Chapter 10

Coinductive Definition: A Counterpoint to Inductive Definition

10.1 Introduction

Up to now we have made substantial and effective use of inductive definitions. We have used them to describe the abstract syntax of our languages, as well as their operational semantics. We've used them to justify equational schemes for defining total functions that map inductively-defined sets to arbitrary other sets. And, of course, we've exploited proof by induction to prove properties of inductively-defined sets, sometimes of every element of the set, and other times a single element of the set (by ruling out all others).

Indeed, induction is a workhorse of PL Theory, but it is not the only tool at our disposal, nor is it the best definition and proof method for all situations. During the mid-twentieth century, researchers in computer science and logic nearly simultaneously came across another definition principle, and corresponding proof method, that is closely related to induction, but better suited for some circumstances [Sangiorgi, 2009]. This definition method, viewed as a counterpoint to inductive definitions, has been given the name *coinductive*.[1]

These notes introduce coinductive definition and proof by coinduction, and show how they can be used to quite cleanly define sets and prove properties for which the inductive approach can be awkward.

10.2 Inductive Definition Without Derivations

As far as mathematical techniques go, coinduction is quite young and still under development, but even a little bit of it can go a long way. To lead you into understanding it though, I think it helps to first give an alternative introduction to inductive definition and proof that is equivalent to the derivation-based approach that we've covered so far, but makes it easier to show why coinduction really is a counterpoint to induction, and make clear that neither of them are magical: it's just set theory in action. Along the way, though, you'll ideally get a more transparent understanding of where our principles of induction come from, essentially by seeing how we could prove them correct.

10.2.1 Rules Rule

As before, we begin with inductive rules, but what's interesting here is that we will *not* use rules to define derivations. Let's take as our running example our definition of big-step evaluation for the Boolean language.

[1] In this context and many others in math and logic, the prefix "co-" is short for "counter-", in contrast to its use in e.g., "cohabit". It refers to a recurring and useful phenomenon called *duality*: that many concepts in mathematics have a "natural" counterpart concept.

$\boxed{\Downarrow \subseteq \textsc{Term} \times \textsc{Value}}$

$$\frac{}{\mathsf{true} \Downarrow \mathsf{true}}\ (\text{etrue}) \qquad \frac{}{\mathsf{false} \Downarrow \mathsf{false}}\ (\text{efalse}) \qquad \frac{t_1 \Downarrow \mathsf{true} \quad t_2 \Downarrow v_2}{\mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \Downarrow v_2}\ (\text{eif-t})$$

$$\frac{t_1 \Downarrow \mathsf{false} \quad t_3 \Downarrow v_3}{\mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \Downarrow v_3}\ (\text{eif-f})$$

As before, each rule represents the set of all of its instances: we combine the rules to form one big set:

$$\mathcal{R}_{\Downarrow} = (\text{etrue}) \cup (\text{efalse}) \cup (\text{eif-t}) \cup (\text{eif-f}).$$

To review, our usual approach is to use the rule set $\mathcal{R}_{\Downarrow}$ to form the set of derivations $\textsc{Deriv}[\mathcal{R}_{\Downarrow}]$, and appeal to the $:: \in \textsc{Deriv}[\mathcal{R}_{\Downarrow}] \to \textsc{Term} \times \textsc{Value}$ relation (which happens to be a total function) to determine which judgment a derivation justifies. These in turn help us define our intended set:

$$(\cdot \Downarrow \cdot) = \{\, \langle t, v\rangle \in \textsc{Term} \times \textsc{Value} \mid \exists \mathcal{D} \in \textsc{Deriv}[\mathcal{R}_{\Downarrow}].\, \mathcal{D} :: \langle t, v\rangle \,\}\,.$$

Let's not do that. Instead let's do something different that gets us to the same place at the end. We will use the same rule set $\mathcal{R}_{\Downarrow}$ to define a function on $\textsc{Term} \times \textsc{Value}$ subsets $S \in \mathcal{P}(\textsc{Term} \times \textsc{Value})$.

$$\mathcal{M}[\mathcal{R}_{\Downarrow}] : \mathcal{P}(\textsc{Term} \times \textsc{Value}) \to \mathcal{P}(\textsc{Term} \times \textsc{Value})$$
$$\mathcal{M}[\mathcal{R}_{\Downarrow}](S) = \left\{ \langle t, v\rangle \in \textsc{Term} \times \textsc{Value} \;\middle|\; \exists \frac{S'}{\langle t, v\rangle} \in \mathcal{R}_{\Downarrow}.\, S' \subseteq S \right\}$$

As for $\textsc{Deriv}$, let's just call the function $\mathcal{M}$ on the assumption that we know which rule set $\mathcal{R}_{\Downarrow}$ is relevant. It's pretty clear that the above is a legitimate function definition so we won't worry too much about that: let's consider its "behaviour". In essence, given some set of judgments S, it produces the set of judgments S_0 that can be deduced from S using one step of forward reasoning, as embodied by the rules in $\mathcal{R}_{\Downarrow}$.

Let's consider some equations that this definition justifies:

$$\mathcal{M}(\emptyset) = \{\, \langle \mathsf{true}, \mathsf{true}\rangle\,, \langle \mathsf{false}, \mathsf{false}\rangle \,\}$$

Indeed the only rules $\dfrac{S'}{\langle t, v\rangle}$ where $S' \subseteq \emptyset$ are the "axioms", where $S' = \emptyset$.

$$\mathcal{M}(\{\, \langle \mathsf{false}, \mathsf{true}\rangle \,\}) = \{\, \langle \mathsf{true}, \mathsf{true}\rangle\,, \langle \mathsf{false}, \mathsf{false}\rangle \,\} \cup \{\, \langle \mathsf{if\ false\ then\ false\ else}\ t_3, \mathsf{true}\rangle \in \textsc{Term} \times \textsc{Value} \mid t_3 \in \textsc{Term} \,\}$$

Since $\emptyset \subseteq \{\, \langle \mathsf{false}, \mathsf{true}\rangle \,\}$, we get all the elements of $\mathcal{M}(\emptyset)$ plus some extras, justified by instances of the (eif-t) rule where $t_1 = t_2 = \mathsf{false}$ and $v_2 = \mathsf{true}$: any term t_3 will do! These extras are ludicrous, but what do you expect, we started with a ludicrous assumption: garbage-in garbage-out!

Let's make a further observation, based on the first example above. $\mathcal{M}(\emptyset)$ yields the set of all judgments that could be justified with a derivation $\mathcal{D}$ of height one: just axioms. With a little thought, we could see that $\mathcal{M}^2(\emptyset) \equiv \mathcal{M}(\mathcal{M}(\emptyset))$ produces the set of all judgments that could be justified by a derivation $\mathcal{D}$ of height two *or less*. The "or less" part comes because $\emptyset \subseteq \mathcal{M}(\emptyset)$ so the corresponding rules are still in play. We could keep iterating this process over and over, and each $\mathcal{M}^n(\emptyset)$ would give us the set of all judgments that can be justified with a derivation of height n or less. Since *every* derivation that we consider for inductive definitions has finite height,[2] we can prove that:

$$\Downarrow = \bigcup_{n \in \mathbb{N}} M^n(\emptyset).$$

[2]Technically there is such thing as an inductive definition justified by a derivation of *infinite* height, but it is still "bounded" by something called an *ordinal number*, which you can think of as a number on steroids. We won't be delving into that crazy town in this class, so worry not!

Mathematically, $\bigcup_{n\in\mathbb{N}} M^n(\emptyset)$ is shorthand notation for the expression:

$$\bigcup \{\, S \in \mathcal{P}(\textsc{Term} \times \textsc{Value}) \mid \exists n \in \mathbb{N}.\, S = M^n(\emptyset) \,\},$$

which is justified by the axiom schema of separation and the axiom of union.

This is one of those examples of crazy set-theoretic gymnastics sometimes necessary to get stuff done. What's crazy about this? Well, first we can prove the following theorem, which will be important shortly.

Proposition 41 ($\mathcal{M}$ is monotone with respect to $\subseteq$)**.** *If $S_1 \subseteq S_2$ then $\mathcal{M}(S_1) \subseteq \mathcal{M}(S_2)$*

To summarize the above property, we usually say that $\mathcal{M}$ is *monotone* with respect to $\subseteq$. This proposition leads to another immediately relevant theorem:

Corollary 1. *If $n_1 < n_2$ then $\mathcal{M}^{n_1}(\emptyset) \subseteq \mathcal{M}^{n_2}(\emptyset)$*

Proof Sketch. This is a consequence of $\emptyset \subseteq \mathcal{M}(\emptyset)$ and monotonicity of $\mathcal{M}$. □

So here's the thing: if $\mathcal{M}^n(\emptyset)$ for every *bigger* n contains all of the elements of $\mathcal{M}^m(\emptyset)$ for any *smaller* m, then *why the heck are we taking the union of all of them?!?*...that's to say, why don't we just take the biggest one and be done with it? Well the answer is because *there is no biggest natural number!*. So instead, we play this reindeer game where we collect the set of all sets $M^n(\emptyset)$, because the rules of the game say we can, and then we take their union, also because the rules of the game say we can, and that gives us what you might call the *limit* of the sequence of expanding sets $M^n(\emptyset)$. Look, I don't make the rules, so don't shoot the messenger!

Okay: to recap we now we have *two* different but equivalent ways of describing the set inductively defined by the rules $\mathcal{R}_{\Downarrow}$:

1. The set of all judgments justified by derivations $\textsc{Deriv}[\mathcal{R}_{\Downarrow}]$
2. The union of sets $\mathcal{M}^n[\mathcal{R}_{\Downarrow}](\emptyset)$ for all $n \in \mathbb{N}$.

Now it's time for a *third* equivalent definition. Take heart though, this next definition will also use $\mathcal{M}[\mathcal{R}_{\Downarrow}]$, but it's not as obvious, at least to me, that this one is equivalent, but it will be especially useful for rigorously justifying our principles of proof by induction.

We start with a definition. A set $S \subseteq \textsc{Term} \times \textsc{Value}$ is called *$\mathcal{M}$-forward-closed* if $\mathcal{M}(S) \subseteq S$. The justification for the name "closed" is an analogy with a subfield of mathematics called topology, about which we don't presently give a hoot, so don't worry about the name, just the property. Let's examine it in turn. Basically if a set S is $\mathcal{M}$-closed, then any attempt to justify new judgments by applying inductive rules to elements of S will just get you judgments that you already had: elements of S! If you think of "closed" as "self-contained", then the name makes a bit of sense. The "forward" part, introduced, I think, by the computer scientist Davide Sangiorgi, will become relevant when we get to coinduction.

Ok, so if we think back to our original definition of $\Downarrow$, then we would expect that from judgments of $\Downarrow$ we cannot use inductive rules to introduce any new judgments, because we already picked based on *all* derivations. We don't expect to be able to derive anything more. So intuitively, we can guess, rightfully, that $\Downarrow$ is $\mathcal{M}$-forward-closed. However, other sets are $\mathcal{M}$-forward-closed too. For example, the entire set $\textsc{Term} \times \textsc{Value}$ is vacuously $\mathcal{M}$-forward-closed because it necessarily contains any subset of itself (including itself). And there are probably others. But $\Downarrow$ is special: it's the "smallest" $\mathcal{M}$-closed set!

Proposition 42. $\Downarrow = \bigcap \{\, S \in \mathcal{P}(\textit{Term} \times \textit{Value}) \mid \mathcal{M}(S) \subseteq S \,\}$.

This is our *third* inductive definition of $\Downarrow$ in terms of the rule-set $\mathcal{R}_{\Downarrow}$. I'm not going to prove it here, but the rough intuition comes from our second definition of $\Downarrow$. We know that $\emptyset \subsetneq \mathcal{M}(\emptyset)$, and as we iterate, it keeps getting bigger and bigger and bigger, and we only stop where we get something such that $\mathcal{M}(\Downarrow) \subseteq \Downarrow$ which is a real turnaround! So, to put things loosely, how could there be any smaller $\mathcal{M}$-closed set? In fact, since $\mathcal{M}(\Downarrow) \subseteq \Downarrow$ and $\Downarrow \subseteq \mathcal{M}(\Downarrow)$ we immediately deduce that $\mathcal{M}(\Downarrow) = \Downarrow$.

Often people like to focus on the fact that an inductively-defined set is the least *fixed point* of $\mathcal{M}$, i.e. the least set S such that $\mathcal{M}(S) = S$, but it's actually more important that it is the least $\mathcal{M}$-forward-closed set. In fact, that property is fundamental to our principles of induction!

Proposition 43 (Principle of Monotone Induction for $\Downarrow$)**.** *Let $S \subseteq \textit{Term} \times \textit{Value}$. Then $\mathcal{M}(S) \subseteq S$ implies $\Downarrow \subseteq S$.*

10.2.2 Recovering Rule-based Induction Principles

Huh: what does Prop 43, a tiny and somehwat opaque proposition, have to do with the induction principles we have been using up to now? It's not immediately obvious, but given this weird property of $\mathcal{M}$, set containment, and sets, we can prove our earlier principles correct!

To demonstrate, let's *calculate* the principle of rule induction for $t \Downarrow v$ from the principle of monotone induction. Suppose that Φ is a property of term-value pairs $\langle t, v\rangle$. Then we can construct the set $S[\Phi]$ of pairs that satisfy P as follows.

$$S[\Phi] = \{\, \langle t, v\rangle \in \textsc{Term} \times \textsc{Value} \mid \Phi(t, v) \,\}$$

Well then, we would know that $\Phi(t, v)$ holds for all $\langle t, v\rangle \in \Downarrow$ if and only if $\Downarrow \subseteq S[\Phi]$. By our principle of monotone induction, it suffices to show that $\mathcal{M}(S[\Phi]) \subseteq S[\Phi]$, which expands to $\forall\, \langle t, v\rangle \in \textsc{Term} \times \textsc{Value}.\ \langle t, v\rangle \in \mathcal{M}(S[\Phi]) \Rightarrow \langle t, v\rangle \in S[\Phi]$, which in turn expands to $\forall\, \langle t, v\rangle \in \textsc{Term} \times \textsc{Value}.\ \langle t, v\rangle \in \mathcal{M}(S[\Phi]) \Rightarrow \Phi(t, v)$ Now, we *could* use this principle directly to prove things, but doing so is kind of clunky. Our rule-based principles, on the other hand, give some guiding structure and scaffolding to our proofs. Plus they may be better suited to intuitive understanding. So let's recover those rule-based principles!

Rule-structured $\mathcal{M}$ Our next step to recovering our beloved rule-based induction principle is to decompose the definition of $\mathcal{M}$ into separate parts that expose the independent contribution of each inductive rule.

$$\mathcal{M}[\mathcal{R}_\Downarrow](S) = \left\{ \langle t, v\rangle \in \textsc{Term} \times \textsc{Value} \;\middle|\; \exists \frac{S'}{\langle t', v'\rangle} \in \mathcal{R}_\Downarrow.\, t = t' \wedge v = v' \wedge S' \subseteq S \right\}$$

$$= \left\{ \langle t, v\rangle \in \textsc{Term} \times \textsc{Value} \;\middle|\; \exists \frac{\emptyset}{\langle \mathsf{true}, \mathsf{true}\rangle} \in \mathcal{R}_\Downarrow.\, t = \mathsf{true} \wedge v = \mathsf{true} \wedge \emptyset \subseteq S \right\}$$

$$\cup \left\{ \langle t, v\rangle \in \textsc{Term} \times \textsc{Value} \;\middle|\; \exists \frac{\emptyset}{\langle \mathsf{false}, \mathsf{false}\rangle} \in \mathcal{R}_\Downarrow.\, t = \mathsf{false} \wedge v = \mathsf{false} \wedge \emptyset \subseteq S \right\}$$

$$\cup \left\{ \langle t, v\rangle \in \textsc{Term} \times \textsc{Value} \;\middle|\; \exists \frac{\{\, \langle t_1, \mathsf{true}\rangle, \langle t_2, v_2\rangle \,\}}{\langle \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3, v_2\rangle} \in \mathcal{R}_\Downarrow. \begin{array}{l} t = \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \wedge v = v_2 \wedge \\ \{\, \langle t_1, \mathsf{true}\rangle, \langle t_2, v_2\rangle \,\} \subseteq S \end{array} \right\}$$

$$\cup \left\{ \langle t, v\rangle \in \textsc{Term} \times \textsc{Value} \;\middle|\; \exists \frac{\{\, \langle t_1, \mathsf{false}\rangle, \langle t_3, v_3\rangle \,\}}{\langle \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3, v_3\rangle} \in \mathcal{R}_\Downarrow. \begin{array}{l} t = \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \wedge v = v_3 \wedge \\ \{\, \langle t_1, \mathsf{false}\rangle, \langle t_3, v_3\rangle \,\} \subseteq S \end{array} \right\}$$

Since the definition of $\mathcal{M}$ simply collects term-value pairs that can be justified by the input set S according to the rule instances, we can break that collection process up according to the rules as above. Here we have taken liberty to abbreviate the existential notation in our set comprehensions. Here is an example of unpacking it:

$$\exists \frac{\{\, \langle t_1, \mathsf{true}\rangle, \langle t_2, v_2\rangle \,\}}{\langle \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3, v_2\rangle} \in \mathcal{R}_\Downarrow.\, Q(t_1, t_2, t_3, v_2) \equiv$$

$$\exists t_1, t_2, t_3 \in \textsc{Term}, v_2 \in \textsc{Value}, R \in \mathcal{R}_\Downarrow.\, R = \frac{\{\, \langle t_1, \mathsf{true}\rangle, \langle t_2, v_2\rangle \,\}}{\langle \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3, v_2\rangle} \wedge Q(t_1, t_2, t_3, v_2).$$

The unpacking above is quite systematic, just to show that there's no magic involved. However, the resulting conditions are more complicated than they need to be, so let's simplify further. We'll exploit a straightforward property of set comprehensions (which is easy to prove based on the axiom schema of separation).

Proposition 44. $(\forall x \in X.P(x) \Leftrightarrow Q(x)) \Rightarrow \{\, x \in X \mid P(x) \,\} = \{\, x \in X \mid Q(x) \,\}$

In words, this means that we can always replace the condition of a set comprehension with any equivalent logical predicate and still end up with a definite description of the same set. We can exploit this to simplify each set comprehension above, in particular by proving that the condition on membership in $\mathcal{R}_{\Downarrow}$ is redundant, e.g.,

Proposition 45.

$$\forall S \in \mathcal{P}(\textsc{Term} \times \textsc{Value}), \langle t, v\rangle \in \textsc{Term} \times \textsc{Value}.$$
$$\left(\exists \frac{\{\langle t_1, \mathit{true}\rangle, \langle t_2, v_2\rangle\}}{\langle \mathit{if}\ t_1\ \mathit{then}\ t_2\ \mathit{else}\ t_3, v_2\rangle} \in \mathcal{R}_{\Downarrow}. \quad t = \mathit{if}\ t_1\ \mathit{then}\ t_2\ \mathit{else}\ t_3 \wedge v = v_2 \wedge \{\langle t_1, \mathit{true}\rangle, \langle t_2, v_2\rangle\} \subseteq S\right)$$
$$\Leftrightarrow \exists t_1, t_2, t_3 \in \textsc{Term}, v_2 \in \textsc{Value}. \quad t = \mathit{if}\ t_1\ \mathit{then}\ t_2\ \mathit{else}\ t_3 \wedge v = v_2 \wedge \langle t_1, \mathit{true}\rangle \in S \wedge \langle t_2, v_2\rangle \in S.$$

This proposition and analogous propositions for each condition let us further simplify our rule-structured definition of $\mathcal{M}$:

$$\begin{aligned}
\mathcal{M}[\mathcal{R}_{\Downarrow}](S) = {} & \{\langle t, v\rangle \in \textsc{Term} \times \textsc{Value} \mid t = \mathsf{true} \wedge v = \mathsf{true}\} \\
& \cup \{\langle t, v\rangle \in \textsc{Term} \times \textsc{Value} \mid t = \mathsf{false} \wedge v = \mathsf{false}\} \\
& \cup \left\{\langle t, v\rangle \in \textsc{Term} \times \textsc{Value} \;\middle|\; \begin{array}{l}\exists t_1, t_2, t_3 \in \textsc{Term}, v_2 \in \textsc{Value}. \\ \quad t = \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \wedge v = v_2 \wedge \langle t_1, \mathsf{true}\rangle \in S \wedge \langle t_2, v_2\rangle \in S.\end{array}\right\} \\
& \cup \left\{\langle t, v\rangle \in \textsc{Term} \times \textsc{Value} \;\middle|\; \begin{array}{l}\exists t_1, t_2, t_3 \in \textsc{Term}, v_3 \in \textsc{Value}. \\ \quad t = \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \wedge v = v_3 \wedge \langle t_1, \mathsf{false}\rangle \in S \wedge \langle t_3, v_3\rangle \in S.\end{array}\right\}
\end{aligned}$$

From $\mathcal{M}$ forward-closure to Rule-based Induction Finally, we can in several steps unpack our sufficient condition for $\Downarrow \subseteq S[\Phi]$, namely:

$$\forall \langle t, v\rangle \in \textsc{Term} \times \textsc{Value}.\ \langle t, v\rangle \in \mathcal{M}[\mathcal{R}_{\Downarrow}](S[\Phi]) \Rightarrow \Phi(t, v)$$

into a rule-based induction principle that we can recognize. This will take us a couple of steps. First, let's expand $\langle t, v\rangle \in \mathcal{M}[\mathcal{R}_{\Downarrow}](S[\Phi])$:

$$\begin{aligned}
\langle t, v\rangle \in \mathcal{M}[\mathcal{R}_{\Downarrow}](S[\Phi]) \Leftrightarrow {} & (t = \mathsf{true} \wedge v = \mathsf{true}) \vee \\
& (t = \mathsf{false} \wedge v = \mathsf{false}) \vee \\
& \left(\begin{array}{l}\exists t_1, t_2, t_3 \in \textsc{Term}, v_2 \in \textsc{Value}. \\ \quad t = \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \wedge v = v_2 \wedge \Phi(t_1, \mathsf{true}) \wedge \Phi(t_2, v_2)\end{array}\right) \vee \\
& \left(\begin{array}{l}\exists t_1, t_2, t_3 \in \textsc{Term}, v_3 \in \textsc{Value}. \\ \quad t = \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \wedge v = v_3 \wedge \Phi(t_1, \mathsf{false}) \wedge \Phi(t_3, v_3)\end{array}\right).
\end{aligned}$$

Second, we can expand and exploit the tautologies $(A \vee B) \Rightarrow C \Leftrightarrow (A \Rightarrow C) \wedge (B \Rightarrow C)$ and $(\forall x.A(x) \wedge B(x)) \Leftrightarrow ((\forall x.A(x)) \wedge (\forall x.B(x)))$ to distribute the expansion of our condition to form:

$$\begin{aligned}
& (\forall \langle t, v\rangle \in \textsc{Term} \times \textsc{Value}.\ \langle t, v\rangle \in \mathcal{M}[\mathcal{R}_{\Downarrow}](S[\Phi]) \Rightarrow \Phi(t, v)) \\
\Leftrightarrow {} & (\forall \langle t, v\rangle \in \textsc{Term} \times \textsc{Value}.\ (t = \mathsf{true} \wedge v = \mathsf{true}) \Rightarrow \Phi(t, v)) \wedge \\
& (\forall \langle t, v\rangle \in \textsc{Term} \times \textsc{Value}.\ (t = \mathsf{false} \wedge v = \mathsf{false}) \Rightarrow \Phi(t, v)) \wedge \\
& \left(\forall \langle t, v\rangle \in \textsc{Term} \times \textsc{Value}.\ \left(\begin{array}{l}\exists t_1, t_2, t_3 \in \textsc{Term}, v_2 \in \textsc{Value}. \\ \quad t = \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \wedge v = v_2 \wedge \Phi(t_1, \mathsf{true}) \wedge \Phi(t_2, v_2)\end{array}\right) \Rightarrow \Phi(t, v)\right) \wedge \\
& \left(\forall \langle t, v\rangle \in \textsc{Term} \times \textsc{Value}.\ \left(\begin{array}{l}\exists t_1, t_2, t_3 \in \textsc{Term}, v_3 \in \textsc{Value}. \\ \quad t = \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \wedge v = v_3 \wedge \Phi(t_1, \mathsf{false}) \wedge \Phi(t_3, v_3)\end{array}\right) \Rightarrow \Phi(t, v)\right).
\end{aligned}$$

Finally, we can simplify this conjunction of four propositions, exploiting the (generalization of the) tautology

$$(\forall z.(\exists x, y.\, z = f(x) \wedge Q(x,y)) \Rightarrow D(z)) \Leftrightarrow (\forall x, y. Q(x,y) \Rightarrow D(f(x)))$$

to get our final form:

$$\begin{aligned}
&\mathcal{M}[\mathcal{R}_{\Downarrow}](S[\Phi]) \subseteq S[\Phi] \\
\Leftrightarrow &\Phi(\mathsf{true}, \mathsf{true}) \wedge \\
&\Phi(\mathsf{false}, \mathsf{false}) \wedge \\
&(\forall t_1, t_2, t_3 \in \textsc{Term}.\, \forall v_2 \in \textsc{Value}. \\
&\quad \Phi(t_1, \mathsf{true}) \wedge \Phi(t_2, v_2) \Rightarrow \Phi(\mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3, v_2)) \wedge \\
&(\forall t_1, t_2, t_3 \in \textsc{Term}.\, \forall v_3 \in \textsc{Value}. \\
&\quad \Phi(t_1, \mathsf{false}) \wedge \Phi(t_3, v_3) \Rightarrow \Phi(\mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3, v_3)).
\end{aligned}$$

Phew! The latter part of this last bi-implication is *exactly* the requirements of the principle of rule induction for $t \Downarrow v$. In short, we just have to show for each rule instance (say, by covering one entire rule at a time) that the rule-instance preserves the property Φ from premises to conclusions (taking account of side-conditions where necessary to constrain the rule instances). That's basically a direct reading of our principles of induction! Notice how the instances of $\Phi(t,v)$ that we typically view as "induction hypotheses" arise in this proposition: they came from the fact that establishing $\mathcal{M}[\mathcal{R}_{\Downarrow}](S[\Phi]) \subseteq S[\Phi]$ involves *presuming* that $\langle t, v\rangle \in \mathcal{M}[\mathcal{R}_{\Downarrow}](S[\Phi])$ in the expansion of the definition of $\subseteq$. This becomes especially relevant when we later discover that proofs by coinduction *do not* have a corresponding notion of "coinduction hypothesis" that you get to exploit. Instead they have what you might call "coinduction obligations" that you must fulfill!

Note that we can also use this approach to motivate our principles of induction on derivations: let Φ be a property of derivations $\mathcal{D} :: \langle t, v\rangle$. Then we can construct the analogous set $S[\Phi]$ of pairs $\langle t, v\rangle$ that correspond to derivations which satisfy Φ as follows.

$$S[\Phi] = \{\, \langle t, v\rangle \in \textsc{Term} \times \textsc{Value} \mid \exists \mathcal{D} \in \textsc{Deriv}[\mathcal{R}].\mathcal{D} :: \langle t, v\rangle \wedge \Phi(\mathcal{D}) \,\}$$

In short, our third characterization of inductively defining $\Downarrow$ using $\mathcal{R}$ serves as the source of our principles of proof by induction. The structure of our reasoning principles *is* determined by the structure of our definitions.

To summarize, we have not one, not two, but *three* different definite descriptions of $\Downarrow$ in terms of inductive rules $\mathcal{R}$, all of which we count as "inductive definition":

1. the set of all judgments justified by derivations $\mathcal{D} \in \textsc{Deriv}[\mathcal{R}]$ (call this *proof-theoretic*, taking derivations as proofs). This one lets us use derivations as scaffolding for reasoning about inductive definitions;
2. the set of all judgments justified by iteratively applying $\mathcal{M}[\mathcal{R}]$ to $\emptyset$ until we reach a fixed point (this is called *Kleene iteration* after the same dude who brought you regular expressions). This one lets us see induction as the process of progressively building up derivations;
3. the least $\mathcal{M}[\mathcal{R}]$-forward-closed set (call this *$\mathcal{M}$-forward-closure-based definition*). This one most directly justifies our proof techniques;

Phew! I recommend that you play with these different definitions: grab your favourite neighborhood inductive definition, use it to produce a monotone $\mathcal{M}$ function, see what happens when you throw things at it, and see if you can derive the principle of induction from the closure-based definition of the set.

Shallow Reasoning Principles We saw above that from the principle of monotone induction we could deduce our usual principle of induction. It turns out that we can also deduce our most basic forward and backward reasoning principles. They derive from two properties of $\Downarrow$ that we previously established:

Proposition 46.

1. $\mathcal{M}[\mathcal{R}_{\Downarrow}](\Downarrow) \subseteq \Downarrow$;
2. $\Downarrow \subseteq \mathcal{M}[\mathcal{R}_{\Downarrow}](\Downarrow)$;

Together, these two properties ensure that $\Downarrow$ is a fixed point of $\mathcal{M}[\mathcal{R}_{\Downarrow}]$, but each separately can be unpacked: forward-closedness implies a set of forward reasoning principles:

$$\begin{aligned}
&\mathcal{M}[\mathcal{R}_{\Downarrow}](\Downarrow) \subseteq \Downarrow \Leftrightarrow \\
&\forall \langle t, v\rangle \in \textsc{Term} \times \textsc{Value}.\ \langle t, v\rangle \in \mathcal{M}[\mathcal{R}_{\Downarrow}](\Downarrow) \Rightarrow \langle t, v\rangle \in \Downarrow \Leftrightarrow \\
\forall \langle t, v\rangle \in \textsc{Term} \times \textsc{Value}.\ &(t = \mathsf{true} \wedge v = \mathsf{true}) \vee \\
&(t = \mathsf{false} \wedge v = \mathsf{false}) \vee \\
&\left(\begin{array}{l}\exists t_1, t_2, t_3 \in \textsc{Term}, v_2 \in \textsc{Value}. \\ \quad t = \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \wedge v = v_2 \wedge \langle t_1, \mathsf{true}\rangle \in \Downarrow \wedge \langle t_2, v_2\rangle \in \Downarrow\end{array}\right) \vee \\
&\left(\begin{array}{l}\exists t_1, t_2, t_3 \in \textsc{Term}, v_3 \in \textsc{Value}. \\ \quad t = \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \wedge v = v_3 \wedge \langle t_1, \mathsf{false}\rangle \in \Downarrow \wedge \langle t_3, v_3\rangle \in \Downarrow.\end{array}\right) \\
&\qquad \Rightarrow \langle t, v\rangle \in \Downarrow \Leftrightarrow \\
&(\langle \mathsf{true}, \mathsf{true}\rangle \in \Downarrow) \wedge \\
&(\langle \mathsf{false}, \mathsf{false}\rangle \in \Downarrow) \wedge \\
&\left(\begin{array}{l}\forall t_1, t_2, t_3 \in \textsc{Term}, v_2 \in \textsc{Value}. \\ \quad \langle t_1, \mathsf{true}\rangle \in \Downarrow \wedge \langle t_2, v_2\rangle \in \Downarrow \Rightarrow \langle \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3, v_2\rangle \in \Downarrow\end{array}\right) \wedge \\
&\left(\begin{array}{l}\forall t_1, t_2, t_3 \in \textsc{Term}, v_3 \in \textsc{Value}. \\ \quad \langle t_1, \mathsf{false}\rangle \in \Downarrow \wedge \langle t_3, v_3\rangle \in \Downarrow \Rightarrow \langle \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3, v_3\rangle \in \Downarrow\end{array}\right).
\end{aligned}$$

while backward-closedness implies a set of backward reasoning principles (the variant that does not make any distinctions). I encourage you to work out the details for practice.

10.2.3 A Simpler Example

To get more comfortable with this machinery, we now inductively define a *finite* set. This will give us something that we can wrap our small brains around without too much trouble.

Let $\mathcal{U} = \{ a, b, c, d, e, f, g \}$, and consider the following inductive definition.

$\boxed{\mathcal{I} \subseteq \mathcal{U}}$

$$\frac{}{a} \qquad \frac{a \quad b}{f} \qquad \frac{a}{b} \qquad \frac{c}{d} \qquad \frac{d}{c} \qquad \frac{g}{e}$$

I have not bothered with the "$\in \mathcal{I}$" syntactic sugar here for two reasons: first the set is contrived, but more importantly, we will eventually use the same set of rules to define a *different* subset of $\mathcal{U}$.

Okay, let's call the set of rules $\mathcal{R}_g$ so it has a name, and we can use the induced function $\mathcal{M}_g = \mathcal{M}[\mathcal{R}_g]$ to calculate the contents of $\mathcal{I}$.

$$\begin{aligned}
\mathcal{M}_g^0(\emptyset) &= \emptyset \\
\mathcal{M}_g^1(\emptyset) = \mathcal{M}_g(\emptyset) &= \{ a \} \\
\mathcal{M}_g^2(\emptyset) = \mathcal{M}_g(\{ a \}) &= \{ a, b \} \\
\mathcal{M}_g^3(\emptyset) = \mathcal{M}_g(\{ a, b \}) &= \{ a, b, f \} \\
\mathcal{M}_g^4(\emptyset) = \mathcal{M}_g(\{ a, b, f \}) &= \{ a, b, f \} \\
\mathcal{M}_g^5(\emptyset) = \mathcal{M}_g(\{ a, b, f \}) &= \{ a, b, f \} \\
&\vdots
\end{aligned}$$

Conveniently enough, iterating $\mathcal{M}_g^n(\emptyset)$ converges in 3 steps, and

$$\mathcal{I} = \bigcup_{n\in\mathbb{N}} M_g^n(\emptyset) = \{\, a, b, f \,\}$$

Yes we still take the union of all iterations, even if almost all (infinity) of them are the same!

Now let's consider the $\mathcal{M}_g$-forward-closed sets:[3]

$$\{\, S \in \mathcal{P}(\mathcal{U}) \mid \mathcal{M}_g(S) \subseteq S \,\} = \left\{ \begin{array}{l} \{\, a, b, f \,\}, \{\, a, b, e, f \,\}, \{\, a, b, c, d, f \,\}, \{\, a, b, e, f, g \,\}, \\ \{\, a, b, c, d, e, f \,\}, \{\, a, b, c, d, e, f, g \,\} \end{array} \right\}.$$

From here we can calculate that

$$\mathcal{I} = \bigcap \{\, S \in \mathcal{P}(\mathcal{U}) \mid \mathcal{M}_g(S) \subseteq S \,\} = \{\, a, b, f \,\}.$$

Well, that's comforting: both definitions describe the same set, as expected.

10.3 Coinduction by example

Let's take our previous simple example of an inductive definition $\mathcal{I}$ and twist it a bit. The result, we'll eventually find, is a *coinductively*-defined set.

One way to define $\mathcal{I}$ inductively is to iteratively apply $\mathcal{M}_g$ to *the empty set*, and then take the *union* of all the sets you get. That's cool, but what if instead we iteratively apply $\mathcal{M}_g$ to the *universe* $\mathcal{U}$? Let's see:

$$\begin{aligned} \mathcal{M}_g^0(\mathcal{U}) &= \mathcal{U} \\ \mathcal{M}_g^1(\mathcal{U}) &= \mathcal{M}_g(\mathcal{U}) = \{\, a, b, c, d, e, f \,\} \\ \mathcal{M}_g^2(\mathcal{U}) &= \mathcal{M}_g(\{\, a, b, c, d, e, f \,\}) = \{\, a, b, c, d, f \,\} \\ \mathcal{M}_g^3(\mathcal{U}) &= \mathcal{M}_g(\{\, a, b, c, d, f \,\}) = \{\, a, b, c, d, f \,\} \\ \mathcal{M}_g^4(\mathcal{U}) &= \mathcal{M}_g(\{\, a, b, c, d, f \,\}) = \{\, a, b, c, d, f \,\} \\ &\vdots \end{aligned}$$

This process looks a lot like our inductive definition, but it's a bit *twisted around*. We start with the biggest set that we can, and for any universe $\mathcal{U}$ of judgments, we have $n_1 < n_2 \Rightarrow \mathcal{M}^{n_2}(\mathcal{U}) \subseteq \mathcal{M}^{n_1}(\mathcal{U})$. Since iterating gets progressively smaller, we can define the "limit" of this process as:

$$\mathcal{C} = \bigcap_{n\in\mathbb{N}} \mathcal{M}_g^n(\mathcal{U}).$$

Funky set we've got there! Continuing with our efforts, let's see what happens if we flip around the idea of $\mathcal{M}_g$-forward-closedness. Call a set $S \subseteq \mathcal{U}$ *$\mathcal{M}_g$-backward-closed* if $S \subseteq \mathcal{M}_g(S)$. Whereas $\mathcal{M}_g$-closed meant that applying rules to the elements of a set can only justify existing elements of the set, $\mathcal{M}_g$-backward-closed roughly means that every element of S is *immediately justified* by *other* elements of S (possibly including that very element itself).

Okay, let's look at the $\mathcal{M}_g$-backward-closed sets:[4]

$$\{\, S \in \mathcal{P}(\mathcal{U}) \mid S \subseteq \mathcal{M}_g(S) \,\} = \left\{ \begin{array}{l} \emptyset, \{\, a \,\}, \{\, a, b \,\}, \{\, d, c \,\}, \{\, a, c, d \,\}, \\ \{\, a, b, f \,\}, \{\, a, b, c, d \,\}, \{\, a, b, c, d, f \,\} \end{array} \right\}.$$

The $\mathcal{M}_g$-backward-closed sets are quite different from the $\mathcal{M}_g$-forward-closed sets! I think one of my favourites among them is $\{\, d, c \,\}$, especially if you look at the rules that cause them to immediately justify one another:

$$\frac{c}{d} \qquad\qquad \frac{d}{c}$$

[3] Yes I am lazy and wrote a program to compute this!

[4] More programming: program, program, program!

Hmm, if I were to try and build derivation trees out of these rules, I'd end up in two infinite regresses:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{d}}{c}}{d}}{c}}{d} \qquad\qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{c}}{d}}{c}}{d}}{c}$$

It's like we're going in *circles*! Hold that thought!

Earlier we defined $\mathcal{I}$ by taking the *intersection* of the $\mathcal{M}_g$-forward-closed sets. Now let's keep being twisty by taking the *union* of the $\mathcal{M}_g$-backward-closed sets which, lo and behold, gives us $\mathcal{C}$ again!

$$\mathcal{C} = \bigcup \{ S \in \mathcal{P}(\mathcal{U}) \mid S \subseteq \mathcal{M}_g(S) \} = \{ a, b, c, d, f \}$$

We can confirm, and this is no coincidence, that $\mathcal{C}$ is the *greatest* $\mathcal{M}_g$-backward-closed function.

Notice that $\mathcal{M}_g(\mathcal{C}) = \mathcal{C}$: we can argue that this is the greatest fixed point of $\mathcal{M}_g$, in contrast to $\mathcal{I}$ being its least fixed point. To summarize, $\mathcal{I} \subseteq \mathcal{U}$ is the set *inductively* defined by the rules $\mathcal{R}_g$ and $\mathcal{C} \subseteq \mathcal{U}$ is the set *coinductively* defined by $\mathcal{R}_g$. So with every set of rules you get not one, but *two* set definitions. Whether you want the second one at all is a different story, but you can have it if you so choose.

10.3.1 Proof by Coinduction

Earlier, we motivated our principles of proof by induction by seeing how they arise from characterizing an inductively-defined set as the least $\mathcal{M}[\mathcal{R}]$-forward-closed set. Analogously, we get a proof principle for coinduction. Let's state the specific one for our set $\mathcal{C}$.

Proposition 47 (Principle of Monotone Coinduction for $\mathcal{C}$)**.** *Let $S \subseteq \mathcal{U}$. Then if $S \subseteq \mathcal{M}_g(S)$ then $S \subseteq \mathcal{C}$.*

It's worth taking some time to compare this statement to Prop. 43. That proposition gives sufficient conditions for proving that an inductively-defined set is a subset of some other set, which we often define to represent some property of interest, especially when we want to prove that every element of an inductively-defined set has the property. However, bear in mind that our backward-reasoning lemmas are degenerate instances of induction, and they only prove a property for particular elements of the inductively-defined set. Such is the versatility of proof by induction.

The principle of coinduction at first glance looks weird. It shows how to prove that the coinductively-defined set is a *superset* of some set of interest. So what's the use of that? Well, this setup is particularly useful when a *property* is defined coinductively. For instance, we'll see that for a language like IMP, we can define divergence (i.e., nontermination) coinductively and then use the principle of coinduction to prove that some subset of command configurations all diverge. Make no mistake, sometimes coinduction is also useful for defining sets of objects, especially "infinite" or "cyclic" objects of interest. For now we won't go there since it's not our primary concern at the moment. But in general both induction and coinduction are useful for defining sets of objects and properties-as-sets. Which approach is better (if either) will depend on the problem at hand.

As with our earlier example of $\Downarrow$, let's use our inductive rules to transform Prop. 47 into a form that exposes the logical structure of the rules used to define $\mathcal{C}$. First, unpack the definition of $\mathcal{M}$, expanding subset conditions into elementhood ones on the fly:

$$\begin{aligned}\mathcal{M}(S) = &\{ u \in \mathcal{U} \mid u = a \} \cup \\ &\{ u \in \mathcal{U} \mid u = f \wedge a \in S \wedge b \in S \} \cup \\ &\{ u \in \mathcal{U} \mid u = b \wedge a \in S \} \cup \\ &\{ u \in \mathcal{U} \mid u = d \wedge c \in S \} \cup \\ &\{ u \in \mathcal{U} \mid u = c \wedge d \in S \} \cup \\ &\{ u \in \mathcal{U} \mid u = e \wedge g \in S \}.\end{aligned}$$

Now, use this to elaborate $S \subseteq \mathcal{M}(S)$[5]

$$\begin{aligned} S \subseteq \mathcal{M}(S) \equiv (\forall s \in S.\, s \in \mathcal{M}(S)) \Leftrightarrow \\ \forall s \in S.\, s = a \vee \\ (s = f \wedge a \in S \wedge b \in S) \vee \\ (s = b \wedge a \in S) \vee \\ (s = d \wedge c \in S) \vee \\ (s = c \wedge d \in S) \vee \\ (s = e \wedge g \in S). \end{aligned}$$

This suffices to give us:

Proposition 48 (Principle of Rule Coinduction for $\mathcal{C}$)**.** *Let $S \subseteq \mathcal{U}$. Then $S \subseteq \mathcal{C}$ if for every $s \in S$ one of the following holds:*

1. $s = a$;
2. $s = f$ *and* $\{ a, b \} \subseteq S$;
3. $s = b$ *and* $a \in S$;
4. $s = d$ *and* $c \in S$;
5. $s = c$ *and* $d \in S$;
6. $s = e$ *and* $g \in S$.

The structure of this principle of coinduction is quite different than the principle of induction, which we state here for the corresponding inductively-defined set, even if we ignore the set-versus-property difference.

Proposition 49 (Principle of Rule Induction for $x \in \mathcal{I}$)**.** *Let Φ be some property of elements $u \in \mathcal{U}$. Then $\Phi(x)$ holds for all $x \in \mathcal{I}$ if:*

1. $\Phi(a)$;
2. $\Phi(a)$ *and* $\Phi(b)$ *imply* $\Phi(f)$.
3. $\Phi(a)$ *implies* $\Phi(b)$.
4. $\Phi(c)$ *implies* $\Phi(d)$.
5. $\Phi(d)$ *implies* $\Phi(c)$.
6. $\Phi(g)$ *implies* $\Phi(e)$.

Okay comparison time.

1. Notice that in the principle of induction, the universal quantifier is over the inductively defined set $\mathcal{I}$, and appears in the conclusion of the proposition. In contrast, in the principle of coinduction, the universal quantifier is over elements of the chosen set S, and appears in the premise.
2. Notice that the components of the principle of induction are conjunctions, so can be proven completely independently as lemmas. In contrast, the principle of rule coinduction universally quantifies over set elements S (which can each be handled independently, at least in principle), but the clauses are disjunctions, so must be dealt with as an ensemble for each set S element (we must show that one of the cases holds).
3. Notice how each clause of induction is an implication (possibly vacuous) while each clause of coinduction is a conjunction.

[5]Remember that if $s \in \mathcal{U}$ then $s \in \{ u \in \mathcal{U} \mid \Phi(u) \} \Leftrightarrow \Phi(s)$.

All of these differences make it hard to relate the two principles to one another in a straightforward way. The coinductive clauses do not induce "coinduction hypotheses" the way that the induction clause implications do. Instead you have a set S of what I might call "co-theses", a set of peer elements, which simultaneously support the proof of their containment in the coinductive set. Furthermore, each clause of the principle of coinduction comes with (zero or more) "coinduction obligations:" elements that must *also* be in the set S in order to justify the presence of the particular element under scrutiny.

Despite these wild differences, the principles of monotone induction and monotone coinduction expose a close relationship between these two reasoning principles. Such is the mystery of proof by coinduction.[6]

Before we move on, let's prove a (fantastically obviously true) theorem using coinduction. The point here is not to learn some profound fact about $\mathcal{C}$, but rather to better understand the structure of a proof by coinduction.

Proposition 50. $c \in \mathcal{C}$.

Proof. By coinduction on $\mathcal{C}$. Let $S = \{ c, d \}$, and consider each element $s \in S$ in turn.

Case 33 ($s = c$). Then $d \in S$, so this case holds.

Case 34 ($s = d$). Then $c \in S$, so this case holds.

Thus, $S \subseteq \mathcal{C}$ which implies $c \in \mathcal{C}$. □

Notice how this proof went through. First, it wasn't enough to just work with $S = \{ c \}$: we also *needed* to consider d so as to meet c's coinduction obligation. Luckily c in turn meets d's coinduction obligation, so the two support each other (think back to the "cycling" infinite regress derivations from earlier). In general, a proof by coinduction involves examining each element of your set of "co-theses" S, possibly grouped by some aggregate structure, and then proving that for each case, the coinduction obligations are met.

As a side-note, you could imagine taking a *lazy* approach to figuring out what S to use: start with $S = \{ c \}$, and keep adding obligations to it and checking *them* on an as-needed basis. In this particular example, that would work out fine, and has a feel of "cacheing" obligations. Sometimes, however, you truly have an infinite set of elements S, and you need to recognize that a priori in order to complete the proof. These observations have implications for automated theorem proving!

Finally, note that since $\mathcal{C}$ is a fixed point of $\mathcal{M}_g$, just like $\mathcal{I}$, that both sets share related properties:

1. $\mathcal{M}_g(\mathcal{I}) \subseteq \mathcal{I}$ and $\mathcal{I} \subseteq \mathcal{M}_g(\mathcal{I})$.
2. $\mathcal{M}_g(\mathcal{C}) \subseteq \mathcal{C}$ and $\mathcal{C} \subseteq \mathcal{M}_g(\mathcal{C})$;

As a direct result, both sets can be shown to satisfy the *same* shallow forward and backward reasoning properties! So it's the deep properties, the principles of induction and coinduction respectively, that differentiate the reasoning tools we have at our disposal for each kind of definition.

10.3.2 Coinductive Definition With Derivations

So far we have reviewed our original approach to inductive definition, using derivations, then introduced two other approaches, and exploited both of the latter to provide two notions of coinductive definition. In both cases, switching from induction to coinduction involved "twisting" things, replacing subsets with supersets, or intersection with unions, or iteration starting from nothing to iteration starting from everything. This notion of twisting is often called *duality*, much like intersection is seen as dual to union in set theory, and conjunction is seen as dual to disjunction in logic. My take on coinduction is that it's just a twisted form of induction, but because we are not used to defining things this way it is a bit foreign.[7]

Now we will come full circle and talk about how coinductive definitions can be justified by derivations. First I'll give the punch-line, and then I'll try to explain: an element of a coinductively-defined set is any object that can be can be justified by a derivation of finite *or infinite* height! If you think about it, this sort of jives with our attempt to prove (inductively) that an infinite looping program big-steps, only to find that

[6] More broadly, this points to the mystery of *duality*.

[7] The great mathematician and early computer pioneer John Von Neumann is quoted to have once said "In mathematics you don't understand things. You get used to them." I try not to be quite so pessimistic, but…

we keep going up and up and up and never come back down. Now that phenomenon will be accepted as a derivation for a coinductive definition.

To me the iterative notions of inductive and coinductive definition are most helpful here. Let's start with inductive. As described earlier, we can conceive of the iteration $\mathcal{M}^n(\emptyset)$ as iteratively building finite-height derivations of height n, starting at the axioms and working their way down. In principle there can be lots of finite-height derivations of different height for the same judgment. In our simple example that doesn't happen.

Looking at things from the other direction, we can conceive of the iterations $\mathcal{M}^n(\mathcal{U})$ as an attempt to build both complete *and incomplete* derivations of judgments *from the bottom up*. Whenever a judgment *disappears* at some iteration, we have "proven" that no derivation of it is possible: it's been "voted off the island". Sometimes, on the other hand, as with say $a \in \mathcal{U}$, we can build a closed proof bottom-up in one step, and that's the only derivation that we can build. At every iteration $n > 1$, we essentially keep producing the same height-1 derivation. On the other hand, in our interesting case $\{ d, c \}$, iterating on *that* with $\mathcal{M}_g$ keeps building up towers of partial proofs that never lead to a *refutation* of membership, nor do they close off at the top. "In the limit" we end up with an infinite derivation for c and another for d. Note that in this case we can *represent* these two derivations as a *system of equations* on derivations:

$$\mathcal{D}_1 = \dfrac{\dfrac{\mathcal{D}_2}{d}}{c}, \qquad \mathcal{D}_2 = \dfrac{\dfrac{\mathcal{D}_1}{c}}{d}$$

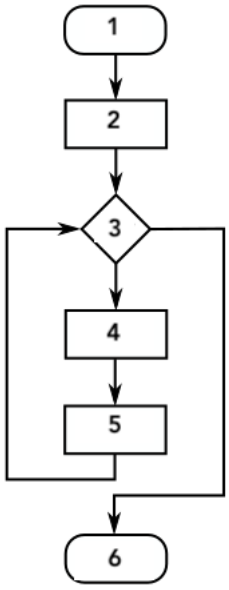
The system of equations itself is "cyclic", but it is meant to be interpreted as constraints on the structure of a possibly infinite derivation tree. You may see stated in the literature that coinductive derivations are cyclic. Not so much. They can *literally* be infinite, and infinitely changing. But they can also have a *regular* structure that is finitely represented (e.g., by a system of equations). When that finite representation looks cyclic, it represents something that is infinite but with a discernable pattern. This is exactly analogous to how regular expressions can describe infinite repetitive (not cyclic) strings. It's also analogous to how a set of recursive equations can definitely describe a function.

With this, we can define the set of *finite and infinite* derivation trees $\mathcal{D} \in \textsc{Deriv}^\star[\mathcal{R}]$ and say that

$$\mathcal{C} = \{ x \in \mathcal{U} \mid \exists \mathcal{D} \in \textsc{Deriv}^\star[\mathcal{R}]. \mathcal{D} :: x \}$$

10.4 More Control Flow Analysis: Necessary Reachability Revisited

Previously we introduced a control-flow graph for a program:



Which we represented using sets:

$$n \in \textsc{Node} := \{\, 1, 2, 3, 4, 5, 6 \,\}$$
$$e \in \textsc{Edge} := \{\, \langle 1, 2\rangle, \langle 2, 3\rangle, \langle 3, 4\rangle, \langle 3, 6\rangle, \langle 4, 5\rangle, \langle 5, 3\rangle \,\}$$

And then defined a "necessary reachability" relation among the program's nodes:

$$(\bullet\ [\mathrm{R}]\ \bullet) \subseteq \textsc{Node} \times \textsc{Node}$$

which represents the idea that if execution ever arrives at one node, then execution is sure to eventually reach some other node.

However, our definition was circuitous. We first defined "possible non-reachability" inductively and then took its complement. This definition is effective, but does not immediately provide any interesting reasoning principles about necessary reachability. But now that we are equipped with coinductive definition, we *can* give a direct definition.

$\boxed{(\bullet\ [\mathrm{R}]\ \bullet) \subseteq \textsc{Node} \times \textsc{Node}}$ **Necessary Reachability**

$$\frac{\{\, \langle n_2, n_3\rangle \textbf{ for } n_2 \in \{\, n \in \textsc{Node} \mid \langle n_1, n\rangle \in \textsc{Edge} \wedge n \neq n_3 \,\} \,\}}{\langle n_1, n_3\rangle} \quad \exists n \in \textsc{Node}.\ \langle n_1, n\rangle \in \textsc{Edge}$$

This rule is rather involved, so it is worth considering in pieces. First, the side condition ensures that n_1 has *at least* one outgoing edge: a terminal node does not *necessarily* reach any other node because ...well because it does not reach any other node under any circumstance! Second, the set expression in the premise says that the premises are all pairs of the form $\langle n_2, n_3\rangle$ for which n_1, the source node in the conclusion, immediately reaches n_2, except for n_3. Intuitively this means that n_1 must reach n_3 if every edge from n_1 either reaches n_3 immediately or goes to a node n_2 that in turn must reach n_3, and there is surely an edge. Since $\textsc{Node} \times \textsc{Node}$ is finite, we can compute $(\bullet\ [\mathrm{R}]\ \bullet)$ by using the rule (or its corresponding monotone function) to filter out the pairs that are inconsistent with this constraint.

[RG: Explain how to derive this definition from possible non-reachability]

Bibliography

P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, chapter C.7, pages 739–782. North-Holland, 1977.

J. Avigad. Reliability of mathematical inference. *Synthese*, 2020. doi: 10.1007/s11229-019-02524-y. `https://doi.org/10.1007/s11229-019-02524-y`.

F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, New York, NY, USA, 1998. ISBN 0-521-45520-0.

J. Bagaria. Set theory. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. The Metaphysics Research Lab, winter 2014 edition, 2014. URL `http://plato.stanford.edu/archives/win2014/entries/set-theory/`.

R. Baldoni, E. Coppa, D. C. D'elia, C. Demetrescu, and I. Finocchi. A survey of symbolic execution techniques. *ACM Comput. Surv.*, 51(3):50:1–50:39, May 2018. ISSN 0360-0300. doi: 10.1145/3182657. URL `http://doi.acm.org/10.1145/3182657`.

H. P. Barendregt. *The lambda calculus - its syntax and semantics*, volume 103 of *Studies in logic and the foundations of mathematics*. North-Holland, 1985. ISBN 978-0-444-86748-3.

A. Bauer. Proof of negation and proof by contradiction. Mathematics and Computation Blog, March 2010. `http://math.andrej.com/2010/03/29/proof-of-negation-and-proof-by-contradiction/`.

L. Crosilla. Set Theory: Constructive and Intuitionistic ZF. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2020 edition, 2020.

M. Felleisen and D. P. Friedman. A syntactic theory of sequential state. *Theoretical Computer Science*, 69(3):243–287, 1989. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(89)90069-8. URL `http://www.sciencedirect.com/science/article/pii/0304397589900698`.

M. Felleisen, R. B. Findler, and M. Flatt. *Semantics Engineering with PLT Redex*. MIT Press, 1st edition, 2009.

J. Ferreirós. The early development of set theory. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2019 edition, 2019. URL `https://plato.stanford.edu/archives/sum2019/entries/settheory-early/`.

D. Grossman, G. Morrisett, T. Jim, M. Hicks, Y. Wang, and J. Cheney. Region-based memory management in cyclone. In *Proceedings of the ACM SIGPLAN 2002 Conference on Programming Language Design and Implementation*, PLDI '02, pages 282–293, New York, NY, USA, 2002. ACM. ISBN 1-58113-463-0. doi: 10.1145/512529.512563. URL `http://doi.acm.org/10.1145/512529.512563`.

P. R. Halmos. *Naive Set Theory*. Springer-Verlag, first edition, Jan. 1960. ISBN 0387900926.
A classic introductory textbook on set theory.

P. R. Harper. *Practical Foundations for Programming Languages*. Cambridge University Press, New York, NY, USA, 2012. URL `http://www.cs.cmu.edu/%7Erwh/plbook/1sted-revised.pdf`.

D. J. Howe. Proving congruence of bisimulation in functional programming languages. *Inf. Comput.*, 124 (2):103–112, Feb. 1996. ISSN 0890-5401.

G. Kahn. Natural semantics. In *4th Annual Symposium on Theoretical Aspects of Computer Sciences on STACS 87*, pages 22–39, London, UK, UK, 1987. Springer-Verlag.

R. Krebbers. *The C standard formalized in Coq*. PhD thesis, Radboud University Nijmegen, December 2015. URL `https://robbertkrebbers.nl/thesis.html`.

C. Kuratowski. Sur la notion de l'ordre dans la théorie des ensembles. *Fundamenta Mathematicae*, 2(1): 161–171, 1921. doi: 10.4064/fm-2-1-161-171. `https://web.archive.org/web/20190429103938/http://matwbn.icm.edu.pl/ksiazki/fm/fm2/fm2122.pdf`.

P. J. Landin. The mechanical evaluation of expressions. *Comput. J.*, 6(4):308–320, 1964. doi: 10.1093/comjnl/6.4.308. URL `https://doi.org/10.1093/comjnl/6.4.308`.

X. Leroy and H. Grall. Coinductive big-step operational semantics. *Inf. Comput.*, 207(2):284–304, Feb. 2009. ISSN 0890-5401.

P. Maddy. Believing the axioms. I. *The Journal of Symbolic Logic*, 53(02):481–511, 1988.
An interesting (though complicated) analysis of why set theorists believe in their axioms.

J. McCarthy. Recursive functions of symbolic expressions and their computation by machine, part I. *Commun. ACM*, 3(4):184–195, 1960.

J. H. J. Morris. *Lambda-Calculus Models of Programming Languages*. PhD thesis, Massachusetts Institute of Technology, Feb. 1969. URL `http://hdl.handle.net/1721.1/64850`.

A. M. Pitts. *Nominal sets: Names and symmetry in computer science*. Cambridge University Press, 2013.

G. D. Plotkin. The origins of structural operational semantics. *J. Log. Algebr. Program.*, 60-61:3–15, 2004a. doi: 10.1016/j.jlap.2004.03.009. URL `http://dx.doi.org/10.1016/j.jlap.2004.03.009`.

G. D. Plotkin. A structural approach to operational semantics. *J. Log. Algebr. Program.*, 60-61:17–139, 2004b.

B. Popik. "pull yourself up by your bootstraps". Weblog entry, September 2012. `https://www.barrypopik.com/index.php/new_york_city/entry/pull_yourself_up_by_your_bootstraps/`.

E. Reck and G. Schiemer. Structuralism in the Philosophy of Mathematics. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Spring 2020 edition, 2020.

D. Sangiorgi. On the origins of bisimulation and coinduction. *ACM Trans. Program. Lang. Syst.*, 31(4): 15:1–15:41, May 2009. ISSN 0164-0925.

W. Sieg and D. Schlimm. Dedekind's analysis of number: Systems and axioms. *Synthese*, 147(1):121–170, Oct 2005.

J. Spolsky. The law of leaky abstractions. Joel on Software Blog, November 2002. `https://www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/`.

G. L. Steele. Debunking the "expensive procedure call"" myth or, procedure call implementations considered harmful or, lamdba: The ultimate goto. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1977. URL `http://dspace.mit.edu/handle/1721.1/5753`.

S. Stenlund. Descriptions in intuitionistic logic. In S. Kanger, editor, *Proceedings of the Third Scandinavian Logic Symposium*, volume 82 of *Studies in Logic and the Foundations of Mathematics*, pages 197 – 212. Elsevier, 1975. URL `http://www.sciencedirect.com/science/article/pii/S0049237X08707328`.

M. Tiles. Book Review: Stephen Pollard. Philosophical Introduction to Set Theory. *Notre Dame Journal of Formal Logic*, 32(1):161–166, 1990.
A brief introduction to the philosophical issues underlying set theory as a foundation for mathematics.

C. Urban and M. Norrish. A formal treatment of the Barendregt variable convention in rule inductions. In *Proceedings of the 3rd ACM SIGPLAN Workshop on Mechanized Reasoning about Languages with Variable Binding*, MERLIN '05, page 25–32, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595930728. doi: 10.1145/1088454.1088458. URL `https://doi.org/10.1145/1088454.1088458`.

C. Urban, S. Berghofer, and M. Norrish. Barendregt's variable convention in rule inductions. In *Proceedings of the 21st International Conference on Automated Deduction: Automated Deduction*, CADE-21, page 35–50, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 9783540735946. doi: 10.1007/978-3-540-73595-3_4. URL `https://doi.org/10.1007/978-3-540-73595-3_4`.

D. van Dalen. *Logic and structure (3. ed.)*. Universitext. Springer, 1994. ISBN 978-3-540-57839-0.

A. K. Wright and M. Felleisen. A syntactic approach to type soundness. *Inf. Comput.*, 115(1):38–94, Nov. 1994.

B. Zimmer. figurative "bootstraps". email to linguistlist mailing list, August 2005. `http://listserv.linguistlist.org/pipermail/ads-l/2005-August/052756.html`.