Coalgebra, lecture 13: Induction; coinduction in lattices and categories

Jurriaan Rot

December 5, 2016

In the first part of this lecture, we'll have a look at induction in the setting of lattices (Sections 1 and 2). In the second part, the lattice-theoretic perspective is related to the coalgebraic perspective (Sections 3 and 4).

1 Processes with a finite trace

We start with a basic example of an inductive predicate.[1] Assume a labelled transition system $(X, \to)$ over a set of labels A. We'll refer to the elements $p \in X$ as processes (or states). A process p is said to be *stopped* if it has no outgoing transition. We would like to define a predicate $\downarrow$ on processes (states $p \in X$), such that $p \downarrow$ holds if there exists a path to a stopped state. This predicate should is characterised by the following rules:

$$\frac{p \text{ stopped}}{p\downarrow} \qquad \frac{p \xrightarrow{a} p' \quad p' \downarrow}{p \downarrow}$$

(actually, on the right we have one rule for each $a \in A$). We interpret these rules *inductively*: the set of processes which satisfy $p \downarrow$ is the least set satisfying the above two rules, read from top to bottom: if a process p is stopped, or it transitions to a state p' with $p' \downarrow$, then $p \downarrow$.

This is formalised in terms of the lattice $\mathcal{P}(X)$ of predicates (subsets) on X, with the inclusion order; the join is given by union, and the meet by intersection. Consider $b_\downarrow \colon \mathcal{P}(X) \to \mathcal{P}(X)$ defined by

$$b_\downarrow(P) = \{p \in X \mid p \text{ is stopped, or } p \xrightarrow{a} p' \text{ and } p' \in P \text{ for some } a \in A, p' \in P\}$$

A *pre-fixed point* of $b_\downarrow$ is a predicate P such that $b_\downarrow(P) \subseteq P$. This means that

- $p \in P$ for all stopped processes p, and

[1] The examples in this section are taken from: D. Sangiorgi, Introduction to Bisimulation and Coinduction, Cambridge Universty Press.

- if $p \xrightarrow{a} p'$ and $p' \downarrow$, then $p \in P$.

Our predicate $p \downarrow$ is the *least pre-fixed point* (actually, the least fixed point $\mathsf{lfp}(b_\downarrow)$) of $b_\downarrow$. Since $p \downarrow$ is defined in this way, we call it an *inductive predicate*.

Just like for coinduction, we get an inductive proof principle from the fact that the set of processes satisfying $p \downarrow$ is the least pre-fixed point: it says that, if $f(P) \subseteq P$, then for all $p \in X$: if $p \downarrow$ then $p \in P$.

As an example of such a proof, consider a partial function $f\colon X \to \mathbb{N}$ defined by $f(p) = 0$ if p is stopped, and otherwise

$$f(p) = \min\{f(p') + 1 \mid p \xrightarrow{a} p' \text{ for some } a, p' \text{ s.t. } f(p') \text{ is defined}\}\,.$$

The set of which we're taking a minimal element may be empty; in that case, $f(p)$ is undefined. We write $\mathsf{dom}(f)$ for the domain of f, that is, the subset of X on which f is defined. Claim: if $p \downarrow$, then $p \in \mathsf{dom}(f)$. To prove this, we should use that $p \downarrow$ is defined by induction. Formally, this means showing that $b_\downarrow(\mathsf{dom}(f)) \subseteq \mathsf{dom}(f)$. So we should show:

- $p \in \mathsf{dom}(f)$ for all stopped processes p, and
- if $p \xrightarrow{a} p'$ and $p'\mathsf{dom}(f)$, then $p \in \mathsf{dom}(f)$.

Both follow directly from the definition of f. Hence $\mathsf{dom}(f)$ is a pre-fixed point of $b_\downarrow$, so that for all p: if $p \downarrow$ then $p \in \mathsf{dom}(f)$.

1.1 Infinite paths

It is nice to contrast this to a coinductive definition of the existence of an infinite path of processes, which we denote by $p \uparrow$. We'd like to define this by the rule

$$\frac{p \xrightarrow{a} p' \qquad p' \uparrow}{p \uparrow}$$

(again, actually one rule for all $a \in A$). The inductive interpretation is not quite right here: it gives the empty set! Because that's the least set which is closed under the rule (read from top to bottom).

Instead, we should use a coinductive interpretation: the set of processes satisfying $p \uparrow$ is the *greatest* subset of X such that *every process in it is justified by the above rule*. More concretely, we mean, it is the greatest predicate such that

- if $p \uparrow$ then there is a $p' \in X$ such that $p \xrightarrow{a} p'$ for some a, and $p' \uparrow$.

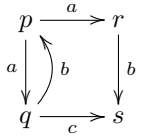
This is quite a different reading of the inference rule than in the inductive interpretation.

Formally, define $b_\uparrow\colon \mathcal{P}(X) \to \mathcal{P}(X)$ as the monotone function given by

$$b_\uparrow(P) = \{p \in P \mid \text{there is a } p' \in X \text{ such that } p \xrightarrow{a} p' \text{ for some } a, \text{ and } p' \in P\}$$

The set of processes satisfying $p \uparrow$ is the greatest (post-)fixed point of $b_\uparrow$.

So to prove that some process has an infinite path, it suffices to prove that it is contained in a set P which is a post-fixed point: $P \subseteq b_\uparrow(P)$. Consider the following transition system:



Here $\{p, q\}$ is the (only) predicate such that $\{p, q\} \subseteq b_\uparrow(\{p, q\})$. Hence $x \uparrow$ iff $x = p$ or $x = q$.

2 Induction in a lattice

The first example highlights the main points of induction in a lattice. The starting point is a monotone function $b\colon P \to P$ on a complete lattice P.

We already know that a *fixed point* (of b) is an element $x \in P$ such that $b(x) = x$, and a post-fixed point is an element x such that $x \leq b(x)$. It shouldn't come as a big surprise that a pre-fixed point is an element x such that $b(x) \leq x$.

The main concept of interest for *coinduction* in a lattice is post-fixed points; the main concept for induction is pre-fixed points. In particular, we sometimes speak about the *inductive predicate* defined by b, which we define to be the least fixed point $\mathsf{lfp}(b)$ of b.

This least fixed point always exists, since b is a monotone function on a complete lattice (dual to what we've seen last week), and it is the *least pre-fixed point*. So, abstractly, we have the following principle:

$$\frac{b(x) \leq x}{\mathsf{lfp}(b) \leq x}$$

that is, any pre-fixed point is below the least fixed point. We can think of this as an inductive proof principle.

At this abstract level, the duality with coinduction becomes clear: there, we look at $\mathsf{gfp}(b)$, which we sometimes call the *coinductive predicate* defined by b. There, the proof principle looks as follows:

$$\frac{x \leq b(x)}{x \leq \mathsf{gfp}(b)}$$

So induction and coinduction are, respectively, the *least* or the *greatest* fixed point of a monotone function b. Rephrased a little, they are respectively the least or greatest solution of the equation $b(x) = x$. The *inductive* predicate is also the least solution of the *inequation* $b(x) \leq x$, the *coinductive* predicate the greatest solution of the the inequation $x \leq b(x)$; these are the proof principles. The proof principle for induction allows you to prove that the inductive predicate lies *below* something of interest, the proof principle for coinduction that the coinductive predicates lies *above* some object of interest.

2.1 Induction on natural numbers

Of course, the most well-known instance of induction, is induction on the natural numbers. We should be able to recover that in the setting of complete lattices. The starting point is the lattice $\mathcal{P}(\mathbb{N})$ of natural numbers, ordered by inclusion.

Then we define $b\colon \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ by $b(P) = \{0\} \cup \{n+1 \mid n \in P\}$. You may convince yourself that the least fixed point of b is the set of natural numbers. And the inductive proof principle tells us that, for any set $P \subseteq \mathbb{N}$, if $b(P) \subseteq P$ then $\mathbb{N} \subseteq P$. But $b(P) \subseteq P$ simply means:

- $0 \in P$;
- if $n \in P$, then $n+1 \in P$,

which should look familiar. So to prove a property be induction, we let $P = \{n \mid n$ satisfies some property about the natural numbers that we want to show$\}$ and prove that it is a pre-fixed point by checking the above two points.

2.2 Induction (and coinduction) for languages

In this example, we're going to look at structural induction on words, for a simple example of languages. We'll also use coinduction, in a somewhat curious way: in this example, induction and coinduction define the same language. But the proof principles are different, and serve a different purpose.

Consider the context-free grammar $S \to aSa \mid bSb \mid a \mid b \mid \varepsilon$. It generates the palindromes. Let's prove that formally, by induction and coinduction. First, we slightly rephrase the problem and look at the equation

$$L = aLa + bLb + a + b + 1$$

on languages over $A = \{a, b\}$, that is, the set $\mathcal{P}(A^*)$. It has a unique solution, which is the language generated by the grammar. We reformulate this equation as

$$L = f(L)$$

for a function $f\colon \mathcal{P}(A^*) \to \mathcal{P}(A^*)$ on languages, defined by

$$f(K) = \{awa \mid w \in K\} \cup \{bwb \mid w \in K\} \cup \{\varepsilon, a, b\}\,.$$

So our language L is the *unique* fixed point of f. This means it's both the least and the greatest: and we can use both induction and coinduction to prove things about it.

Now, let P be the language of palindromes: we would like to prove that $P = L$, which we do by showing two inclusions.

1. For $L \subseteq P$, we use induction: if we show $f(P) \subseteq P$, then $L \subseteq P$. Showing $f(P) \subseteq P$ means:

(a) if $w \in P$, then $awa \in P$ and $bwb \in P$;

(b) $\varepsilon, a, b \in P$.

Both are clear.

2. For $P \subseteq L$, we use coinduction: if we show $P \subseteq f(P)$, then $P \subseteq L$. This means that given a palindrome w (over the alphabet $A = \{a, b\}$), we either have

   (a) $w = ava$ or $w = bvb$ for some palindrome v, or

   (b) w is the empty word, or a single letter.

   This follows from the definition of palindromes.

This concludes the proof that $L = P$, that is, L really is the language of palindromes. We've used induction to prove that everything in L is indeed a palindrome, and coinduction to prove that every palindrome is the language defined by the grammar.

3 Lattice-theoretic coinduction and coalgebras

In the course, we've talked a lot about coinduction; but it still depends a bit on the setting what we mean with it. In any case, we have two formalisations of coinduction: the first in terms of final coalgebras, where the final coalgebra allows us, for instance, to define operations on streams or languages, and prove behavioural equivalence. In this context, we may refer informally to coinduction as the use of finality of coalgebras. The second formalisation of coinduction is in terms of fixed points and lattices, which allows us to define coinductive predicates in a rather flexible way.

It turns out that the lattice-theoretic perspective on coinduction is a special case of the coalgebraic one. To see this, first recall that any partial order (so in particular any complete lattice) $(P, \leq)$ defines a category, whose set of objects is just the set P, and where there is a (unique) arrow $x \to y$ iff $x \leq y$.

If we take partial orders P and Q, seen as a category, then a *functor* from P to Q boils down to a monotone function $b \colon P \to Q$ (if you don't see this, it's useful to work out the details as an exercise!).

Now, take a functor (monotone function) $b \colon P \to P$ on a single partial order, seen as a category. A *coalgebra* for such a functor (monotone function) is, by just instantiating the definition, an object x with an arrow $x \to b(x)$. So that means, an element $x \in P$ such that $x \leq b(x)$. So coalgebras are post fixed points!

Again, just instantiating the definitions, we find that a *final* coalgebra is precisely a *greatest post-fixed point* of b. Again, this is not immediate, and it is useful to work out the details. So the coinductive proof principle in lattices turns out to be a special case of the use of final coalgebras.

All in all, we've seen two main uses of coalgebras, which carry a significantly different intuition.

1. Coalgebras as *state-based systems*: for instance, coalgebras for $B\colon \mathsf{Set} \to \mathsf{Set}$, $B(X) = 2 \times X^A$ (deterministic automata).

2. Coalgebras as *(proofs of) coinductive predicates*: coalgebras as post fixed points of a monotone function.

So shouldn't we look at lattices at all and just work with coalgebras, since it's more general anyway? This would be a wrong conclusion: as we've seen, coinduction in lattices give us a lot of flexibility, and also are very useful as an intuition. And they may just suffice for certain purposes. Further, they can be a source of inspiration for coalgebraic constructions, as we will see later.

4 Bisimulations and relation lifting

In the final part of this lecture, we look at another connection between the theory of coalgebras and the lattice-theoretic picture of coinduction. The point is this: we know well what a bisimulation between coalgebras is: a relation R with a coalgebra structure on it, turning its projections into coalgebra homomorphisms. But we've also seen that we can capture a bisimulation, on automata and streams at least, as a post-fixed point: a relation R such that $R \subseteq b(R)$ for some b. We next show how to systematically define such a b from the functor at hand: this gives another version of coalgebraic bisimulations, in terms of post fixed points.

The main categorical tool is the notion of *relation lifting*. The idea is that, given a functor $B\colon \mathsf{Set} \to \mathsf{Set}$ (whose coalgebras we're interested in) and a relation $R \subseteq X \times X$, we would like to construct a relation on $B(X)$. Formally, given a relation $R \subseteq X \times X$, we define

$$\begin{aligned} &\mathsf{Rel}_B(R) \subseteq B(X) \times B(X) \text{ by} \\ &\mathsf{Rel}_B(R) = \{((B\pi_1)(z), (B\pi_2)(z)) \mid z \in B(R)\} \end{aligned}$$

where π_1, π_2 are the projections of R. This means we apply $B\pi_1$ and $B\pi_2$ to all the stuff that's in $B(R)$. Typically, things in $B(R)$ will look like things that have pairs in them. And $B\pi_1$ and $B\pi_2$ take the first, respectively the second projection of all those pairs, given things in BX. These are the pairs in $\mathsf{Rel}_B(R)$.

For example, take our familiar functor of streams: $B(X) = A \times X$. Then

$$\begin{aligned} \mathsf{Rel}_B(R) &= \{((\mathsf{id} \times \pi_1)(z), (\mathsf{id} \times \pi_2)(z)) \mid z \in A \times R\} \\ &= \{((a, x), (a, y)) \mid (a, (x, y)) \in A \times R\} \\ &= \{((a, x), (a, y)) \mid a \in A,\ (x, y) \in R\} \end{aligned}$$

Now, let $\langle o, f\rangle\colon X \to A \times X$ be stream system, and let Rel_X be the lattice of relations on X. We define $b_{\langle o,f\rangle}\colon \mathsf{Rel}_X \to \mathsf{Rel}_X$ by:

$$b_{\langle o,f\rangle}(R) = \{(x, y) \mid (\langle o, f\rangle(x), \langle o, f\rangle(y)) \in \mathsf{Rel}_B(R)\}\,.$$

Then, after a bit of careful staring at definitions, we see

$$b_{\langle o,f\rangle}(R) = \{(x,y) \mid o(x) = o(y) \text{ and } (f(x), f(y)) \in R\}\,.$$

That's a familiar function to those who've done the exercise last week: we have $R \subseteq b_{\langle o,f\rangle}$ if and only if R is a bisimulation on streams.

The nice thing is that this construction works not only for streams, but for any functor (on Set). In general, given a functor $B\colon \mathsf{Set} \to \mathsf{Set}$, and a coalgebra $f\colon X \to B(X)$, we define $b_f\colon \mathsf{Rel}_X \to \mathsf{Rel}_X$ by

$$b_f(R) = \{(x,y) \mid (f(x), f(y)) \in \mathsf{Rel}_B(R)\}\,.$$

This gives us a notion of bisimulation: we say $R \subseteq X \times X$ is a *Hermida-Jacobs bisimulation* (after Claudio Hermida and Bart Jacobs) if $R \subseteq b_f(R)$.

And we say R is an Aczel-Mendler bisimulation on a (single) coalgebra $f\colon X \to B(X)$ if it is a bisimulation in the sense we know: if there exists a coalgebra $r\colon R \to B(R)$ such that the following diagram commutes.

$$\begin{array}{ccccc} X & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & X \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle r} & & \downarrow{\scriptstyle f} \\ B(X) & \xleftarrow[B(\pi_1)]{} & B(R) & \xrightarrow[B(\pi_2)]{} & B(X) \end{array}$$

The following result establishes the connection.

Theorem 4.1. *Let $B\colon \mathsf{Set} \to \mathsf{Set}$ be a functor, and $f\colon X \to B(X)$ a coalgebra. A relation $R \subseteq X \times X$ is a Hermida-Jacobs bisimulation if and only if it is an Aczel-Mender bisimulation.*

Proof. Exercise (relax, it's an optional one). The main tool is to observe that there is a *surjective* map e making the following diagram commute:

$$\begin{array}{ccccc} B(X) & \xleftarrow{B(\pi_1)} & B(R) & \xrightarrow{B(\pi_2)} & B(X) \\ & {\scriptstyle \pi_1'}\nwarrow & \downarrow{\scriptstyle e} & \nearrow{\scriptstyle \pi_2'} & \\ & & \mathsf{Rel}_B(R) & & \end{array}$$

where π_1', π_2' are the projections of $\mathsf{Rel}_B(R)$. □

One of the nice things about relation lifting is that it's possible to concretely define it for large classes of functors, such as the polynomial ones. These can then be used to derive concrete descriptions of bisimulations for such functors.

We now have two equivalent definitions of bisimulations; for Set functors, at least. These are somewhat different in nature: one of them views a bisimulation as a relation with a *"transition" structure* on it (Aczel-Mendler), while the other views a bisimulation as a relation with a *property* (Hermida-Jacobs). For a lot more information on Hermida-Jacobs bisimulations, and the relation with Aczel-Mendler bisimulations, have a look at Chapter 3 of Jacobs's book.