from} = \mathrm{CoIt}\,\mathrm{id}_{\mathbb{N}}\,\mathrm{suc}$$

Stream is a functor, that is given a function $f : A \to B$ we can define

$$\begin{aligned}\mathrm{Stream}\,f &: \mathrm{Stream}\,A \to \mathrm{Stream}\,B\\ \mathrm{head}\,(\mathrm{Stream}\,f\,\vec{a}) &= f\,(\mathrm{head}\,a)\\ \mathrm{tail}\,(\mathrm{Stream}\,f\,\vec{a}) &= \mathrm{Stream}\,f\,(\mathrm{tail}\,\vec{a})\end{aligned}$$

Exercise 59 Define $\mathrm{Stream}\,f$ using only the coiterator.

We can use uniqeness to prove equations, for example we want to verify

$$\mathrm{from}\,(\mathrm{suc}\,n) = \mathrm{Stream}\,\mathrm{suc}\,(\mathrm{from}\,n)$$

for any number n. We turn this into an equation for functions $\mathbb{N} \to \mathrm{Stream}\,\mathbb{N}$:

$$\mathrm{from} \circ \mathrm{suc} = \mathrm{Stream}\,\mathrm{suc} \circ \mathrm{from}$$

We observe that

$$\begin{aligned}\mathrm{head}\,(\mathrm{from}\,(suc\,n)) &= \mathrm{suc}n\\ \mathrm{head}\,(\mathrm{Stream}\,\mathrm{suc}\,(\mathrm{from}\,n)) &= \mathrm{suc}\,(\mathrm{head}\,(\mathrm{from}\,n)\\ &= \mathrm{suc}\,n\\ \mathrm{tail}\,(\mathrm{from}\,(suc\,n)) &= \mathrm{from}\,(\mathrm{suc}\,(\mathrm{suc}\,n))\\ \mathrm{tail}\,(\mathrm{Stream}\,\mathrm{suc}\,(\mathrm{from}\,n)) &= \mathrm{Stream}\,\mathrm{suc}\,(\mathrm{tail}\,(\mathrm{from}\,n))\\ &= \mathrm{Stream}\,\mathrm{suc}\,(\mathrm{from}\,(\mathrm{suc}\,n))\end{aligned}$$

That is both side are equal to $\mathrm{CoIt}\,\mathrm{suc}\,\mathrm{suc}$ and hence equal due to uniqueness.

Exercise 60 Prove that $\mathrm{Stream}\, A$ is isomorphic to $\mathbb{N} \to A$.

5.5 Terminal coalgebras

In general a terminal coalgebra of an endofunctor $T : \underline{\mathbf{C}} \to \underline{\mathbf{C}}$ is given by an object $\nu T : |\underline{\mathbf{C}}|$ and a morphism $\mathrm{out}_T :: \underline{\mathbf{C}}(\nu\, T, T\,(\nu\, T))$ such that for any T-coalgebra, that is a pair of $A : |\underline{\mathbf{C}}|$ and a morphism $f : \underline{\mathbf{C}}(A, T\, A)$ there is a unique morphism $\mathrm{CoIt}_T\, f : \underline{\mathbf{C}} A \nu T$ that makes the following diagram commute:

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, dashed, "\mathrm{CoIt}_T\, f"] & T\, A \arrow[d, "T\,(\mathrm{CoIt}_T\, f"] \\
\nu\, T \arrow[r, "\mathrm{out}_T"] & T\,(\nu T)
\end{tikzcd}$$

Indeed, the terminal coalgebra is just the terminal object in the category of T-coalgebras, whose objects are T-coalgebras and given T-coalgebras $f : \underline{\mathbf{C}}(A, T\, A)$ and $g : \underline{\mathbf{C}}(B, T\, B)$ a morphism is given by a morphism $h : \underline{\mathbf{C}}(A, B)$ such that the following diagram commutes:

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "h"] & T\, A \arrow[d, "T\, h"] \\
B \arrow[r, "g"] & T\, B
\end{tikzcd}$$

As for algebras, Identity and composition is given by identity and composition in $\underline{\mathbf{C}}$, it is easy to see that the corresponding diagrams commute.

It is clear that $\mathrm{Stream}\, A$ is the terminal coalgebra of the functor $T_{\mathrm{Stream}\, A}\, X = A \times X$. We can look at initial algebras and terminal coalgebras for the same functor. In the case of $T_{\mathrm{Stream}\, A}$ this is not very interesting because the initial algebra is just the empty set. However, natural numbers $T_{\mathbb{N}}\, X = 1 + X$ and lists $T_{\mathrm{List}\, A}\, X = 1 + A \times X$ are more interesting: here the terminal coalgebras are called the conatural numbers and colists: they include both finite elements and infinite elements.

Exercise 61 Show that the initial algebra of $T_{\mathrm{Stream}\, A}\, X = A \times X$ is isomorphic to the empty set

Exercise 62 Define addition for conatural numbers ($\mathbb{N}^\infty = \nu X.1 + X$) using only the coiterator.

Exercise 63 Show explicitely that Lambek's lemma hold for terminal coalgebras.