https://doi.org/10.1007/s00165-019-00497-w
BCS © 2019
Formal Aspects of Computing (2019) 31: 733–749

Formal Aspects of Computing

Bisimulation and Coinduction Enhancements: A Historical Perspective

Damien Pous[1] and Davide Sangiorgi[2,3]

[1]Univ Lyon, CNRS, EnsL, UCBL, LIP, 69342, Lyon Cedex 07, France
[2]University of Bologna, Bologna, Italy
[3]INRIA, Rocquencourt, France

Abstract. Bisimulation is an instance of coinduction. Both bisimulation and coinduction are today widely used, in many areas of Computer Science, as well as outside Computer Science. Over, roughly, the last 25 years, enhancements of the principles and methods related to bisimulation and coinduction (i.e., techniques to make proofs shorter and simpler) have become a research topic on its own. In the paper the origins and the developments of the topic are reviewed.

1. Introduction

Bisimilarity has emerged as one of the most robust concepts discovered in Concurrency Theory, and is today widely used in Computer Science. It is also used outside Computer Science, in areas such as Mathematics and Cognitive Science. Bisimulation has also spurred the study of coinduction; indeed bisimilarity is an example of a coinductive definition.

Bisimilarity was introduced (formulated by Park [Par81], refining ideas from Milner [Mil80]) as the notion of behavioural equality for processes. The meaning of equality on processes has produced a rich and profound debate (yet not exhausted) in Concurrency Theory, particularly in the 1970s and 1980s. The insights so produced have made an immense contribution to establish the foundations of the area.

Bisimilarity is usually defined as the union of all bisimulations. And a bisimulation is a relation on the terms of a language that is invariant under the observables of the language (i.e., what can be observed of the terms). Thus the definition itself immediately leads to a well-established proof technique for bisimilarity: to prove two terms bisimilar, find a bisimulation relation containing the two terms as a pair. This has turned out to be a powerful proof method and one of the reasons for the success of bisimilarity. Indeed, in contrast with the common inductive proof principle, the method can be naturally employed on terms denoting possibly infinite behaviours.

Over the years several enhancements of the proof method have been put forward, with the goal of making it more effective (easier to use, both in paper proofs and in tools for automated or semi-automated analysis) and more broadly applicable. For instance, in languages for process mobility or in higher-order languages, the bisimilarity enhancements appear necessary to be able to carry out any non-trivial proofs of equality. Over, say, the last 25 years, the bisimulation enhancements have become a research topic on its own. Theories of enhancements have been proposed, with an algebraic flavour, and with connections to abstract mathematical structures such as complete lattices and categories of coalgebras.

Correspondence to: D. Sangiorgi, E-mail: Davide.Sangiorgi@gmail.com

The objective of this paper is to track the history of the progress that has been made in the topic of enhancements of the bisimulation proof method. We stress that the goal here is not to report the technical details of the enhancements—though we will give intuitions and appropriate references. Moreover, we will not report on the discovery of bisimilarity and coinduction; for this piece of history, see [San09]. Sometimes our search has not been easy: especially at the beginning, enhancements have often been used as 'minor' auxiliary tools, with little or no effort in isolating the concepts and crediting the relevant papers.

Bisimilarity and the bisimulation proof method are instances of a coinductive definition and of the coinduction proof method. Although we will not discuss coinduction here, the reader should bear in mind that the enhancements outlined in the current paper for bisimulation apply to other coinductively defined notions, including preorder relations; see [PS12] for a presentation of the coinduction enhancements following fixed-point theory.

A special treatment is devoted to weak bisimilarity, that is, bisimilarity that distinguishes the observables of a term from its internal activity. In programming languages, for instance, these forms of bisimilarity are pragmatically the most relevant ones. This refinement makes the theory of enhancements considerably more difficult than in the strong case.

We first review the definition of bisimilarity, in Sect. 2. Then, in Sect. 3, we explain how the first basic forms of enhancements were introduced. In Sect. 4 we review more advanced enhancements that were studied subsequently, in various settings. In Sect. 5 we discuss progresses that were made toward an algebraic and compositional theory of. Enhancements for weak forms of bisimilarity are discussed in Sect. 6. We finally describe the parallel development of enhancements in category theory, using coalgebra, in Sect. 7.

Notation. Given two sets X, Y, we write $\mathcal{P}(X)$ for the set of subsets of X, $X \times Y$ for the Cartesian product of X and Y, and X^Y for the set of functions from Y to X. We write 2 for the set with two elements (Booleans). We let $\mathcal{R}$ range over relations on sets, i.e., elements of $\mathcal{P}(X \times X)$ for some set X. We write $\mathcal{RS}$ for the composition of two relations (i.e., $(x, z) \in \mathcal{RS}$ holds if there exists y with $(x, y) \in \mathcal{R}$ and $(y, z) \in \mathcal{S}$). We often use the infix notation for relations, writing $x\,\mathcal{R}\,y$ for $(x, y) \in \mathcal{R}$.

2. Bisimulation

We present bisimulation on *Labelled Transition Systems* (LTSs) because these are the most common structures on which bisimulation has been studied. LTSs are essentially labelled directed graphs.

Definition 2.1 (Labelled Transition Systems) A *Labelled Transition System* is a triple $(Prc, Act, \xrightarrow{\mu})$ where Prc is a set of *states*, Act is a set *labels*, and for each label $\mu \in Act$, $\xrightarrow{\mu}$ is a relation on Prc called the *transition relation*.

In the definition above, the elements of Prc will be called *states* or *processes* as this is the usual terminology in concurrency. We use P, Q to range over such elements, and μ to range over the labels in Act. Following the infix notation for relations, we write $P \xrightarrow{\mu} Q$ when $(P, Q) \in \xrightarrow{\mu}$; in this case we call Q a *μ-derivative of P*, or simply a *derivative of P*. We sometimes consider LTSs in which the states are produced by a grammar; for instance the terms of the process language CCS [Mil89]. In these cases the LTS is often defined by means of rules in the style of Plotkin's Structured Operational Semantics (SOS) [Plo04a, Plo04b].

Definition 2.2 (Bisimulation) A binary relation $\mathcal{R}$ on the states of an LTS is a *bisimulation* if whenever $P\,\mathcal{R}\,Q$:

1. for all P', μ with $P \xrightarrow{\mu} P'$ there is Q' such that $Q \xrightarrow{\mu} Q'$ and $P'\,\mathcal{R}\,Q'$;
2. the converse, on the derivatives of Q.

Bisimilarity, written $\sim$, is the union of all bisimulations; thus $P \sim Q$ holds if there is a bisimulation $\mathcal{R}$ with $P\mathcal{R}Q$.

The definition immediately suggests a proof technique: to demonstrate that P and Q are bisimilar, find a bisimulation relation containing the pair (P, Q). This is the *bisimulation proof method*. We will not discuss here the effectiveness of this proof method; the interested reader may consult concurrency textbooks in which bisimilarity is taken as the main behavioural equivalence for processes, such as [Mil89, San12, SR12].

In the remainder of the chapter we often write challenge-response clauses along the lines of the those in Definition 2.2: the universal and existential quantifiers in clauses (1) and (2) of Definition 2.2 can be used to read the definition of bisimulation as a game, see e.g., [San12]. For simplicity we will omit these quantifiers; for instance clause (1) above would be thus written:

1. if $P \xrightarrow{\mu} P'$ then $Q \xrightarrow{\mu} Q'$ and $P' \mathcal{R} Q'$.

The definition of bisimilarity is an instance of a *coinductive definition*, and the bisimulation proof method an instance of the *coinduction proof method* [MT91]. In the same way, the enhancements of the bisimulation proof method discussed in the following sections are instances of enhancements of the coinduction proof method [PS12].

3. Early ad-hoc enhancements

Bisimulation enhancements have been introduced as a technical tool to simplify bisimulation proofs, so to allow more flexibility in the requirements for matching derivatives (the requirement $P' \mathcal{R} Q'$ of Definition 2.2(1)).

The key observation is that a bisimulation relation often contains redundancies. What is a redundancy? Intuitively a pair in a bisimulation $\mathcal{R}$ may be regarded as *redundant* if it can be inferred from other pairs in $\mathcal{R}$ using certain reasonning rules. Usually those rules correspond to properties of bisimilarity, such as transitivity and substitutivity. However this is by no means a rule: there can be surprising counterexamples, see e.g., Sect. 6 or [PS12].

3.1. The first enhancement: bisimulation up to bisimilarity

There is little doubt that the idea of enhancement is due to Milner, with 'bisimulation up to bisimilarity'. It is more difficult to trace the first documented occurrence. This is however likely to be Milner's influential paper on synchrony and asynchrony [Mil83], where the technique is used to prove a substitutivity property of bisimilarity in recursive definitions. However, in the paper the technique is not introduced in the way today we are used to (as found, e.g., in the book [Mil89] and as presented in Definition 3.1 below): a 'bisimulation up to bisimilarity' is simply a relation $\mathcal{R}$ such that $\sim \mathcal{R} \sim$ is a bisimulation. No 'game conditions' are proposed, though the use of the enhancement in the paper corresponds to the game of Definition 3.1. Similar uses of the enhancement appear in works in the following years that deal with substitutivity of bisimulation-like relations under recursion, e.g., [Mil87, Wal87].

Milner realises that the technique can be used more generally to remove certain annoying redundancies found in bisimulation relations. His observation is well exemplified in the way in which the technique is introduced in his landmark book on CCS [Mil89]. Milner has to prove a simple result, namely that a binary semaphore (as originally conceived by Dijkstra; Milner uses `get` and `put` for the operations called P and V by Dijkstra) can be implemented as a parallel composition of two unary semaphores. A unary semaphore is written as a recursive definition:

$$\mathtt{Sem} \triangleq \mathtt{get.put.Sem}$$

In the CCS syntax [Mil89], $\mu.P$ is an action prefixing (the action μ should be performed first, and then the continuation P is run). Similarly, a binary semaphore $\mathtt{Sem}_2(0)$ is specified as follows:

$$\begin{aligned} \mathtt{Sem}_2(0) &\triangleq \mathtt{get.Sem}_2(1) \\ \mathtt{Sem}_2(1) &\triangleq \mathtt{get.Sem}_2(2) + \mathtt{put.Sem}_2(0) \\ \mathtt{Sem}_2(2) &\triangleq \mathtt{put.Sem}_2(1). \end{aligned}$$

where + is the operator of sum (or choice). We refer to [Mil89] for the SOS semantics of CCS. A natural equality to be proved is $\mathtt{Sem} \mid \mathtt{Sem} \sim \mathtt{Sem}_2(0)$, where '|' is parallel composition. A bisimulation that demonstrates such a

result is, for $\mathtt{Sem}' \triangleq \mathtt{put.Sem}$,

$$\begin{aligned}\mathcal{S} \triangleq \{ &(\mathtt{Sem}_2(0), \mathtt{Sem} \mid \mathtt{Sem}) \\ &(\mathtt{Sem}_2(1), \mathtt{Sem} \mid \mathtt{Sem}') \\ &(\mathtt{Sem}_2(1), \mathtt{Sem}' \mid \mathtt{Sem}) \\ &(\mathtt{Sem}_2(2), \mathtt{Sem}' \mid \mathtt{Sem}') \ \}\end{aligned}$$

(The result actually holds for any n, the case $n = 2$ is the simplest.) In $\mathcal{S}$, the pairs $(\mathtt{Sem}_2(1), \mathtt{Sem} \mid \mathtt{Sem}')$ and $(\mathtt{Sem}_2(1), \mathtt{Sem}' \mid \mathtt{Sem})$ differ only in the order of the components of the parallel composition. Since parallel composition is commutative for bisimilarity (i.e., the law $P \mid Q \sim Q \mid P$ holds), the diagram chasing arguments on one pair imply those on the other pair. In other words, it should not be necessary to check clauses (1) and (2) of Definition 2.2 on both pairs. Yet, if we remove one of the pairs, the remaining relation is not a bisimulation. For instance, if $\mathcal{S}'$ is the relation without the the pair $(\mathtt{Sem}_2(1), \mathtt{Sem}' \mid \mathtt{Sem})$ then $\mathcal{S}'$ is not a bisimulation because from the pair $(\mathtt{Sem}_2(0), \mathtt{Sem} \mid \mathtt{Sem})$, the challenge $\mathtt{Sem} \mid \mathtt{Sem} \xrightarrow{\mathtt{get}} \mathtt{Sem}' \mid \mathtt{Sem}$ cannot be matched by $\mathtt{Sem}_2(0)$. However, $\mathcal{S}'$ is a *bisimulation up to* $\sim$.

Definition 3.1 A relation $\mathcal{R}$ on processes is an *bisimulation up to* $\sim$ if whenever $P \mathcal{R} Q$,

1. if $P \xrightarrow{\mu} P'$ then $Q \xrightarrow{\mu} Q'$ and $P' \sim \mathcal{R} \sim Q'$;
2. the converse, on the derivatives of Q.

We recall that the relational composition $\sim\mathcal{R}\sim$ means that there are P'', Q'' with $P' \sim P''$, $Q' \sim Q''$, and $P'' \mathcal{R} Q''$. In the proof of soundness of the technique, one shows that the relation $\sim\mathcal{R}\sim$ (that contains $\mathcal{R}$ because $\sim$ is reflexive) is a bisimulation. Intuitively, soundness exploits the transitivity of bisimilarity: instead of proving the bisimilarity of the derivatives P', Q' we prove that for P'' and Q'', with the proviso that $P' \sim P''$ and $Q' \sim Q''$. Thus from $P'' \sim Q''$ we can derive $P' \sim Q'$ by transitivity.

In Milner's example above about semaphores, the smaller relation $\mathcal{S}'$ is indeed a bisimulation up to $\sim$. The problematic diagram-chasing argument on the challenge $\mathtt{Sem} \mid \mathtt{Sem} \xrightarrow{\mathtt{get}} \mathtt{Sem}' \mid \mathtt{Sem}$ is now solved using $\mathtt{Sem}_2(1)$ as an answer: $\mathtt{Sem}_2(1)$ is related in $\mathcal{S}'$ to $\mathtt{Sem} \mid \mathtt{Sem}'$, which is strongly bisimilar to $\mathtt{Sem}' \mid \mathtt{Sem}$.

The example brings up the essence of bisimulation enhancements, namely the possibility of carrying out proofs using relations that are not themselves bisimulations, as required in the ordinary bisimulation proof method, but contained in bisimulations. And while in this specific example the benefits of the enhancement are quite limited, in general they can be substantial. For instance, generalising the above example to n semaphores, the enhancement would allow us to save exponentially many pairs. Several non-trivial proofs in Milner's book [Mil89] make use of the technique, often in connection with weak bisimilarity (Sect. 6).

3.2. Self-bisimulations

In [Cau90], Caucal defines a notion of *self-bisimulation* in the setting of BPA processes (they can be viewed as the processes generated by a context-free grammar) that allows him to eliminate common prefixes and suffixes in the derivatives of two processes. For instance, if P and Q are processes of a self-bisimulation and P has a transition $P \xrightarrow{\mu} R.P'$, then Q may answer with the transition $Q \xrightarrow{\mu} R.Q'$ if P' and Q' are also in the self-bisimulation relation (the common prefix R has been cancelled). Self-bisimulations have been used in [Cau90], as well as in a number of other papers (e.g., [CHS95, HJM96b, HJM96a]), to establish decidability results for the classes of BPA and BPP processes (roughly, the latter differ from the former in that the composition operator is commutative). Caucal's use of a bisimulation enhancement is interesting because it is more than just a proof simplification: it is an essential tool to produce the decidability results. The key idea is that while bisimulations usually are infinite, one can show in this settings that any pair of equivalent processes is related by a *finite* self-bisimulation.

3.3. Other enhancements

In [MPW89] (an earlier handwritten note with the technique is [Par87]), Milner, Parrow and Walker introduce *bisimulation up to restriction*, as a way of removing, in the derivatives of two π-calculus processes on which the bisimulation game is played, common outermost restrictions. The technique is introduced to simplify the proof

of substitutivity of bisimilarity with respect to the operator of parallel composition. The simplification takes care of the dynamic creation of fresh names—the extrusion of the scope of a restriction, something that does not happen in CCS. The technique is used to prove a few other laws, also combined with up-to-bisimilarity, obtaining 'bisimulation up to bisimilarity and restriction'.

4. Theories of enhancements

Until mid 1990s most of enhancements are forms of 'bisimulation up to bisimilarity' (for various kinds of bisimilarity, including strong and weak versions, see also Sect. 6). Enhancements are viewed as auxiliary tools to simplify proofs. Generally, the simplifications, while elegant, are not critical, in that a proof without the enhancement would not have been much more complicated—with the exception of Caucal's self-bisimulations in the decidability proofs recalled earlier.

The situation changes in the 1990s with the development of operational theories of languages for name mobility such as the π-calculus [MPW89] and its many dialects, and of languages including higher-order features (where variables may be instantiated with arbitrary terms), such as λ-calculi (following on Abramsky's applicative bisimilarity [Abr90]), CHOCS [Tho89], Higher-order π-calculus [San92], Mobile Ambients [CG98], and so on. In these languages the enhancements seem essential to be able to obtain any non-trivial proof: defining an appropriate bisimulation can be considerably hard (i.e., a bisimulation relation containing the pairs of interest), let alone carrying out the whole proof.

The study of enhancements, as a topic on its own, is proposed in [San95]. This means both understanding existing enhancements and looking for new forms of enhancements, and (above all) studying theories of enhancements, with an algebraic flavour, in which complex up-to techniques are derived by composing simpler techniques by means of appropriate operators. The paper [San95] focuses on bisimilarity (in fact, strong bisimilarity). It introduces a few new forms of enhancements (e.g., 'up to context') and makes a proposal for an algebra of enhancements, viewing enhancements as functions on relations (in the algebra, 'up to bisimilarity' turns out to be derivable from a few constant functions). Following work, in particular Pous [Pou07a, Pou16] refines all this and makes it even more general, as a fixed-point theory applicable to coinductive objects other than bisimilarity. We recall below 'up to context' and a few other enhancements, deferring algebras of enhancements to Sect. 5.

4.1. Up to context

The enhancement called *'up to context'* [San95] (the technique had actually already appeared, in the π-calculus, in [San94], and is anticipated in the conclusions of [SM92]) allows us to cancel a common context in matching derivatives. Here we are assuming—without getting into the mathematical details—that the process language is defined by means of a grammar. We use C to range over context, i.e., terms with a hole.

Definition 4.1 A relation $\mathcal{R}$ on processes is a *bisimulation up to context* if whenever $P\mathcal{R}Q$,

1. if $P \xrightarrow{\mu} P'$ then $Q \xrightarrow{\mu} Q'$ and there is a context C, and processes P'', Q'' with $P' = C[P'']$, $Q' = C[Q'']$, and $P''\mathcal{R}Q''$;
2. the converse, on the derivatives of Q.

In this case, intuitively, the soundness of the technique relies on the substitutivity of bisimilarity, i.e., the fact that bisimilarity is preserved by contexts. Hence from bisimilarity of P'' and Q'' we can infer bisimilarity of the derivatives $C[P'']$ and $C[Q'']$. Up-to-restriction is a special case of the up-to-context technique.

The up-to-context technique is important in languages that include name mobility or higher-order features. Intuitively, in these languages terms may move; for instance, the values that are passed around may contain 'code'. As a consequence, the ways in which a given term may evolve depend on what its outside environment provides. Then up-to-context may allow one to separate concerns; for instance the contribution of the outside environment from the rest of the term. While introduced in concurrency, the technique has been extensively studied in λ-calculi, beginning with Pitts [Pit95], Lassen [Las98a, Las98b, Las99], Sands [San98]; recent studies include Dal Lago and Gavazzo [LG19, Gav19]. Without up-to-context, in these languages bisimulation alone would be often rather cumbersome to use, even on small examples, particularly when bisimulation is in the

'environmental' style (Sect. 4.2). In fact up-to-context has sometimes been hardwired into the definition itself of bisimulation, e.g., [KW06]. Interesting examples of uses of the techniques include those by Merro and Zappa Nardelli to derive algebraic properties of the Ambient calculus [MN05], and by Aristizabal et al. [ABLP17] for a λ-calculus with delimited-control operators. (Note that these are all uses of 'weak' forms of bisimulations, as discussed in Sect. 6.)

The up-to-context techniques can however be useful also in ordinary (as opposed to 'higher-order') languages, to exploit the structure of the studied objects. See Sect. 4.3 for a striking example.

4.2. Other forms of enhancement

In *'up to injective substitutions'* [San95], one is allowed to close the bisimulation game using an injective renaming on the free identifiers of the derivatives. Again this technique makes sense on terms that have a structure and where, therefore, it is possible to talk about 'free objects' such as identifiers (or names). The technique is useful in languages in which, during the bisimulation game, name substitutions may be applied to terms (e.g., name-passing calculi such as the π-calculus), or where the observables of the bisimulation game include binders with universal quantifications on the possible choices for instantiation of the binders. The enhancement intuitively exploits the invariance of bisimilarity under injective substitutions.

The previous enhancements exploit basic properties of bisimilarity, such as transitivity, substitutivity, invariance under injective renaming. Bisimilarity has however been used on a variety of languages, sometimes taking shapes much richer than that of Definition 2.2. For instance, the language may be typed and the pairs on which the bisimulation game is played may become triples so to accommodate a typing environment. The extra component may also be an environment that intuitively collects the knowledge that the external observer has so far accumulated about the values received from the tested terms. This kind of bisimulations has appeared in concurrency [PS97, BS98a, AG98, BDP99] and has then been widely used in λ-calculi, e.g., [SP04, SP05, KW06, SKS07]. It may also be that the tested terms themselves are enriched with extra components, for instance representing a store [KW06], or an execution stack [JPR09]. The tested terms may also be collections of terms, for instance probability distributions as in forms of bisimilarity on languages with probabilities [SV16, CPV16].

In these cases, bisimulation enhancements may be introduced to be able to manipulate the extra components so added, exploiting properties of bisimilarity on such components. For instance, a typing environment may be modified by removing entries that mention identifiers that do not appear in the terms, or by strengthening or weakening the types by applying appropriate subtyping relations. A number of enhancements have been proposed along these lines; others adapt standard enhancements to such enriched settings. The range of possibilities is too wide for us to be able to mention all of them. It is however at least worth mentioning that the case of transition systems exposing probabilities (e.g., the probability that a certain transition will occur) is delicate, sometimes even in (apparently) basic enhancements such as up-to-bisimilarity; see e.g., the bisimulation up to Markovian bisimulation equivalence in [BBG98], a development of Milner's bisimulation up to bisimilarity. Enhancements for languages with probabilities or metrics have been studied by Vignudelli et al. [SV16, CPV16], and by Bonchi et al. [BKK17, BKP18]. Enhancements were also used recently for the analysis of systems of polynomial ordinary differential equations [Bor19]. Generally these are rather recent contributions, and the area remains a hot research topic.

4.3. Enhancements for automata algorithms

In the early 1970s [HK71], Hopcroft and Karp proposed a simple algorithm to check language equivalence on deterministic finite automata (DFA), using a so-called 'union-find' data structure to record equivalence classes. Tarjan subsequently proved that this algorithm is almost linear [Tar75]. The algorithm is nowadays recognised as a coinductive one: language equivalence in a DFA can be characterised as the largest bisimulation (for an appropriate notion of bisimulation on DFA), so that to check language equivalence of two states, it suffices to look for a bisimulation containing them. Applying this coinductive technique naively however only leads to a quadratic algorithm. To achieve almost linear time complexity, Hopcroft and Karp exploit the equivalence property. Using modern terminology, their algorithm looks for bisimulations up to equivalence rather than plain bisimulations. By doing so, they are able to reduce the search space: bisimulations up to equivalence have size at most linear (while bisimulations can have quadratic size).

This implicit use of enhanced coinduction was noticed by Bonchi and Pous [BP13, BP15], who extended the idea to non-deterministic finite automata (NFA). Deciding language equivalence for NFA is harder: the problem is PSPACE-complete. It can be solved by using the powerset construction to determinise the automata, and then applying Hopcroft and Karp's algorithm. Such a procedure requires exponential time and space in worst case since the determinised automata may contain exponentially many reachable states. A key observation is that the states of the determinised automata bear some structure: determinised states are subsets of states from the initial automata, and the language of a union of subsets is precisely the union of the languages of those subsets (in symbols, $L(X \cup Y) = L(X) \cup L(Y)$). This structure makes it possible to define further enhancements and to improve the algorithm. First, one can use bisimulations up to context by considering set-theoretical union as a syntactic operator. Second, one can combine up-to-equivalence from Hopcroft and Karp's algorithm with this notion of up-to-context in order to obtain an up-to-congruence technique. (The resulting technique is similar in spirit to Caucal's self-bisimulations [Cau90] mentioned in Sect. 3.2.) While the worst case theoretical complexity remains the same, the technique can yield significant efficiency improvements: a bisimulation up to congruence does not need to explore all reachable subsets of the initial automata, so that the resulting algorithm often solves in polynomial time families of NFA whose determinised automata have exponentially many states.

5. Compositions and algebras of enhancements

Different up-to techniques may sometimes be composed, so to magnify their usefulness. Examples of this have been given in Sect. 4.3, in connection with up-to-context. Indeed in an up-to-context it seldom happens that the common context already appears in the two derivatives, as required by Definition 4.1. Usually the derivatives need some massage to bring the context out. The massaging can be for instance achieved by applying some algebraic laws for bisimilarity, which means combining up-to-context with up-to-bisimilarity. Clause (1) of Definition 4.1 thus becomes:

1. if $P \xrightarrow{\mu} P'$ then $Q \xrightarrow{\mu} Q'$ and there is a context C, and processes P'', Q'' with $P' \sim C[P'']$, $Q' \sim C[Q'']$, and $P'' \mathcal{R} Q''$;

Early occurrences of combinations of up-to techniques include Milner et al. [MPW89] 'bisimulation up to bisimilarity and restrictions', mentioned in Sect. 3.3 (it is a special case of the composition above, involving up-to-context); and Caucal's self-bisimulations [Cau90] as well as bisimulations up to congruence for non-deterministic automata [BP13, BP15], which combine up-to-equivalence with a form of up-to-context, discussed in Sects. 3.2 and 4.3.

Nevertheless, combinations of up-to techniques remain rather rare and always ad hoc, until mid of the 1990s. A theory of up-to techniques, with the possibility of combining them, is the main contribution in [San95]. In the paper, a 'bisimulation up to' is a relation $\mathcal{R}$ for which one can play the bisimulation game and relate the derivatives in a larger relation $\mathcal{S}$. This motivates the following notion of *progression*: Given two relations $\mathcal{R}$ and $\mathcal{S}$, we say that $\mathcal{R}$ *progresses to* $\mathcal{S}$, written $\mathcal{R} \rightarrowtail \mathcal{S}$ if, whenever $P \mathcal{R} Q$

1. if $P \xrightarrow{\mu} P'$ then $Q \xrightarrow{\mu} Q'$ and $P' \mathcal{S} Q'$;
2. the converse, on the derivatives of Q.

When $\mathcal{R}$ and $\mathcal{S}$ coincide, the above clauses are the ordinary ones for the definition of a bisimulation relation.

Using this definition one can view enhancements as functions $\mathcal{F}$ from relations to relations which are *sound* with respect to bisimilarity, i.e., such that for all relation $\mathcal{R}$, $\mathcal{R} \rightarrowtail \mathcal{F}(\mathcal{R})$ entails $\mathcal{R} \subseteq \sim$. For instance, up-to-bisimilarity corresponds to the function that given a relation $\mathcal{R}$ returns the composite relation $\sim \mathcal{R} \sim$; up-to-equivalence corresponds to the function returning the equivalence closure of a relation; up-to-context to that returning its contextual closure. Relevant questions are: which functions are sound? Which properties are satisfied by the class of sound functions? Which conditions ensure soundness of functions?

It happens that the class of sound functions is not compositional: there are sound functions whose pointwise union is not sound, and similarly for other function constructors, e.g., composition. This means that one cannot freely use two sound functions in a bisimulation proof. To circumvent this difficulty, [San95] suggests a simple functorial-like condition, called *respectfulness*. This condition requires that if $\mathcal{R} \subseteq \mathcal{S}$ and $\mathcal{R} \rightarrowtail \mathcal{S}$ hold, then $\mathcal{F}(\mathcal{R}) \subseteq \mathcal{F}(\mathcal{S})$ and $\mathcal{F}(\mathcal{R}) \rightarrowtail \mathcal{F}(\mathcal{S})$ must hold too. The paper then goes on to prove the soundness of respectful functions and to show that the class of respectful functions contains non-trivial functions and to study the closure

properties of the class with respect to various important function constructors, like composition, union, iteration, chaining (chaining gives us relational composition). These properties allow one to construct sophisticated sound functions—and hence sophisticated proof techniques for bisimilarity—from simpler ones. For instance, bisimulation up to bisimilarity and its soundness are derived from two very simple basic functions, namely the identity function and the constant-to-$\sim$ function, which maps every relation onto $\sim$, applying the operator of chaining.

Hirschkoff [Hir99] has formally verified the theory of respectful functions in Coq, and has used the theory to develop a prototype for mechanically verifying bisimilarity results. This has been the first non-trivial mechanisation of up-to techniques. At the time it was also one of the largest software developments made in Coq.

Pous has later generalised this theory to arbitrary coinductive predicates, by stating it in the context of complete lattices [Pou07a, PS12]. The starting point in this setting is Knaster-Tarski's theorem [Kna28, Tar55]: in every complete lattice, every monotone function has a greatest fixpoint, which is obtained as the union of all post-fixpoints. Bisimilarity for all kinds of systems can be presented in this way, as well as coarser behavioural equivalences or preorders (see Sect. 6 below). For instance, the following monotone function on relations admits bisimilarity as its greatest fixpoint:

$$b : \mathcal{R} \mapsto \{(P, Q) \mid \text{if } P \xrightarrow{\mu} P' \text{ then } Q \xrightarrow{\mu} Q' \text{ and } P' \mathcal{R} Q'; \text{ and the converse on the derivatives of } Q\}$$

(Note that $\mathcal{R} \subseteq b(\mathcal{S})$ precisely if $\mathcal{R} \rightarrowtail \mathcal{S}$.) Pous proposes to simplify the notion of respectfulness to that of compatibility, which is simpler to state: a monotone function f is *compatible* with b if $f \circ b \subseteq b \circ f$ (roughly, this amounts to dropping the set-inclusion requirements in the definition of respectfulness). Compatible functions share all the good properties of respectful functions: they are sound, the composition of two compatible functions remains compatible, and the pointwise union of a family of compatible functions is compatible. They are however more restrictive: there are respectful functions that are not compatible (e.g., up-to-context).

Hur et al. [HNDV13] exploited the complete lattice setting to propose *parameterised coinduction*: a technique making it possible to perform and write coinductive proofs on the fly, without having to announce the bisimulation candidate beforehand. This is convenient because it matches the actual practice: when trying to prove something by coinduction it is in general difficult to correctly guess the appropriate bisimulation candidate from scratch. Instead, one generally starts with a small bisimulation candidate, which one enlarges whenever one realises that a new pair of processes is needed. The idea of computing bisimulations 'on the fly' was not new: algorithms for this existed in the 1990s, e.g., [FM91]. The main interest of [HNDV13] is for proof assistants, where one needs to construct proofs rather than to execute algorithms, and in the implementation of the approach in the proof assistants Isabelle/HOL and Coq. In the same paper, Hur et al. also show how to combine parameterised coinduction with enhancements, by exploiting the straightforward property that, since respectful functions are closed under arbitrary unions, there exists a largest respectful function (namely, the pointwise union of all respectful functions). As a consequence, one does not need to choose the enhancement at the beginning of a proof: one can always blindly use the largest respectful function: during the course of the coinductive proof, this function will allow us to use any function which is known to be respectful.

Pous subsequently explored the idea of a largest enhancement and proposed to focus on the largest compatible function, called the *companion* [Pou16]. This function enjoys many good properties (for instance, this is a closure operator) and, surprisingly, coincides with the largest respectful function: those functions which are respectful but not compatible are nevertheless contained in the companion. Moreover, Pous shows that the companion is itself a coinductive object, so that enhancements can be used to prove that a given function is below the companion. These second-order techniques are useful to validate enhancements like up-to-context.

The companion can also be characterised in terms of Kleene's construction of the greatest fixpoint [PW16]. Indeed, the greatest fixpoint of a function b can be obtained as the limit of a sequence over ordinals defined by iterating the function b. This sequence intuitively consists of approximations of the greatest fixpoint (e.g., for bisimilarity the nth element in the sequence is the notion of truncated bisimilarity where the bisimulation game ends after n steps—n potentially beeing an ordinal). A function is below the companion if and only if it preserves all the elements of this sequence of approximations. This approach to enhancements as 'approximation preserving' functions has been implemented in Agda [Dan18], where *sized types* (intuitively, types indexed with a bound on the size their elements) give an explicit access to the sequence of approximations: approximation preserving functions become *size-preserving* functions (at least for coinductive datatypes that are obtained by a

sequence of approximations that converges at the first infinite ordinal ω: it remains unclear whether one can go beyond this ordinal with sized types—moreover the development of a dependent type theory with sized types is still an ongoing research program [Dan18, end of p.4]).

6. Weak bisimilarity

The bisimilarity discussed in earlier sections is often too restrictive, as it does not abstract over the internal behaviour of processes. To address this problem, *weak* bisimilarity has been introduced [HM85, Mil89]: it allows processes to play the bisimulation game modulo silent transitions. By contrast, the ordinary bisimilarity is therefore often called *strong* bisimilarity. Since the process transitions are more involved and the equivalence itself is coarser, the enhancements of the proof method for weak bisimilarity are even more important than those for strong bisimilarity. Unfortunately the theory for the weak case is also more complex.

We briefly recall the definition of weak bisimilarity, referring to [San12] for more details both on its motivations and on its technicalities. First, in the LTS we distinguish a special action, τ, that represents internal activity (i.e., an internal evaluation, or a synchronisation between two processes). We call *visible* the remaining actions, and use ℓ to range over them (whereas, as before, μ ranges over all actions). We then set:

- $\Longrightarrow$ as the reflexive and transitive closure of $\stackrel{\tau}{\longrightarrow}$; i.e., $P \Longrightarrow P'$ holds if P can evolve into P' by performing some silent steps—possibly none.
- $\stackrel{\mu}{\Longrightarrow}$ as $\Longrightarrow\stackrel{\mu}{\longrightarrow}\Longrightarrow$ (the composition of the thee relations); i.e., $P \stackrel{\mu}{\Longrightarrow} P'$ holds if there are P_1 and P_2 with $P \Longrightarrow P_1$, $P_1 \stackrel{\mu}{\longrightarrow} P_2$ and $P_2 \Longrightarrow P'$.
- $\stackrel{\widehat{\mu}}{\Longrightarrow}$ as $\stackrel{\mu}{\Longrightarrow}$ if $\mu \neq \tau$, and as $\Longrightarrow$ if $\mu = \tau$.

Definition 6.1 (weak bisimulation and bisimilarity) A relation $\mathcal{R}$ is a *weak bisimulation* if whenever $P\mathcal{R}Q$:

1. for all P', μ with $P \stackrel{\mu}{\longrightarrow} P'$ there is Q' such that $Q \stackrel{\widehat{\mu}}{\Longrightarrow} Q'$ and $P'\mathcal{R}Q'$;
2. the converse, on the derivatives of Q.

Weak bisimilarity, written $\approx$, is the union of all bisimulations.

Below, when discussing enhancements we simply indicate how the requirement $P'\mathcal{R}Q'$ of clause (1) above is modified; it is intended that a similar modification is made on clause (2).

Note that in Definition 6.1, the challenges for the bisimulation game are 'strong' (using the strong transition relation $\stackrel{\mu}{\longrightarrow}$) whereas the answers are 'weak' (using the weak transition relation $\stackrel{\widehat{\mu}}{\Longrightarrow}$). It would be possible to use the weak relations also on the challenger side but this would make the associated proof method (and its enhancements) harder to use in practice, as there would be more challenges to examine.

As pointed out earlier, the most common form of bisimulation enhancement is 'bisimulation up to bisimilarity'. As in the strong case, in the weak case the first mention of the technique we have found is in Milner's paper [Mil83], used to prove the substitutivity of bisimilarity under recursion. In both cases, a 'bisimulation up to bisimilarity' is defined to be a relation $\mathcal{R}$ that is contained in $\asymp\mathcal{R}\asymp$, where $\asymp$ is the intended bisimilarity (strong or weak). (The particular use of the up-to in [Mil83] matches the requirement $(*)$ below of a 'bisimulation up-to $\sim$ and $\approx$'.) Subsequent research in the 1980s continues to treat 'bisimulation up to bisimilarity' in the weak case in the same way as in the strong case. This leads to taking a 'bisimulation up to $\approx$' to be a relation $\mathcal{R}$ in which the requirement $P'\mathcal{R}Q'$ of Definition 6.1 becomes:

$$P' \approx \mathcal{R} \approx Q' \qquad (*)$$

This appears for instance in [Mil87] (proof of unique solution of equations), in the first version of the CCS book [Mil89] (this was amended by an errata note by Milner, November 1990, concerning the theorem itself and its applications in the book, and was then finally adjusted in the second edition of the book), as well as in papers dealing with other weak behavioural relations (e.g., the divergence-sensitive preorder in [Wal87]).

Unfortunately, the combination of strong and weak transitions in (∗) makes the technique unsound. This was proved, independently, by Sjödin and Jonsson and by Sangiorgi (both being private communications to Milner, early 1990, see also [SM92, p. 35]) using more or less the same counterexample, namely $\mathcal{R} \triangleq \{(\tau.a.\mathbf{0}, \mathbf{0})\}$. The processes $\tau.a.\mathbf{0}$ and $\mathbf{0}$ are not weakly bisimilar, but $\mathcal{R}$ does satisfy the above requirements. The problems of 'bisimulation up to bisimilarity' specific to the weak case are discussed in [SM92]. A simple solution consists in replacing, on the challenger side, the occurrence of weak bisimilarity with strong bisimilarity, thereby modifying the requirement of Definition 6.1(1) with

$$P' \sim \mathcal{R} \approx Q' \qquad (**)$$

A relation satisfying these requirements is usually called a *weak bisimulation up to* $\sim$ *and* $\approx$.

However in this solution the presence of $\sim$ may represent a too heavy constraint. The goal is to replace $\sim$ with something as coarse as possible yet capable of guaranteeing soundness. The most useful solution proposed in [SM92] involves the *expansion* preorder. The idea underlying expansion is roughly that if Q expands P, then P and Q are weakly bisimilar, and in addition, during the bisimulation game P never performs more τ transitions than Q. Expansion is not an equivalence, it is just a preorder. Intuitively, expansion provides some control on the number of τ-actions performed by related processes and this is sufficient to maintain the soundness of the technique. Below, $P \stackrel{\widehat{\mu}}{\longrightarrow} P'$ holds if $P \stackrel{\mu}{\longrightarrow} P'$ or ($\mu = \tau$ and $P = P'$);

Definition 6.2 (Expansion relation) A relation $\mathcal{R}$ is an *expansion* if whenever $P\mathcal{R}Q$,

1. $P \stackrel{\mu}{\longrightarrow} P'$ implies $Q \stackrel{\mu}{\Longrightarrow} Q'$ and $P'\mathcal{R}Q'$ for some Q'
2. $Q \stackrel{\mu}{\longrightarrow} Q'$ implies $P \stackrel{\widehat{\mu}}{\longrightarrow} P'$ and $P'\mathcal{R}Q'$ for some P'.

Q *expands* P, written $P \preceq Q$, or $Q \succeq P$, if $P\mathcal{R}Q$ for some expansion $\mathcal{R}$. The preorder $\preceq$ is the *expansion relation*.

Expansion had been proposed, some time earlier and independently, by Arun-Kumar and Hennessy [AH91], for completely different reasons, namely to study a preorder giving information about the 'efficiency' of processes.

In the *bisimulation up to expansion and bisimilarity* [SM92] the requirement $P'\mathcal{R}Q'$ of Definition 6.1(1) is replaced by the coarser $P' \succeq \mathcal{R} \approx Q'$. This form of up-to is used in several subsequent works. A number of variants exists, sometimes less powerful but easier to define. An example is 'bisimulation up to deterministic reduction', in which the requirement becomes

there is a processes P'' with $P' \Longrightarrow_{\mathrm{d}} P''$ and such that $P''\mathcal{R} \approx Q'$

where $P' \Longrightarrow_{\mathrm{d}} P''$ indicates that P' evolves into P'' in a deterministic manner, that is, the silent transitions that bring from P' to P'' are the only transitions that the processes involved may perform. If $P' \Longrightarrow_{\mathrm{d}} P''$ then $P' \succeq P''$ holds, hence the technique is less powerful; however it may be more convenient to use, as it does not require introducing an auxiliary relation such as expansion. This is how it is used by Fournet and Gonthier [FG05].

A limitation of the expansion preorder is that, for $P \preceq Q$ to hold, P must be more efficient than Q at any point in time. To relax this constraint, Pous has studied techniques that rely on termination guarantees [Pou05, Pou06, Pou07b]. Those take inspiration from rewriting theory techniques like Newman's lemma [New42] (local confluence and termination implies confluence) or *decreasing diagrams* [BKvO98] (intuitively, a method originally proposed to reduce the problem of showing confluence of a rewrite relation to showing its local confluence under the condition that the confluence diagrams are decreasing with respect to some labelling). Such techniques are quite powerful and make it possible to handle complex proofs on abstract machines [Pou08]; however, they tend to be non-compositional: using the terminology from Sect. 5, those are sound techniques which are neither compatible nor respectful.

The uses of the up-to-context for weak semantics are often coupled with up-to-expansion. In functional languages, sometimes Sands' *improvement* preorder is used in place of expansion, e.g., [Las98a, San98]. The expansion and improvement preorder reproduce the same idea, namely efficiency. A direction for exploring up-to-context and related techniques for weak bisimilarity is to use equations, and derive the techniques from theorems about unique solution of equations [DHS17]. Equations may also be replaced by preorders called contractions [San15] that play a role similar to that of expansion. The relationship between ordinary bisimulation enhancements and techniques derived from 'unique-solution theorems' is not yet fully understood.

7. Coalgebra

In category theory, coalgebras make it possible to model state based systems in a unified way (e.g., processes, automata, streams, weighted automata) [Jac16]. Given a functor F, an F-coalgebra is just an object X together with a morphism $\alpha : X \to FX$. The idea is that the functor F describes a type of state-based system, and a coalgebra for it is a system described by its state space (X), and dynamics (α). For instance, a coalgebra for $FX = \mathcal{P}(A \times X)$ is an LTS; a coalgebra for $FX = 2 \times X^A$ is a deterministic automaton on the alphabet A, a coalgebra for $FX = 2 \times \mathcal{P}(X)^A$ is a non-deterministic automaton, and so on.

A given functor F often has a *final* F-coalgebra, i.e, a coalgebra onto which every other coalgebra maps, in a unique way.

Definition 7.1 An F-coalgebra (Ω, ω) is *final* if for every F-coalgebra (X, α), there exists a unique morphism $[\cdot] : X \to \Omega$ such that the following diagram commutes:

$$\begin{array}{ccc} X & \xrightarrow{[\cdot]} & \Omega \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\omega} \\ FX & \xrightarrow[F[\cdot]]{} & F\Omega \end{array}$$

When it exists, the final coalgebra determines the denotational semantics of the considered systems. For instance, for deterministic automata ($FX = 2 \times X^A$), the final coalgebra Ω consists of formal languages over A, and a state x of a given coalgebra (i.e., of an automaton) is mapped to the language $[x]$ it recognises. This is called the *final semantics*: two states in a coalgebra are behaviourally equivalent if they are mapped to the same value in the final coalgebra.

This setting has made it possible to study two kinds of coinductive enhancements: enhancements of the associated corecursion schemes (Sect. 7.1) and enhancements of the bisimulation proof method for arbitrary state-based systems (Sect. 7.2).

7.1. Enhanced corecursion schemes

Defining an object as a final coalgebra gives us a powerful way to construct its elements (e.g., LTSs, languages, streams), simply by exhibiting appropriate coalgebras: this is the corecursion scheme, which is dual to the recursion scheme given by an initial algebra. Take for instance streams in $\mathbb{R}^{\mathbb{N}}$, which are the final coalgebra for $FX = \mathbb{R} \times X$ and which have been studied in details by Rutten from the coalgebraic point of view [Rut00, Rut05, NR11]. The first component of F intuitively corresponds to the head of the stream, and the second component to its tail.

One can define by corecursion the stream *from* (n) of natural numbers starting from a given number n by using the coalgebra $(\mathbb{N}, n \mapsto (n, n+1))$. The commuting square characterising the unique map $\mathit{from}(\cdot) : \mathbb{N} \to \mathbb{R}^{\mathbb{N}}$ obtained by finality (Definition 7.1) precisely specifies that the head of the stream *from* (n) is n, and that its tail is *from* $(n+1)$.

Similarly, one can define the function that pointwise adds two streams, by using the coalgebra $((\mathbb{R}^{\mathbb{N}})^2, (\sigma, \tau) \mapsto (\sigma_0 + \tau_0, (\sigma', \tau')))$, where for a given stream σ, σ_0 denotes its head and σ' denotes its tail. By finality, one obtains a function $\cdot + \cdot : (\mathbb{R}^{\mathbb{N}})^2 \to \mathbb{R}^{\mathbb{N}}$ such that $(\sigma + \tau)_0 = \sigma_0 + \tau_0$ and $(\sigma + \tau)' = \sigma' + \tau'$.

In both cases, we use a corecursion principle: in order to define a stream (or a family of streams), we define its head and its tail, and we are allowed to use corecursively the concept being defined for doing so. There are however constraints in order to ensure that the definition is not circular: with corecursion by finality, one only has access to a name for the currently defined stream, not to the stream itself (e.g., in the first example, $n+1$ is only a preliminary name for the stream *from* $(n+1)$ to be defined, one cannot compute its tail when definining the coalgebra). Similarly, in the second example, the pair (σ', τ') in the definition of the coalgebra is a name for the stream $\sigma' + \tau'$ being defined.

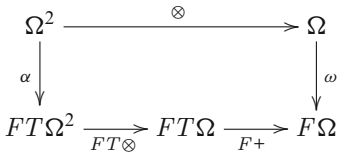
It is often convenient to relax this condition. Suppose for instance that we want to define the *shuffle product* of two streams, usually defined by the following equations:

$$(\sigma \otimes \tau)_0 = \sigma_0 \times \tau_0 \qquad (\sigma \otimes \tau)' = \sigma' \otimes \tau \; + \; \sigma \otimes \tau'$$

Due to the outer occurrence of + in the second equation, we cannot turn them into an F-coalgebra: the preliminary names for $\sigma' \otimes \tau$ and $\sigma \otimes \tau'$ are not enough to call the previously defined function +. In fact, the existence and unicity of a solution to these equations depends on the behaviour of the function +.

Now consider the functor $TX = X^2$. One can define an FT-coalgebra as follows: $((\mathbb{R}^{\mathbb{N}})^2, (\sigma, \tau) \mapsto (\sigma_0\tau_0, ((\sigma', \tau), (\sigma, \tau'))))$. Such an FT-coalgebra should be thought of as an *F-coalgebra up to T*: instead of giving directly a name for the tail, we are allowed to give two names, (σ', τ) and (σ, τ'), whose corresponding streams should be combined by a function which still remains to be specified. In this case, we want to use the function +, which is a T-algebra on the final F-coalgebra $\Omega = \mathbb{R}^{\mathbb{N}}$: a function from $T\Omega = (\mathbb{R}^{\mathbb{N}})^2$ to $\mathbb{R}^{\mathbb{N}}$.

In symbols, the function $\otimes$ is the unique function satisfying the following diagram:



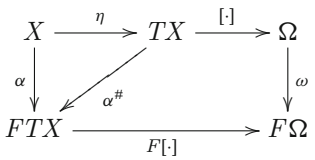
Various conditions have been proposed in the literature to ensure the existence and unicity of solutions to such equations [Bar04, LPW00, UVP01, Jac06, MMS13]. All of them essentially require that the T-algebra arises from a distributive law of T over F, i.e., a natural transformation $\lambda : TF \to FT$. This is not surprising since the use of such distributive laws is standard in many developments about coalgebra and operational semantics [TP97, Kli11].

The notion of compatible function [Pou07a, PS12] (Sect. 5) actually is a special case of such distributive laws in preorder categories. Conversely, the notion of companion (Sect. 5) can be generalised to the categorical setting [PR17, BPR17]: it becomes a final distributive law, and under certain conditions it can be computed as the codensity monad of the final sequence.

This led to the following alternative condition for the enhanced corecursion scheme above to be valid: the T-algebra should be *causal* [PR17], i.e., in the case streams, the nth value of its result should depend only on the n first values of its inputs: the operation does not perform *lookaheads*. Operations defined using the GSOS format [Plo04b] typically satisfy this condition: lookaheads are not permitted in this format. So do *size-preserving* functions from [Dan18].

The *generalised powerset construction* [SBBR10] provides another way to look at such enhancements of corecursion schemes. There, the key idea comes from the concrete example of finite automata: deterministic finite automata (DFA) denote formal languages by final semantics, and non-deterministic finite automata (NFA) can be seen as deterministic automata 'up to non-determinism'. Indeed, recall that a DFA is coalgebra for $FX = 2 \times X^A$, and let $T = \mathcal{P}$ be the powerset functor. A NFA is a coalgebra for $FTX = 2 \times \mathcal{P}(X)^A$, i.e., an F-coalgebra up to T: a state may have a set of successors along a given letter, rather than just a single successor.

The standard powerset construction makes it possible to determinise a NFA. It can be presented using the following diagram, where (Ω, ω) stands for the final F-coalgebra of formal languages.



On the left, (X, α) is a NFA, an F-coalgebra up to T. It is extended into a DFA, a plain F-coalgebra $(TX, \alpha^\#)$ with state space $TX = \mathcal{P}X$, out of which we obtain the semantics, by finality. The situation is similar to that of bisimulations up-to: 'bisimulations up to valid principles', even though they are not bisimulations, can be extended to bisimulations, and are thus contained in bisimilarity.

A nice observation from [SBBR10] is that the above construction works whenever T is a monad and when there exists a distributive law of this monad T over F (η in the diagram is the unit of the monad, and $\alpha^\#$ is obtained from α, the distributive law, and the multiplication of the monad). Thus we find again the same ingredients as before, but here the emphasis is put on the concrete F-coalgebra which is constructed.

7.2. Categorical presentation of bisimulation up to context

Aczel and Mendler [AM89] and Turi and Plotkin [TP97] showed that when the functor F preserves weak pullbacks (which many functors do), then the notion of behavioural equivalence naturally obtained through the final semantics coincides with the following abstract notion of bisimilarity, based on spans of coalgebra homomorphisms.

Definition 7.2 An F-bisimulation between two F-coalgebras (X, α) and (Y, β) is an F-coalgebra (R, ρ) together with two morphisms $f : R \to X$ and $g : R \to Y$ making the following diagram commute:

$$\begin{array}{ccccc} X & \xleftarrow{f} & R & \xrightarrow{g} & Y \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \rho} & & \downarrow{\scriptstyle \beta} \\ FX & \xleftarrow{Ff} & FR & \xrightarrow{Fg} & FY \end{array}$$

(Aczel and Mendler restrict to spans that actually correspond to set-theoretical binary relations, but this is not crucial.) This notion actually maps in most cases to the standard and concrete notions defined for the corresponding systems: R should be thought as a relation between X and Y, and the conditions on R amount to saying that it is a bisimulation. (There are other candidates for abstract definitions of bisimilarity, e.g., [HJ98], see [Sta11] for a comprehensive analysis of their relationships.)

Lenisa [Len99] and Bartels [Bar03] have studied enhancements in this abstract setting. For instance, they deal with the case where the considered coalgebra carries some additional structure (e.g., those are coalgebras of terms, like in process algebra, or coalgebras of sets, like with determinised automata). This is done abstractly using a monad T to represent this structure: a coalgebra with structure T is a coalgebra with carrier TX for some object X, and an F-bisimulation up to T between two F-coalgebras (TX, α) and (TY, β) is an FT-coalgebra (R, ρ) together with two morphisms $f : R \to TX$ and $g : R \to TY$ making the following diagram commute:

$$\begin{array}{ccccc} TX & \xleftarrow{f} & R & \xrightarrow{g} & TY \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \rho} & & \downarrow{\scriptstyle \beta} \\ FTX & \xleftarrow{Ff^{\#}} & FTR & \xrightarrow{Fg^{\#}} & FTY \end{array}$$

(Where $f^{\#} : TR \to TX$ is obtained from $Tf : TR \to TTX$ using the multiplication of the monad T, and similarly for $g^{\#}$.) Intuitively, in the case where T is the term monad for a process algebra, the above R is a relation between processes, and the diagram asserts that R is a bisimulation up to context.

Like in the concrete case, one needs to impose conditions for such a technique to be sound; here again it suffices for instance that there exists a distributive law of T over F [Bar03]. As mentioned above, this condition resembles the notion of compatibility in complete lattices ($f \circ b \subseteq b \circ f$—Sect. 5); in fact, it can be shown that when we have such a distributive law, then the function f corresponding to T for up to 'T-context' is indeed compatible with the function b naturally associated to F [BPPR14].

7.3. Friends: enhanced corecursion in Isabelle/HOL

Enhanced corecursion schemes have been implemented recently by Popescu, Blanchette, Traytel et al. [BPT15, BBL+17, BBB+17] in the proof assistant Isabelle/HOL. This makes it possible to work efficiently with many coinductive datatypes, be it to define functions on those datatypes, or to reason about them.

In this line of work, they focus on *bounded natural functors*, which always admit final coalgebra; they generically provide enhanced corecursion schemes using socalled *friends*: functions which are causal and actually correspond to distributive laws, as described in Sect. 7.1. They moreover automatically derive up-to-context techniques, along the lines of Sect. 7.2, to ease proofs of equations on coinductive objects.

8. Conclusions

In this paper we have reviewed the history of the developments of enhancements of the bisimulation proof method and more generally of the coinduction proof method. Those discussed are the origins of the main developments that are known to the authors by the time of writing of this paper (August 2019). However the topic is still very active, and we expect further developments will occur in the years to come.

For instance, we have mentioned that there is currently a lot of work on languages with probabilities or metrics, while theories of enhancements for these languages are still in their early stages. New useful forms of enhancements might be discovered, as well as better ways of transplanting known enhancements from ordinary transition systems to the new settings.

Advances would also be welcome in the area of higher-order languages. As mentioned in Sect. 4.1, the bisimulation enhancements are particularly effective in these languages, yet some basic forms of 'up-to', such as up-to-context, are still poorly understood. An example is the relationship between the substitutivity or congruence properties of bisimilarity with up-to-context. For basic forms of bisimilarity and basic languages, such as applicative bisimilarity and pure call-by-name or call-by-value λ-calculus [Abr90], the proof techniques for congruence of bisimilarity are well established. However the soundness of the corresponding up-to-context techniques remains an open problem.

Another example of an open long-standing problem concerns the asynchronous π-calculus [HT91, Bou92]—widely used as a model for distributed systems. In this calculus, in absence of operators for testing the identity of names, bisimilarity is preserved by name substitutions [San00, BS98b], yet it is unknown whether name substitutions could be used as an up-to technique (such a technique would be defined in the same manner as the up-to injective substitutions in Sect. 4.2 but without the limitation that the substitutions should be injective). The problem is relevant also for other forms of asynchronous calculi, e.g., in the CCS or Higher-Order π-calculus style [San01].

Acknowledgements

We would like to thank the referees for many useful comments. Sangiorgi acknowledges support from the MIUR-PRIN project 'Analysis of Program Analyses' (ASPRA, ID 201784YSZ5_004) and the H2020-MSCA-RISE project ID 778233 "Behavioural Application Program Interfaces (BEHAPI)". Pous was supported by the European Research Council (ERC) under the European Union's Horizon 2020 programme (CoVeCe, grant agreement No 678157).

References

[ABLP17] Aristizábal A, Biernacki D, Lenglet S, Polesiuk P (2017) Environmental bisimulations for delimited-control operators with dynamic prompt generation. Log Methods Comput Sci 13(3)

[Abr90] Abramsky S (1990) The lazy lambda calculus. In: Turner DA (ed) Research topics in functional programming. Addison Wesley, Boston, pp 65–116

[AG98] Abadi M, Gordon AD (1998) A bisimulation method for cryptographic protocols. In: Hankin C (ed) Proceedings of the ESOP'98, volume 1381 of LNCS. Springer, Berlin, pp 12–26

[AH91] Arun-Kumar S, Hennessy M (1991) An efficiency preorder for processes. In: Proceedings of the TACS '91, volume 526 of Lecture notes in computer science. Springer, Berlin, pp 152–175

[AM89] Aczel P, Mendler NP (1989) A final coalgebra theorem. In: Proceedings of the category theory and computer science, volume 389 of LNCS. Springer, Belrin, pp 357–365

[Bar03] Bartels F (2003) Generalised coinduction. Math Struct Comput. Sci. 13(2):321–348

[Bar04] Bartels F (April 2004) On generalised coinduction and probabilistic specification formats. PhD thesis, CWI, Amsterdam

[BBB+17] Biendarra J, Blanchette JC, Bouzy A, Desharnais M, Fleury M, Hölzl J, Kuncar O, Lochbihler A, Meier F, Panny L, Popescu A, Sternagel C, Thiemann R, Traytel D (2017) Foundational (co)datatypes and (co)recursion for higher-order logic. In FroCoS, volume 10483 of LNCS. Springer, Belrin, pp 3–21

[BBG98] Bravetti M, Bernardo M, Gorrieri R (1998) A note on the congruence proof for recursion in markovian bisimulation equivalence. In: Priami C (ed) Proceedings of the 6th internation workshop on process algebras and performance modeling (PAPM '98), pp 153–164

[BBL+17] Blanchette JC, Bouzy A, Lochbihler A, Popescu A, Traytel D (2017) Friends with benefits—implementing corecursion in foundational proof assistants. In: ESOP, volume 10201 of LNCS. Springer, Berlin, pp 111–140

[BDP99] Boreale M, De Nicola R, Pugliese R (1999) Basic observables for processes. Inf Comput 149(1):77–98
[BKK17] Bonchi F, König B, Küpper Sebastian (2017) Up-to techniques for weighted systems. In: TACAS, volume 10205 of LNCS. Springer, Berlin, pp 535–552
[BKP18] Bonchi F, König B, Petrisan D (2018) Up-to techniques for behavioural metrics via fibrations. In: CONCUR, volume 118 of LIPIcs, Schloss Dagstuhl, pp 17:1–17:17
[BKvO98] Bezem M, Klop JW, van Oostrom V (1998) Diagram techniques for confluence. Inf Comput 141(2):172–204
[Bor19] Boreale M (2019) Algebra, coalgebra, and minimization in polynomial differential equations. Log Methods Comput Sci 15(1)
[Bou92] Boudol G (1992) Asynchrony and the π-calculus. Technical Report RR-1702, INRIA-Sophia Antipolis
[BP13] Bonchi F, Pous D (2013) Checking NFA equivalence with bisimulations up to congruence. In: Proceedings of the POPL, ACM, pp 457–468
[BP15] Bonchi F, Pous D (2015) Hacking nondeterminism with induction and coinduction. Commun ACM 58(2):87–95
[BPPR14] Bonchi F, Petrişan D, Pous D, Rot J (2014) Coinduction up-to in a fibrational setting. In: Proceeding of the CSL-LICS, ACM, pp 20:1–20:9
[BPR17] Basold H, Pous D, Rot J (2017) Monoidal company for accessible functors. In: Proceedings of the CALCO, volume 72 of LIPIcs. Schloss Dagstuhl
[BPT15] Blanchette JC, Popescu A, Traytel D (2015) Foundational extensible corecursion: a proof assistant perspective. In: ICFP, ACM, pp 192–204
[BS98a] Boreale M, Sangiorgi D (1998) Bisimulation in name-passing calculi without matching. In: LICS, IEEE, pp 165–175
[BS98b] Boreale M, Sangiorgi D (1998) Some congruence properties for pi-calculus bisimilarities. Theor Comput Sci 198(1–2):159–176
[Cau90] Caucal D (1990) Graphes canoniques de graphes algébriques. ITA 24:339–352
[CG98] Cardelli L, Gordon AD (1998) Mobile ambients. In: Nivat M (ed) Proceedings of the FoSSaCS '98, volume 1378 of LNCS. Springer, Berlin, pp 140–155
[CHS95] Christensen S, Hüttel H, Stirling C (1995) Bisimulation equivalence is decidable for all context-free processes. Inf Comput 121(2):143–148
[CPV16] Chatzikokolakis K, Palamidessi C, Vignudelli V (2016) Up-to techniques for generalized bisimulation metrics. In: Desharnais J, Jagadeesan R (eds) Proceedings of the CONCUR 2016, volume 59 of LIPIcs, Schloss Dagstuhl, pp 35:1–35:14
[Dan18] Danielsson NA (2018) Up-to techniques using sized types. PACMPL 2(POPL):43:1–43:28
[DHS17] Durier A, Hirschkoff D, Sangiorgi D (2017) Divergence and unique solution of equations. In: Meyer R, Nestmann U (eds) 28th International conference on concurrency theory, CONCUR 2017, volume 85 of LIPIcs, Schloss Dagstuhl, pp 11:1–11:16
[FG05] Fournet C, Gonthier G (2005) A hierarchy of equivalences for asynchronous calculi. J Logic Algebr Program 63(1):131–173
[FM91] Fernandez J-C, Mounier L (1991) "On the Fly" verification of behavioural equivalences and preorders. In: CAV, volume 575 of LNCS. Springer, Belin, pp 181–191
[Gav19] Gavazzo F (2019) Coinductive equivalences and metrics for higher-order languages with algebraic effects. PhD thesis, Univ. Bologna
[Hir99] Hirschkoff D (1999) Mise en oeuvre de preuves de bisimulation. PhD thesis, Ecole Nationale des Ponts et Chaussées
[HJ98] Hermida C, Jacobs B (1998) Structural induction and coinduction in a fibrational setting. Inf Comput 145(2):107–152
[HJM96a] Hirshfeld Y, Jerrum M, Moller F (1996) A polynomial algorithm for deciding bisimilarity of normed context-free processes. Theor Comput Sci 158(1&2):143–159
[HJM96b] Hirshfeld Y, Jerrum M, Moller F (1996) A polynomial-time algorithm for deciding bisimulation equivalence of normed basic parallel processes. Math Struct Comput Sci 6(3):251–259
[HK71] Hopcroft JE, Karp RM (1971) A linear algorithm for testing equivalence of finite automata. Technical Report 114, Cornell Univ., December
[HM85] Hennessy M, Milner R (1985) Algebraic laws for nondeterminism and concurrency. J ACM 32:137–161
[HNDV13] Hur C-K, Neis G, Dreyer D, Vafeiadis V (2013) The power of parameterization in coinductive proof. In: POPL, ACM, pp 193–206
[HT91] Honda K, Tokoro M (1991) A small calculus for concurrent objects. In: Proceedings of the workshop on object-based concurrent programming, OOPSLA/ECOOP '90, ACM, New York, NY, USA, pp 50–54
[Jac06] Jacobs B (2006) Distributive laws for the coinductive solution of recursive equations. Inf Comput 204(4):561–587
[Jac16] Jacobs B (2016) Introduction to coalgebra: towards mathematics of states and observation, volume 59 of Cambridge tracts in theoretical computer science. Cambridge University Press, Cambridge
[JPR09] Jagadeesan R, Pitcher C, Riely J (2009) Open bisimulation for aspects. Trans Asp Oriented Softw Dev 5:72–132
[Kli11] Klin B (2011) Bialgebras for structural operational semantics: an introduction. Theor Comput Sci 412(38):5043–5069
[Kna28] Knaster B (1928) Un théorème sur les fonctions d'ensembles. Annales de la Société Polonaise de Mathématiques 6:133–134
[KW06] Koutavas V, Wand M (2006) Small bisimulations for reasoning about higher-order imperative programs. In: Proceedings of the 33rd ACM SIGPLAN-SIGACT symposium on principles of programming languages, pp 141–152
[Las98a] Lassen SB (1998) Relational reasoning about contexts. In: Gordon AD, Pitts AM (eds) Higher order operational techniques in semantics. Cambridge University Press, Cambridge
[Las98b] Lassen SB (1998) Relational reasoning about functions and nondeterminism. PhD thesis, Department of Computer Science, University of Aarhus
[Las99] Lassen SB (1999) Bisimulation in untyped lambda calculus: Böhm trees and bisimulation up to context. Electr Notes Theor Comput Sci 20:346–374
[Len99] Lenisa M (1999) From set-theoretic coinduction to coalgebraic coinduction: some results, some problems. Electr Notes Comput Sci 19:2–22
[LG19] Dal Lago U, Gavazzo F (2019) Effectful normal form bisimulation. In: ESOP '19, volume 11423 of LNCS. Springer, Berin, pp 263–292
[LPW00] Lenisa M, Power J, Watanabe H (2000) Distributivity for endofunctors, pointed and co-pointed endofunctors, monads and comonads. Electr Notes Comput Sci 33:230–260

[Mil80] Milner R (1980) A calculus of communicating systems, volume 92 of LNCS. Springer, Berlin
[Mil83] Milner R (1983) Calculi for synchrony and asynchrony. Theor Comput Sci 25:269–310
[Mil87] Milner R (1987) Operational and algebraic semantics of concurrent processes. Notes, November 1987. Appeared as Tech Rep ECS-LFCS-88-46, Edinburgh 1988, and later as a chapter in Handbook of Theoretical Computer Science (vol. B), pp 1201–1242, MIT Press, 1990
[Mil89] Milner R (1989) Communication and concurrency. Prentice Hall, Englewood Cliffs
[MMS13] Milius S, Moss LS, Schwencke D (2013) Abstract GSOS rules and a modular treatment of recursive definitions. Log Methods Comput Sci 9(3)
[MN05] Merro M, Nardelli FZ (2005) Behavioural theory for mobile ambients. J ACM 52(6):961–1023
[MPW89] Milner R, Parrow J, Walker D (1992) A calculus of mobile processes, Part I and II. Technical report ECS-LFCS-89-85 and -86, University of Edinburgh, 1989. Appeared in J. Inf. Comp. 100:1–77
[MT91] Milner R, Tofte M (1988) Co-induction in relational semantics. Theor Comput Sci 87:209–220, 1991. Also Tech. Rep. ECS-LFCS-88-65, University of Edinburgh
[New42] Newman MHA (1942) On theories with a combinatorial definition of "equivalence". Ann Math 43(2):223–243
[NR11] Niqui M, Rutten J (2011) A proof of Moessner's theorem by coinduction. Higher Order Symb Comput 24(3):191–206
[Par81] Park D (1981) A new equivalence notion for communicating systems. In: Maurer G (ed) Bulletin EATCS, volume 14, pages 78–80, 1981. Abstract of the talk presented at the Second Workshop on the Semantics of Programming Languages, Bad Honnef, March 16–20 1981. Abstracts collected in the Bulletin by B. Mayoh
[Par87] Parrow J (1987) Notes 'jp3' on label passing. Handwritten notes
[Pit95] Pitts AM (1995) An extension of Howe's construction to yield simulation-up-to-context results. Unpublished manuscript
[Plo04a] Plotkin GD (2004) The origins of structural operational semantics. J Logic Algebr Program 60–61:3–15
[Plo04b] Plotkin GD (2004) A structural approach to operational semantics. J Logic Algebr Program 60-61:17–139
[Pou05] Pous D (2005) Up-to techniques for weak bisimulation. In: Proceedings of the ICALP, volume 3580 of LNCS. Springer, Berlin, pp 730–741
[Pou06] Pous D (2006) Weak bisimulation up to elaboration. In: Proceedings of the CONCUR, volume 4137 of LNCS. Springer, Berlin, pp 390–405
[Pou07a] Pous D (2007) Complete lattices and up-to techniques. In: Proceedings of the APLAS '07, volume 4807 of LNCS, pages 351–366. Springer, Belrin
[Pou07b] Pous D (2007) New up-to techniques for weak bisimulation. Theor Comput Sci 380(1–2):164–180
[Pou08] Pous D (2008) Using bisimulation proof techniques for the analysis of distributed algorithms. Theor Comput Sci 402(2–3):199–220
[Pou16] Pous D (2016) Coinduction all the way up. In: Proceeding of the LICS, ACM, pp 307–316
[PR17] Pous D, Rot J (2017) Companions, codensity, and causality. In: Proceedings of the FoSSaCS, volume 10203 of LNCS. Springer, Berlin, pp 106–123
[PS97] Pierce B, Sangiorgi D (2000) Behavioral equivalence in the polymorphic pi-calculus. In: Proceedings of the 24th POPL. ACM Press, 1997. Full paper in JACM 47(3)
[PS12] Pous D, Sangiorgi D Enhancements of the bisimulation proof method. In: Sangiorgi and Rutten [SR12].
[PW16] Parrow J, Weber T (2016) The largest respectful function. Log Methods Comput Sci 12(2)
[Rut00] Rutten J (2000) Universal coalgebra: a theory of systems. Theor Comput Sci 249(1):3–80
[Rut05] Rutten J (2005) A coinductive calculus of streams. Math Struct Comput Sci 15(1):93–147
[San92] Sangiorgi D (1992) Expressing mobility in process algebras: first-order and higher-order paradigms. PhD thesis CST–99–93, Department of Computer Science, University of Edinburgh
[San94] Sangiorgi D (1996) Locality and true-concurrency in calculi for mobile processes. In: TACS'94, volume 789 of LNCS, pages 405–424. Springer Verlag, 1994. Full version in TCS, vol 155, 39–83
[San95] Sangiorgi D (1998) On the bisimulation proof method. In: Wiedermann J, Háiek P (eds) Proceedings of the MFCS'95, volume 969 of LNCS, pp 479–488. Springer, Berlin 1995. Full version in J. MSCS, vol 8, pp 447–479
[San98] Sands D (1998) Improvement theory and its applications. In: Gordon AD, Pitts AM (eds) Higher order operational techniques in semantics, publications of the Newton Institute, Cambridge University Press, pp 275–306.
[San00] Sangiorgi D (2000) Lazy functions and mobile processes. In: Plotkin G, Stirling C, Tofte M (eds) Proof, language and interaction: essays in honour of Robin Milner. MIT Press, Cambridge
[San01] Sangiorgi D (2001) Asynchronous process calculi: the first- and higher-order paradigms. Theor Comput Sci 253(2):311–350
[San09] Sangiorgi D (2009) On the origins of bisimulation and coinduction. ACM Trans Program Lang Syst 31(4):15
[San12] Sangiorgi D (2012) Introduction to bisimulation and coinduction. Cambridge University Press, Cambridge
[San15] Sangiorgi D (2015) Equations, contractions, and unique solutions. In: Rajamani SK, Walker D (eds) POPL 2015, ACM, pp 421–432
[SBBR10] Silva A, Bonchi F, Bonsangue M, Rutten J (2010) Generalizing the powerset construction, coalgebraically. In: FSTTCS, LIPIcs, pp 272–283. Schloss Dagstuhl
[SKS07] Sangiorgi D, Kobayashi N, Sumii E (2007) Environmental bisimulations for higher-order languages. In: Proceedings of the 22nd IEEE symposium on logic in computer science (LICS 2007), pp 293–302. IEEE Computer Society
[SM92] Sangiorgi D, Milner R (1992) The problem of "weak bisimulation up to". In: Proceedings of the 3rd CONCUR, volume 630 of LNCS. Springer, Berlin, pp 32–46
[SP04] Sumii E, Pierce BC (2004) A bisimulation for dynamic sealing. In: Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on principles of programming languages, pp 161–172
[SP05] Sumii E, Pierce BC (2005) A bisimulation for type abstraction and recursion. In: Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on principles of programming languages, pp 63–74
[SR12] Sangiorgi D, Rutten J (eds) (2012) Advanced topics in bisimulation and coinduction. Cambridge University Press, Cambridge
[Sta11] Staton S (2011) Relating coalgebraic notions of bisimulation. Log Methods Comput Sci 7(1)

[SV16] Sangiorgi D, Vignudelli V (2016) Environmental bisimulations for probabilistic higher-order languages. In: Bodík R, Majumdar R (eds) Proceedings of the POPL 2016, ACM, pp 595–607
[Tar55] Tarski A (1955) A lattice-theoretical fixpoint theorem and its applications. Pac J Math 5(2):285–309
[Tar75] Tarjan RE (1975) Efficiency of a good but not linear set union algorithm. J ACM 22(2):215–225
[Tho89] Thomsen B (1989) A calculus of higher order communicating systems. In: POPL'89, ACM, pp 143–154
[TP97] Turi D, Plotkin GD (1997) Towards a mathematical operational semantics. In: LICS, IEEE, pp 280–291
[UVP01] Uustalu T, Vene V, Pardo A (2001) Recursion schemes from comonads. Nord J Comput 8(3):366–390
[Wal87] Walker DJ (1987) Bisimulation and divergence in CCS. Tech report, LFCS, Dept of Comp Sci, Edinburgh Univ

Received 5 May 2019
Accepted in revised form 29 September 2019 by Cliff Jones and Jose N. Oliveira
Published online 8 November 2019