Coalgebra, lecture 2
Streams and Coinduction

Jurriaan Rot

February 3, 2025

In this lecture we study *streams*, which are just infinite sequences over a given data type. They are generated by stream systems—a simple type of state-based system—which forms a canonical instance of the theory of coalgebras. In particular, we will see that streams arise as a final coalgebra. The coalgebraic view provides a technique for defining streams as well as for proving equivalence. Both the definition and proof technique are referred to here as *coinduction*, and are central to the theory of coalgebras. As such, we use streams in this lecture as a first introduction to coalgebra. For a much more elaborate treatment of streams within the coalgebraic framework, see Rutten's book [1].

1 Defining streams coinductively

Let A be an arbitrary set. A *stream* (over A) is a function $\sigma \colon \mathbb{N} \to A$. The set of all streams over A is denoted by

$$A^\omega = \{\sigma \mid \sigma \colon \mathbb{N} \to A\},$$

ranged over by $\sigma, \tau, \ldots$. So a stream σ is just an infinite sequence, which we could also write as $(\sigma(0), \sigma(1), \sigma(2), \ldots)$. For instance, $(1, 1, 2, 3, 5, 8, \ldots)$ is a stream over $\mathbb{N}$, and $(a, b, a, b, a, b, \ldots)$ is a stream over $\{a, b\}$. We introduce two important pieces of notation and terminology: for a stream σ, write

- the *head* of σ is given by $\sigma(0)$;
- the *tail* of σ is denoted by σ', defined by $\sigma'(n) = \sigma(n+1)$.

The head of a stream is also sometimes called it's *initial value*, and the tail the *derivative*. This follows earlier terminology in automata theory (we'll see the so-called *Brzozowski derivatives* of regular expressions in a later lecture); and there are more direct connections to analysis, that we won't go into, but see [1]. It will sometimes be useful to refer to the "n-th derivative" $\sigma^{(n)}$, defined by $\sigma^{(0)} = \sigma$ and $\sigma^{(n+1)} = (\sigma^{(n)})'$.

Further, given $a \in A$ we write $a : \sigma$ for concatenation:

$$a : \sigma = (a, \sigma(0), \sigma(1), \sigma(2), \ldots).$$

An elementary but useful fact is that, for any stream σ, we have

$$\sigma = \sigma(0) : \sigma' .$$

Using this basic notation, we are now ready to introduce coinductive definitions of streams, also referred to as *stream differential equations*. The idea is to define a stream in terms of its initial value (head) and derivative (tail). The technique is introduced by examples.

Example 1.1. Let $a \in A$, and let σ be the unique stream given by

$$\begin{aligned} \sigma(0) &= a \\ \sigma' &= \sigma \end{aligned}$$

This defines the stream $\sigma = (a, a, a, \ldots)$. Indeed, we compute a few steps:

$$\sigma = \sigma(0) : \sigma' = a : \sigma = a : \sigma(0) : \sigma' = a : a : \sigma = \ldots$$

Similarly, we can define the stream $(a, b, a, b, \ldots)$ as follows:

$$\begin{aligned} \sigma(0) &= a & \qquad \tau(0) &= b \\ \sigma' &= \tau & \qquad \tau' &= \sigma \end{aligned}$$

Indeed, the above has a unique solution: $\sigma = (a, b, a, b, \ldots)$ and $\tau = (b, a, b, a, \ldots)$.

In the above cases, it is clear that the solution exists and is unique. In general, this can be more tricky, as we will see in some examples below. We'll see at the end of the lecture how to justify some definition formats using coalgebras. First, we show how to use this technique to define functions between streams.

Example 1.2. Consider the function $\mathsf{even} \colon A^\omega \to A^\omega$, defined for all $\sigma \in A^\omega$ by:

$$\begin{aligned} \mathsf{even}(\sigma)(0) &= \sigma(0) \\ \mathsf{even}(\sigma)' &= \mathsf{even}(\sigma'') \end{aligned}$$

Once again, we compute a few steps:

$$\begin{aligned} \mathsf{even}(\sigma) &= \mathsf{even}(\sigma)(0) : \mathsf{even}(\sigma)' = \sigma(0) : \mathsf{even}(\sigma'') \\ &= \sigma(0) : \mathsf{even}(\sigma'')(0) : \mathsf{even}(\sigma'')' = \sigma(0) : \sigma(2) : \mathsf{even}(\sigma'''') \\ &= \ldots \end{aligned}$$

Indeed, we have $\mathsf{even}(\sigma) = (\sigma(0), \sigma(2), \sigma(4), \ldots)$. Of course, we can define $\mathsf{odd}(\sigma) = (\sigma(1), \sigma(3), \sigma(5), \ldots)$ in a similar way (how?).

Another classic function is $\mathsf{zip} \colon A^\omega \times A^\omega \to A^\omega$, given by

$$\mathsf{zip}(\sigma, \tau) = (\sigma(0), \tau(0), \sigma(1), \tau(1), \ldots) .$$

We leave its definition in terms of stream differential equations as an exercise (spoiler alert: we'll use that definition in some examples below).

Example 1.3. It's nice to assume some more structure on the data; for instance, taking $A = \mathbb{R}$ we can use stream differential equations to define some operations to compute with streams over real numbers:

$$\begin{aligned} \overline{r}(0) &= r & \overline{r}' &= \overline{0} && \text{(for any } r \in \mathbb{R}) \\ (\sigma \oplus \tau)(0) &= \sigma(0) + \tau(0) & (\sigma \oplus \tau)' &= \sigma' \oplus \tau' \\ (\sigma \otimes \tau)(0) &= \sigma(0) \times \tau(0) & (\sigma \otimes \tau) &= (\sigma' \otimes \tau) \oplus (\overline{\sigma(0)} \otimes \tau') \end{aligned}$$

The stream $\overline{r}$ is given by $(r, 0, 0, 0, \ldots)$; the operator $\oplus$ performs pointwise addition of streams, and the operator $\otimes$ is a kind of product known as the *convolution product*—as we will see later, it is strongly related to the concatenation of formal languages. These operators are part of the *stream calculus*, see, once again, [1].

Note that the above example uses a slightly more complicated format than before. In the first couple examples, such as even and zip, in the derivative we made a single recursive call. Instead, in the convolution product $\otimes$, we also make a call to the operator $\oplus$. The convolution product is still well-defined (it has a unique solution), but this is not immediately clear; we will return to this point later. For instance, consider the "definition"

$$\sigma(0) = 1 \qquad\qquad \sigma' = \mathsf{even}(\sigma)$$

In the derivative, this not only makes a recursive call to σ, but also uses even. In fact, the above does not properly define a stream: it has multiple solutions (which?).

2 Proving equivalence of streams

So far we've seen how to define streams and functions thereon. How to prove that such streams are equal? Consider, for instance, the following identities:

$$\begin{aligned} \mathsf{even}(\mathsf{zip}(\sigma, \tau)) &= \sigma \\ \mathsf{odd}(\mathsf{zip}(\sigma, \tau)) &= \tau \\ \mathsf{zip}(\mathsf{even}(\sigma), \mathsf{odd}(\sigma)) &= \sigma \end{aligned}$$

How to prove such equalities, using their definition in terms of stream differential equations? Well, the heads should be equal:

$$\mathsf{even}(\mathsf{zip}(\sigma, \tau))(0) = \mathsf{zip}(\sigma, \tau)(0) = \sigma(0)$$

So far so good, but we also need to compare the tails. For the left-hand side:

$$\mathsf{even}(\mathsf{zip}(\sigma, \tau))' = \mathsf{even}(\mathsf{zip}(\sigma, \tau)'') = \mathsf{even}(\mathsf{zip}(\tau, \sigma')') = \mathsf{even}(\mathsf{zip}(\sigma', \tau'))\,.$$

Is this equal to σ'? Well, that's exactly the question that we started with! Intuitively the two steps above are enough: we compared the heads, and after one step we're again in the same form (where the heads are equal), so we can make another step, after which the heads are again equal, etc ... Such reasoning is made precise using the central notion of *bisimulation*.

Definition 2.1. A relation $R \subseteq A^\omega \times A^\omega$ on streams is a *bisimulation* if for all $(\sigma, \tau) \in R$ we have:

1. $\sigma(0) = \tau(0)$, and
2. $(\sigma', \tau') \in R$.

Thus, the initial value of streams related by a bisimulation should be equal, and their derivative should again be related.

Example 2.2. The relation

$$R = \{(\mathsf{even}(\mathsf{zip}(\sigma, \tau)), \sigma) \mid \sigma, \tau \in A^\omega\}$$

is a bisimulation:

1. $\mathsf{even}(\mathsf{zip}(\sigma, \tau))(0) = \sigma(0)$ (as we've seen above), and
2. $\mathsf{even}(\mathsf{zip}(\sigma, \tau))' = \mathsf{even}(\mathsf{zip}(\sigma', \tau'))\ R\ \sigma'$.

The point of bisimulations is that they characterise equality of streams, as shown in the following theorem.

Theorem 2.3 (Coinduction proof principle)**.** *If R is a bisimulation, then for all $(\sigma, \tau) \in R$: $\sigma = \tau$.*

Proof. Outline: prove by induction that

$$(\sigma^{(i)}, \tau^{(i)}) \in R \tag{1}$$

for all $i \in \mathbb{N}$. Then we obtain:

$$\sigma(i) = \sigma^{(i)}(0) = \tau^{(i)}(0) = \tau(i)$$

for all i, using (1) in the middle equality. Hence $\sigma = \tau$. □

Example 2.4. Since R in Example 2.2 is a bisimulation, we obtain $\mathsf{even}(\mathsf{zip}(\sigma, \tau)) = \sigma$ for any $\sigma \in A^\omega$.

Example 2.5. Let $R = \{(\mathsf{zip}(\mathsf{even}(\sigma), \mathsf{odd}(\sigma)), \sigma) \mid \sigma \in A^\omega\}$. Is this a bisimulation? We check:

1. $\mathsf{zip}(\mathsf{even}(\sigma), \mathsf{odd}(\sigma))(0) = \mathsf{even}(\sigma)(0) = \sigma(0)$, and
2. $\mathsf{zip}(\mathsf{even}(\sigma), \mathsf{odd}(\sigma))' = \mathsf{zip}(\mathsf{odd}(\sigma), \mathsf{even}(\sigma''))$. But the latter is not related to σ' by R ...

So the above is not a bisimulation. We can solve this by adding the necessary pair to a new relation R':

$$R' = R \cup \{(\mathsf{zip}(\mathsf{odd}(\sigma), \mathsf{even}(\sigma'')), \sigma') \mid \sigma \in A^\omega\}$$

The result is a bisimulation (little exercise). So we conclude $\mathsf{zip}(\mathsf{even}(\sigma), \mathsf{odd}(\sigma)) = \sigma$ for all $\sigma \in A^\omega$.

We'll see more examples of the use of bisimulations later on in the course; and also more powerful techniques, which become needed when proving equalities between stream differential equations that are less elementary than zip, even etc. This also appears in one of the exercise of this week. If you're interested: a nice application worked out in [1] is *Moessner's theorem*, which has an elegant proof using bisimulations.

3 Stream systems

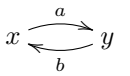
We now present streams coalgebraically. More precisely, we'll discuss a concrete type of coalgebra—stream systems—where streams arise as the final coalgebra.

A *stream system* (over a set A) consists of a set X of states, an output function $o\colon X \to A$ and a transition function $t\colon X \to X$. It is typically written as a coalgebra

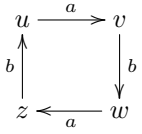
$$\langle o, t\rangle\colon X \to A \times X\,,$$

where $\langle o, t\rangle(x) = (o(x), t(x))$ (called the *pairing* of the functions o and t).

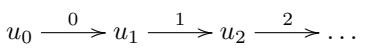
Stream systems can be depicted graphically, for instance:



(2)



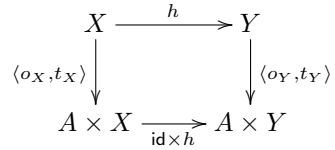
(3)



(4)

The states of a stream system generate streams, by following the unique path and reading labels: e.g. x generates $ababab\ldots$. This will be made more precise in a moment.

A *homomorphism* from a stream system $(X, \langle o_X, t_X\rangle)$ to a stream system $(Y, \langle o_Y, t_Y\rangle)$ is a map $h\colon X \to Y$ such that the following diagram commutes:



where $(\mathsf{id} \times h)(a, x) = (a, h(x))$. So, in terms of equations, h is a homomorphism if for all $x \in X$:

1. $o_X = o_Y \circ h$, and
2. $h \circ t_X = t_Y \circ h$.

And, in terms of arrows: if $x \xrightarrow{a} y$ then $h(x) \xrightarrow{a} h(y)$.

Example 3.1. There is a homomorphism from the stream system in (3) to the one in (2), but not vice versa (why?).

We now introduce a super important stream system: the one whose states consists of streams! The output and transition structure are defined using initial value and derivative. Explicitly, we define $(A^\omega, \langle i, d\rangle)$ where

$$\begin{aligned} i(\sigma) &= \sigma(0) \\ d(\sigma) &= \sigma' \end{aligned}$$

for all $\sigma \in A^\omega$. For a stream $\sigma \in A^\omega$, we can depict this as:

$$\sigma \xrightarrow{\sigma(0)} \sigma' \xrightarrow{\sigma(1)} \sigma'' \xrightarrow{\sigma(2)} \dots$$

If we start in σ and read labels of arrows one by one, following the (single) outgoing path, we get $\dots\sigma$. That is, the "behaviour" of a stream σ, as a state of the above stream system, is the stream itself! Note that $\langle i, d\rangle$ is a bijection, and it performs precisely the decomposition of a stream in terms of head and tail that we mentioned early on in the lecture.

Theorem 3.2. *The stream system $(A^\omega, \langle i, d\rangle)$ is* final*: for any stream system $(X, \langle o, t\rangle)$ there is a unique homomorphism h from $(X, \langle o, t\rangle)$ to $(A^\omega, \langle i, d\rangle)$:*

$$\begin{array}{ccc} X & \overset{h}{\dashrightarrow} & A^\omega \\ {\scriptstyle \langle o,t\rangle}\downarrow & & \downarrow{\scriptstyle \langle i,d\rangle} \\ A \times X & \xrightarrow[\mathsf{id}\times h]{} & A \times A^\omega \end{array}$$

Proof. Define h by $h(x) = (o(x), o(t(x)), o(t(t(x))), \dots)$, more formally:

$$h(x)(n) = o(t^n(x))$$

for all $n \in \mathbb{N}$.

First, we check that h is a homomorphism. We need to prove, for all x:

1. $o(x) = i(h(x))$, and
2. $h(t(x)) = d(h(x))$.

The first is immediate by definition of h and i: we have $i(h(x)) = h(x)(0) = o(x)$. The second by definition of h and d: for all n, we have

$$\begin{aligned} d(h(x))(n) &= h(x)'(n) \\ &= h(x)(n+1) \\ &= o(t^{n+1}(x)) \\ &= o(t^n(t(x))) \\ &= h(t(x))(n)\,. \end{aligned}$$

Thus, h is a homomorphism indeed.

To show that it is unique, suppose that h, k are both homomorphisms. One can then prove $\forall x \in X.\, h(x)(n) = k(x)(n)$ directly by induction on n (exercise). Alternatively, one can use Theorem 2.3 and show that the relation

$$R = \{(h(x), k(x)) \mid x \in X\}$$

is a bisimulation. □

The above result constitutes both a mechanism for *defining* streams, through the unique homomorphism into the final stream system, and for proving their equality, using the uniqueness. In the remainder we show how this recovers the definition and proof methods of the first part of this lecture.

Consider, for instance, the stream system in (2). The unique homomorphism h to the final stream system is characterised by:

$$\begin{aligned} h(x)(0) &= a & \qquad h(y)(0) &= b \\ h(x)' &= h(y) & \qquad h(y)' &= h(x) \end{aligned}$$

That is, it maps x to the stream $h(x) = (a, b, a, b, a, b, \ldots)$ and y to $h(y) = (b, a, b, a, b, a, \ldots)$. This is precisely the definition by stream differential equations that we gave in Example 1.1. Indeed, defining a stream by a stream differential equations amounts to giving a *stream system*; the unique solution is then given by the unique homomorphism to the final coalgebra.

For the definition of *functions* from streams to streams, such as even, we again make use of finality of A^ω. For instance, by making a careful choice of stream system on A^ω, namely the following:

$$\begin{aligned} o(\sigma) &= \sigma(0) \\ t(\sigma) &= \sigma'' \end{aligned}$$

the function $\mathsf{even}\colon A^\omega \to A^\omega$ arises as the unique homomorphism:

$$\begin{array}{ccc} A^\omega & \xrightarrow{\mathsf{even}} & A^\omega \\ {\scriptstyle \langle o,t \rangle}\downarrow & & \downarrow{\scriptstyle \langle i,d \rangle} \\ A \times A^\omega & \xrightarrow[\mathsf{id} \times \mathsf{even}]{} & A \times A^\omega \end{array}$$

Indeed, spelling out commutativity of the diagram yields precisely the definition of even in Example 1.2.

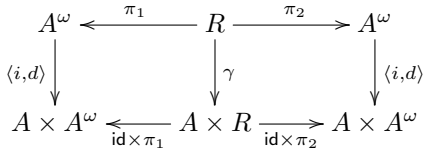
Similarly, $\mathsf{zip}\colon A^\omega \times A^\omega \to A^\omega$ arises by giving a suitable stream system on $A^\omega \times A^\omega$, corresponding to the associated stream differential equations. Thus, the finality result in Theorem 3.2 gives a basic scheme for defining streams and functions between them through stream differential equations. In conclusion: a (basic) definition by stream differential equations *is* a stream system, and its unique solution is given by the unique homomorphism to the final coalgebra.

Note that not all interesting functions fit this format, though. For instance, the convolution product in Example 1.3 does not immediately fit. In fact, convolution product works because the additional operation used in its definition is pointwise sum; which has good properties. Indeed, one can extend the basic coinduction principle to a richer format; we will postpone this to a later moment in the course.

We conclude with a brief discussion of the *proof principle* arising from Theorem 3.2. More specifically, the coinduction proof principle comes from the fact that the homomorphisms into the final coalgebra are unique (conversely, one can first prove Theorem 2.3 and use it to prove that such homomorphism are unique, as we also commented in the proof of Theorem 3.2).

The first step is to characterise bisimulations differently. Recall that any relation $R \subseteq X \times Y$ comes equipped with projections $\pi_1 \colon R \to X$ and $\pi_2 \colon R \to Y$, given by $\pi_1(x, y) = x$ and $\pi_2(x, y) = y$ respectively.

Lemma 3.3. *A relation $R \subseteq A^\omega \times A^\omega$ is a bisimulation if and only if there exists a stream system $\gamma \colon R \to A \times R$ such that the projections $\pi_1, \pi_2 \colon R \to A^\omega$ are homomorphisms, i.e., the following diagram commutes:*



Now, if R is a bisimulation presented as in the above lemma, then by Theorem 3.2 we get $\pi_1 = \pi_2$, that is, $(\sigma, \tau) \in R$ implies $\sigma = \tau$. We thus recover Theorem 2.3 (note that, conversely, one could use Theorem 2.3 in the proof of Theorem 3.2, but that's not necessary of course).

Thus, the final stream system gives us the coinductive definition and proof principles of the first half of this lecture. As we will soon see, the notion of coalgebra generalises all this from stream systems to a wide class of state-based systems.

References

[1] J. Rutten. The method of coalgebra: exercises in coinduction. CWI Technical report, available at: `https://ir.cwi.nl/pub/28550/`, 2019.