Coinductive Validity

Rob van Glabbeek
Data61, CSIRO, Sydney, Australia
School of Computer Science and Engineering, University of New South Wales, Sydney, Australia

This note formally defines the concept of coinductive validity of judgements, and contrasts it with inductive validity. For both notions it shows how a judgement is valid iff it has a formal proof. Finally, it defines and illustrates the notion of a proof by coinduction.

Induction and coinduction are techniques for defining sets, namely as least or greatest solutions of certain collections of inequations, and for proving membership of thusly defined sets. Here I deal with the special case of sets of *judgements*, and inequations in the shape of *proof rules*. A *judgement* can for instance be a formula in some logic, or a statement $P \models \chi$ saying that a system P satisfies a temporal formula χ. What we need to know about judgements is that they evaluate to *true* or *false*, and that (co)induction can be used to define the set of *valid* ones—those that evaluate to *true*.

I refer to [San11] for a general introduction to coinduction, to [JR97] for a coalgebraic treatment of coinduction, and to [KS17] for a practical introduction to the use of coinduction. The main contribution of the present paper is a concise definition of what coinductive validity is. Naturally, this is contrasted with inductive validity. I also define and illustrate the concept of a coinductive proof of validity.

Proof rules Given a set Φ of potential *judgements* ψ, the *valid* judgements are defined either inductively or coinductively by means of a set of *proof rules* of the form $\frac{\psi_1\ \psi_2\ \cdots\ \psi_n}{\psi}$. The guideline is:

A judgement ψ is valid if and only if there is a rule
with conclusion ψ of which all premises are valid. (1)

Proof A *formal proof* of a judgement φ is an upwardly branching tree of which the nodes are labelled by judgements, such that the root is labelled by φ, and whenever ψ is the label of a note s and K is the set of labels of the nodes directly above s, then $\frac{K}{\psi}$ is one of the given proof rules. Such a proof is *well-founded* if there is no infinite path that keeps going up.

Inductive validity Let $\mathcal{S}_{ind}$ be the collection of all notions of validity $\mathcal{V} \subseteq \Phi$ that satisfy the "if"-direction of (1). This collection is closed under intersections, and hence has a least element $\mathcal{V}_{ind} \subseteq \Phi$. By definition, this is the notion of a valid judgement that is inductively defined by the given rules.

Observation 1 A judgement is inductively valid iff it has a well-founded formal proof.

It follows that the inductively valid judgements satisfy all of (1), and not just the "if"-direction.

Coinductive validity Let $\mathcal{S}_{coind}$ be the collection of all notions of validity $\mathcal{V} \subseteq \Phi$ that satisfy the "only if"-direction of (1). This collection is closed under arbitrary unions, so has a largest element $\mathcal{V}_{coind} \subseteq \Phi$. By definition, this is the notion of a valid judgement that is coinductively defined by the given rules.

Observation 2 A judgement is coinductively valid iff it has a formal proof.

It follows that the coinductively valid judgements satisfy all of (1), and not just the "only if"-direction.

The key difference between inductive and coinductive validity is well expressed in [KS17]:

> *A property holds by induction if there is good reason for it to hold;*
> *whereas a property holds by coinduction if there is no good reason for it not to hold.*

Proofs by coinduction A method to show that a set $S \subseteq \Phi$ of judgements is coinductively valid, is by means of a *coinductive proof*: it is sufficient to construct, for each judgement $\varphi \in S$, a nonempty proof fragment that may use all elements of S as *coinduction hypotheses*, and derives φ from them.

Example 1 Consider a simple process algebra with expressions E given by the BNF-like grammar

$$E ::= \sum_{i\in I} a_i.E_i \mid \mu X.\sum_{i\in I} a_i.E_i \mid X$$

where I is a finite index set, actions a_i are drawn from an alphabet $\mathcal{A}$, and variables X from a given set *Var*. When $I = \{i_0\}$ is a singleton, the expression $\sum_{i\in I} a_i.E_i$ may be abbreviated as $a_{i_0}.E_{i_0}$.

This language can be seen as a fragment of either CCS [Mil90] or CSP [BHR84]. I didn't include a clause $\mu X.E$ in order to restrict to *guarded recursion* [Mil90]. A *process* or closed expression is one in which each variable X occurs within the scope of an expression $\mu X.E$. Now let $\equiv$ be the binary relation between processes coinductively defined by the following proof rules:

$$\frac{P_i \equiv Q_i \text{ for each } i \in I}{\sum_{i\in I} a_i.P_i \equiv \sum_{i\in I} a_i.Q_i}\;(\textsc{act}) \qquad \frac{E\left\{{}^{\mu X.E}\!/\!_{X}\right\} \equiv Q}{\mu X.E \equiv Q}\;(\textsc{rec-l}) \qquad \frac{P \equiv E\left\{{}^{\mu X.E}\!/\!_{X}\right\}}{P \equiv \mu X.E}\;(\textsc{rec-r}).$$

Formally, the first rule is a *schema*, with an instance for each choice of a finite index set I, actions a_i, and processes P_i and Q_i. Likewise, the other two rules are schemata with an instance for each variable X, expression E and process P or Q. Here $E\left\{{}^{\mu X.E}\!/\!_{X}\right\}$ denotes the result of substituting $\mu X.E$ for X in the expression E.

It is not hard to show that $\equiv$ coincides with the familiar notion of *strong bisimulation equivalence* [Mil90]. Here I merely give proof by coinduction of the statement $\mu X.a.a.X \equiv \mu Y.a.a.a.Y$.

With $S \subseteq \Phi$ a singleton, I merely need to give a nonempty proof fragment of this equation, allowing itself as coinduction hypothesis. It is shown on the right.

$$\begin{array}{cl}
\mu X.a.a.X \equiv \mu Y.a.a.a.Y & \\ \cline{1-1}
a.\mu X.a.a.X \equiv a.\mu Y.a.a.a.Y & (\textsc{act}) \\ \cline{1-1}
a.a.\mu X.a.a.X \equiv a.a.\mu Y.a.a.a.Y & (\textsc{act}) \\ \cline{1-1}
\mu X.a.a.X \equiv a.a.\mu Y.a.a.a.Y & (\textsc{rec-l}) \\ \cline{1-1}
a.\mu X.a.a.X \equiv a.a.a.\mu Y.a.a.a.Y & (\textsc{act}) \\ \cline{1-1}
a.\mu X.a.a.X \equiv \mu Y.a.a.a.Y & (\textsc{rec-r}) \\ \cline{1-1}
a.a.\mu X.a.a.X \equiv a.\mu Y.a.a.a.Y & (\textsc{act}) \\ \cline{1-1}
\mu X.a.a.X \equiv a.\mu Y.a.a.a.Y & (\textsc{rec-l}) \\ \cline{1-1}
a.\mu X.a.a.X \equiv a.a.\mu Y.a.a.a.Y & (\textsc{act}) \\ \cline{1-1}
a.a.\mu X.a.a.X \equiv a.a.a.\mu Y.a.a.a.Y & (\textsc{act}) \\ \cline{1-1}
a.a.\mu X.a.a.X \equiv \mu Y.a.a.a.Y & (\textsc{rec-r}) \\ \cline{1-1}
\mu X.a.a.X \equiv \mu Y.a.a.a.Y & (\textsc{rec-l})
\end{array}$$

References

[BHR84] S.D. Brookes, C.A.R. Hoare & A.W. Roscoe (1984): *A theory of communicating sequential processes*. *Journal of the ACM* 31(3), pp. 560–599, doi:10.1145/828.833.

[JR97] Bart Jacobs & Jan Rutten (1997): *A Tutorial of (co)Algebras and (Co)Induction*. *EATCS Bulletin* 62, pp. 1132–1152. Available at `https://www.cs.ru.nl/B.Jacobs/PAPERS/JR.pdf`.

[KS17] Dexter Kozen & Alexandra Silva (2017): *Practical coinduction*. *Math. Struct. Comput. Sci.* 27(7), pp. 1132–1152, doi:10.1017/S0960129515000493.

[Mil90] R. Milner (1990): *Operational and algebraic semantics of concurrent processes*. In J. van Leeuwen, editor: *Handbook of Theoretical Computer Science*, chapter 19, Elsevier Science Publishers B.V. (North-Holland), pp. 1201–1242. Alternatively see *Communication and Concurrency*, Prentice-Hall, Englewood Cliffs, 1989, of which an earlier version appeared as *A Calculus of Communicating Systems*, LNCS 92, Springer, 1980, doi:10.1007/3-540-10235-3.

[San11] Davide Sangiorgi (2011): *Introduction to Bisimulation and Coinduction*. Cambridge University Press, doi:10.1017/CBO9780511777110.