Flexible coinduction for infinite behaviour[⋆]

Francesco Dagnino

DIBRIS, University of Genova, Italy
`franceso.dagnino@dibris.unige.it`

Abstract. *Generalized inference systems* have been recently defined to overcome the strong dichotomy between inductive and coinductive interpretations. They support a *flexible form of coinduction*, subsuming even induction, which allows one to mediate between the two standard semantics. Recently, this framework has been successfully adopted to define semantic judgments which uniformly model finite and infinite computations. In this communication, we survey these results and outline directions for further developments.

Keywords: inference systems · coinduction · operational semantics · infinite behaviour

1 Introduction

In operational semantics, *finite* behaviour can be easily modelled by inductive techniques. That is, we can define a judgement $c \Rightarrow r$, meaning that the evaluation of the configuration c terminates with final result r, by an inductive[1] inference system, either on top of a small-step relation [17,18], or directly, as in big-step style [10].

However, modelling *infinite* behaviour is not that easy. The simplest infinite behaviour we may want to model is divergence itself, which can be formalized by a judgement $c \Rightarrow \infty$. Such judgement is usually defined at the meta-level in small-step semantics, saying that $c \Rightarrow \infty$ holds if there is "an infinite sequence of steps" starting from c. In big-step style the situation is worse: non-terminating and stuck computation cannot be distinguished.

In literature, several approaches to face this issue can be found: divergence is modelled by a separate coinductive definition [8,11], by interpreting coinductively standard rules [11,2,7], by using definitional interpreters [19] in combination with step indexing, as in [16], or with the partiality monad [6], as in [9].

However, modelling divergence is not enough to formally reason about non-terminating programs, since we want in some cases richer information about their behaviour. Hence we need more sophisticated semantics, which are even more difficult to define. An example is *trace semantics*, where, in addition to the final

[⋆] Special thanks go to Elena Zucca and Davide Ancona for their useful comments and the review of this communication.

[1] Valid judgements are those having a finite derivation.

result, a trace of observed events is produced. Also in this setting there is some literature [12,13,14] where coinduction is adopted to get a correct definition.

All these approaches are not completely satisfactory because they require ad-hoc changes to the natural definition. A possible alternative, shown in [4,5], is to use *generalized inference systems* [3], which are an extension of inference systems [1] allowing flexible coinductive definitions, thanks to a generalized interpretation.

In this communication we summarize the sresults obtained by this approach and we outline possible directions for further developments. More precisely, in Section 2 we briefly present the framework of generalized inference systems, and in Section 3 we deal with their application to infinite behaviour.

2 Flexible coinduction

In this section we briefly introduce *inference systems* [1], and our generalization based on *corules*, which is a smooth extension of what we have proposed in [3].

Assume a set $\mathcal{U}$ called the *universe*, whose elements are called *judgments*. An *inference system* $\mathcal{I}$ consists of a set of *(inference) rules*, which are pairs $\dfrac{Pr}{c}$, with $Pr \subseteq \mathcal{U}$ the set of *premises*, $c \in \mathcal{U}$ the *consequence* (a.k.a. *conclusion*). A rule with an empty set of premises is also called an *axiom*. A *proof tree* is a tree whose nodes are (labeled with) judgments, and there is a node c with set of children Pr only if there exists a rule $\dfrac{Pr}{c}$.

We can associate with an inference system $\mathcal{I}$ a function $F_{\mathcal{I}} : \wp(\mathcal{U}) \to \wp(\mathcal{U})$, monotone on the complete lattice $\wp(\mathcal{U})$ ordered by set inclusion. We define the *inductive interpretation* $[\![\mathcal{I}]\!]^{\mathsf{ind}}$ as the least fixed point of $F_{\mathcal{I}}$, and the *coinductive interpretation* $[\![\mathcal{I}]\!]^{\mathsf{coind}}$ as the greatest fixed point of $F_{\mathcal{I}}$, which exist thanks to Tarski's theorem [20]. An equivalent proof-theoretic characterization is based on proof trees: $[\![\mathcal{I}]\!]^{\mathsf{ind}}$ is the set of judgements which are the root of a well-founded proof tree, while $[\![\mathcal{I}]\!]^{\mathsf{coind}}$ of those which are the root of an arbitrary (well-founded or not) proof tree.

In some cases neither of these two approaches yields the expected semantics. Let us consider an example: let $\mathbb{N}^\infty$ be the set of finite and infinite lists of natural numbers. The predicate $\mathit{maxElem}(l, x)$, stating that $x \in \mathbb{N}$ is the maximum of $l \in \mathbb{N}^\infty$, is defined as follows:

$$\frac{}{\mathit{maxElem}(x, x)} \qquad \frac{\mathit{maxElem}(l, y)}{\mathit{maxElem}(xl, z)}\ z = \max\{x, y\}$$

The inductive interpretation is not enough, because the predicate would be undefined for all infinite lists. Indeed, in order to compute the maximum of an infinite list, we need to inspect all its elements, and this clearly cannot be done in a finitely many steps. On the other hand, the coinductive interpretation does not work as well: for the list L containing only 1s, we can derive by an infinite proof, applying infinitely many times the second rule, the judgement $\mathit{maxElem}(L, n)$

for all $n \geq 1$. That is, the coinductive interpretation is not sound, since it allows us to derive too many judgements. Hence, we need something in the middle between the (least) inductive interpretation and the (greatest) coinductive one.

In [3] we have proposed a generalization of inference systems to allow more flexible interpretations; here we summarize a slight further extension based on the notion of *corules*.

Definition 1 (Inference system with corules). *An* inference system with corules*, or* generalized inference system*, is a pair $\langle \mathcal{I}, \mathcal{I}^{\mathsf{co}} \rangle$ where $\mathcal{I}$ and $\mathcal{I}^{\mathsf{co}}$ are inference systems, whose elements are called* rules *and* corules*, respectively.*

Corules are written $\dfrac{Pr}{c}$ and a corule with empty premises is called *coaxiom*. Analogously to rules, the meaning of corules is to derive a consequence from the premises. However, they can only be used in a special way, as described below.

The interpretation of an inference system with corules $\langle \mathcal{I}, \mathcal{I}^{\mathsf{co}} \rangle$ is defined in two steps:

1. First, we consider the inference system $\mathcal{I} \cup \mathcal{I}^{\mathsf{co}}$ where corules can be used as rules as well, and we take its inductive interpretation $[\![\mathcal{I} \cup \mathcal{I}^{\mathsf{co}}]\!]^{\mathsf{ind}}$.
2. Then, we take the coinductive interpretation of the inference system obtained from $\mathcal{I}$ by keeping only rules with consequence in $[\![\mathcal{I} \cup \mathcal{I}^{\mathsf{co}}]\!]^{\mathsf{ind}}$.

Altogether, we get the following definition.

Definition 2 (Interpretation). *Let $\langle \mathcal{I}, \mathcal{I}^{\mathsf{co}} \rangle$ be a generalized inference system. Then, its* interpretation $[\![\mathcal{I}, \mathcal{I}^{\mathsf{co}}]\!]$ *is defined by* $[\![\mathcal{I}, \mathcal{I}^{\mathsf{co}}]\!] = [\![\mathcal{I}_{|[\![\mathcal{I} \cup \mathcal{I}^{\mathsf{co}}]\!]^{\mathsf{ind}}}]\!]^{\mathsf{coind}}$

where, for a subset $S \subseteq \mathcal{U}$, $\mathcal{I}_{|S}$ denotes the inference system obtained from $\mathcal{I}$ by keeping only rules with consequence in S.

It can be shown that $[\![\mathcal{I}, \mathcal{I}^{\mathsf{co}}]\!]$ is a fixed point of $F_{\mathcal{I}}$, in particular the greatest fixed point *below* the least fixed point of $F_{\mathcal{I} \cup \mathcal{I}^{\mathsf{co}}}$. From this fact, we also get a generalization of the coinduction principle to our setting, called the *bounded coinduction principle*, which is a proof technique to show the completeness of a definition.

An interesting fact is that both the inductive and the coinductive interpretations are subsumed by this generalized semantics. Indeed, if $\mathcal{I}^{\mathsf{co}} = \emptyset$, we get the former, while if $\mathcal{I}^{\mathsf{co}}$ contains a coaxiom for each $c \in \mathcal{U}$ we get the latter.

In proof-theoretic terms, $[\![\mathcal{I}, \mathcal{I}^{\mathsf{co}}]\!]$ is the set of judgments which have an arbitrary (well-founded or not) proof tree in $\mathcal{I}$, whose nodes all have a well-founded proof tree in $\mathcal{I} \cup \mathcal{I}^{\mathsf{co}}$.

Considering again the above example, by adding the coaxiom

$$\dfrac{}{\mathit{maxElem}(xl, x)}$$

we get the expected semantics. Intuitively, the coaxiom forces the result to belong to the list. Then, from the standard inference system we get that $\mathit{maxElem}(l, x)$ iff x is an upper bound of l, and from the coaxiom x must belong to l, hence x must be the maximum of l.

3 Operational semantics for infinite behaviour

We started studying how corules can be used to model infinite behaviour [4,5] by defining some example semantics including also infinite computations and showing how they support formal reasoning on diverging programs.

In [4] we began from the simplest infinite behaviour, which is divergence itself, considering two examples: λ-calculus and a simple imperative FJ-like language. In both cases we defined a big-step semantics through an inference system with corules, where divergence is explicitly modelled by a new result, denoted by ∞. To properly capture the semantics of divergence, we have added specific rules for divergence propagation; for instance, for the λ-calculus, the following rules:

$$\text{(DIV-L)}\frac{e_1 \Rightarrow \infty}{e_1\ e_2 \Rightarrow \infty} \qquad \text{(DIV-R)}\frac{e_1 \Rightarrow v \quad e_2 \Rightarrow \infty}{e_1\ e_2 \Rightarrow \infty}$$

$$\text{(DIV-APP)}\frac{e_1 \Rightarrow \lambda x.e \quad e_2 \Rightarrow v \quad e[v/x] \Rightarrow \infty}{e_1\ e_2 \Rightarrow \infty}$$

The intuition behind such rules is, that if one of the premises diverges, then the conclusion should diverge as well. The problem now is how to interpret the extended definition: the inductive interpretation is not enough because there is no way to introduce divergence[2], but the coinductive interpretation allows to derive too many judgements for non-terminating programs.

Hence, to capture the intended semantics, we designed appropriate corules. Intuitively, corules specify when we are allowed to use coinduction. In this case, they shoud be applied only to derive divergence; thus we added a coaxiom with conclusion $e \Rightarrow \infty$ for each e.

Having defined big-step semantics including divergence, we started studying how to prove properties of non-terminating programs. The first one we have considered is type soundness, which can be rephrased as completeness of the big-step semantics with respect to the type system, hence it can be proved using a variation of the bounded coinduction principle.

In [5] we have modeled richer forms of infinite behaviour, notably *trace semantics*, which, in addition to the final result, produces a trace of events observed during the computation. This notion smoothly adapts to infinite behaviour, since it is enough to consider also infinite traces.

We have considered two example languages: a λ-calculus with output effects and an imperative FJ-like language with I/O primitives; in the former case traces contain printed values, in the latter I/O events. As for the simpler case of divergence, we have added rules for divergence propagation and appropriate corules, which in this case are more complex. Consider for instance those for the λ-calculus:

$$\text{(CO-EMPTY)}\frac{}{e \Rightarrow \langle \infty,\ \varepsilon \rangle} \qquad \text{(CO-OUT)}\frac{e \Rightarrow \langle v,\ o \rangle}{\texttt{out}\ e \Rightarrow \langle \infty,\ o \cdot v \cdot o_\infty \rangle}$$

[2] There is no axiom with conclusion $e \Rightarrow \infty$ for some expression e.

where ε is the empty trace, o a finite trace, o_∞ a possibly infinite trace, and $\cdot$ trace concatenation. As before, corules allow coinduction only to derive diverging judgements. Coaxiom (CO-EMPTY) deals with diverging computations producing a finite trace, namely, computations printing some values and then diverging without producing any other output, as expressed by the empty trace in the conclusion. Corule (CO-OUT), instead, deals with computations producing an infinite output trace: intuitively, those which evaluate infinitely many output expressions. In such cases the infinite trace is unique, hence the corule allows any trace in the conclusion, but it checks that the output expression is actually evaluated, that is, its argument must converge.

The above considered semantics exhibit common patterns, both for rules and corules. An interesting and promising direction for further work is to abstract these patterns to a more general setting, to study general properties of semantics defined by inference systems with corules.

This common structure can be summarized as follows: all semantics with corules contain standard rules for converging computations, rules for divergence propagation and corules to allow coinduction only to derive divergence. Furthermore, the definition of semantics including divergence seems to be driven by the standard semantics for converging computations.

Following these remarks, what we plan to do is to define a *canonical* construction that, starting from a standard big-step semantics for finite computations, produces an extended semantics including diverging computations as well. We would like such general framework to subsume trace semantics and, more generally, other kinds of semantics describing the observable behaviour of programs.

To this end, we need to identify a general algebraic structure for *observations*, and to define a general notion of *semantics with observations*, that is, a semantics which, in addition to the final result, produces also an observation, describing the observable behaviour of the program.

Then, the canonical extension will require to introduce a new result for divergence, extend observations (and their structure) to include infinite ones and to define appropriate divergence propagation rules and corules.

To guarantee the correctness of the proposed construction, we plan to compare the resulting semantics with more standard techniques, such as labelled transition systems. This comparison could be a *parametric proof of equivalence*, which, starting from suitable hypothesis on the semantics for finite computations, proves the equivalence of the entire semantics.

Other possible, more long-term, directions for further developments are a deeper comparison with definitional interpreters, notably those based on the partiality monad [9], and the study of other examples with different notions of infinite behaviour. The former could provide interesting hints for implementation in proof assistants, for instance Agda [15,21], since definitional interpreters are naturally supported by such tools; the latter could higlight the capabilities of corules to support formal reasoning on infinite computations.

References

1. Aczel, P.: An introduction to inductive definitions. In: Barwise, J. (ed.) Handbook of Mathematical Logic, Studies in Logic and the Foundations of Mathematics, vol. 90, pp. 739 – 782. Elsevier (1977)
2. Ancona, D.: Soundness of object-oriented languages with coinductive big-step semantics. In: Noble, J. (ed.) 26th European Conference on Object-Oriented Programming, ECOOP 2012. Lecture Notes in Computer Science, vol. 7313, pp. 459–483. Springer (2012). https://doi.org/10.1007/978-3-642-31057-7_21
3. Ancona, D., Dagnino, F., Zucca, E.: Generalizing inference systems by coaxioms. In: Yang, H. (ed.) 26th European Symposium on Programming, ESOP 2017. Lecture Notes in Computer Science, vol. 10201, pp. 29–55. Springer (2017). https://doi.org/10.1007/978-3-662-54434-1_2
4. Ancona, D., Dagnino, F., Zucca, E.: Reasoning on divergent computations with coaxioms. Proceedings of ACM on Programming Languages **1**(OOPSLA) (2017). https://doi.org/10.1145/3133905
5. Ancona, D., Dagnino, F., Zucca, E.: Modeling infinite behaviour by corules. In: Millstein, T.D. (ed.) 32nd European Conference on Object-Oriented Programming, ECOOP 2018. LIPIcs, vol. 109, pp. 21:1–21:31. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018). https://doi.org/10.4230/LIPIcs.ECOOP.2018.21
6. Capretta, V.: General recursion via coinductive types. Logical Methods in Computer Science **1**(2) (2005). https://doi.org/10.2168/LMCS-1(2:1)2005
7. Charguéraud, A.: Pretty-big-step semantics. In: Felleisen, M., Gardner, P. (eds.) 22nd European Symposium on Programming, ESOP 2013. Lecture Notes in Computer Science, vol. 7792, pp. 41–60. Springer (2013). https://doi.org/10.1007/978-3-642-37036-6_3
8. Cousot, P., Cousot, R.: Inductive definitions, semantics and abstract interpretations. In: Sethi, R. (ed.) 19th Annual ACM Symposium on Principles of Programming Languages, POPL 1992. pp. 83–94. ACM Press (1992). https://doi.org/10.1145/143165.143184
9. Danielsson, N.A.: Operational semantics using the partiality monad. In: Thiemann, P., Findler, R.B. (eds.) 17th ACM International Conference on Functional Programming, ICFP 2012. pp. 127–138. ACM Press (2012). https://doi.org/10.1145/2364527.2364546
10. Kahn, G.: Natural semantics. In: Brandenburg, F., Vidal-Naquet, G., Wirsing, M. (eds.) 4th Annual Symposium on Theoretical Aspects of Computer Science, STACS 1987. Lecture Notes in Computer Science, vol. 247, pp. 22–39. Springer (1987). https://doi.org/10.1007/BFb0039592
11. Leroy, X., Grall, H.: Coinductive big-step operational semantics. Information and Computation **207**(2), 284–304 (2009). https://doi.org/10.1016/j.ic.2007.12.004
12. Nakata, K., Uustalu, T.: Trace-based coinductive operational semantics for while. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) Theorem Proving in Higher Order Logics, TPHOLs 2009. Lecture Notes in Computer Science, vol. 5674, pp. 375–390. Springer (2009). https://doi.org/10.1007/978-3-642-03359-9_26
13. Nakata, K., Uustalu, T.: A Hoare logic for the coinductive trace-based big-step semantics of while. In: Gordon, A.D. (ed.) 19th European Symposium on Programming, ESOP 2010. Lecture Notes in Computer Science, vol. 6012, pp. 488–506. Springer (2010). https://doi.org/10.1007/978-3-642-11957-6_26
14. Nakata, K., Uustalu, T.: Resumptions, weak bisimilarity and big-step semantics for while with interactive I/O: an exercise in mixed induction-coinduction. In:

Aceto, L., Sobocinski, P. (eds.) Structural Operational Semantics, SOS 2010. Electronic Proceedings on Theoretical Computer Science, vol. 32, pp. 57–75 (2010). https://doi.org/10.4204/EPTCS.32.5
15. Norell, U.: Towards a practical programming language based on dependent type theory. Phd thesis, Chalmers University of Technology and Göteborg University (2007)
16. Owens, S., Myreen, M.O., Kumar, R., Tan, Y.K.: Functional big-step semantics. In: Thiemann, P. (ed.) 25th European Symposium on Programming, ESOP 2016. Lecture Notes in Computer Science, vol. 9632, pp. 589–615. Springer (2016). https://doi.org/10.1007/978-3-662-49498-1_23
17. Plotkin, G.D.: A structural approach to operational semantics. Tech. rep., Aarhus University (1981)
18. Plotkin, G.D.: A structural approach to operational semantics. Journal of Logic and Algebraic Programming **60-61**, 17–139 (2004)
19. Reynolds, J.C.: Definitional interpreters for higher-order programming languages. In: ACM 72, Proceedings of the ACM annual conference. vol. 2, pp. 717–740. ACM Press (1972)
20. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics **5**(2), 285–309 (1955)
21. The Agda Team: The Agda Wiki (2018), http://wiki.portal.chalmers.se/agda/pmwiki.php