REASONING ON CIRCULAR PROOFS FOR PROGRAMMING

Partner	Name	Firstname	Current position	Role & resp.	Months
IRIF	Saurin	Alexis	CR CNRS	PI, Paris coord. WP 1–6	32
IRIF	Ehrhrard	Thomas	DR CNRS	WP 1	7,2
IRIF	Guatto	Adrien	MCF, Univ. Paris	WP 1, 2 & 5	9,6
IRIF	Herbelin	Hugo	DR INRIA	WP 3 & 6	7,2
IRIF	Hyvernat	Pierre	MCF, U. Savoie Mt-Blanc	WP 2,4,5 & 6	24
IRIF	Melliès	Paul-André	DR CNRS	WP 1,2,3 & 5	14,4
IRIF	Petrişan	Daniela	MCF, Univ. Paris	WP 1 & 4	7,2
IRIF	Valiron	Benoit	MCF, CentraleSupélec	WP 1 & 3	9,6
LS2N	Jaber	Guilhem	MCF, Univ. Nantes	Local coord. Nantes WP 2,3,5 & 6	15,8
LS2N	Munch-Maccagnoni	Guillaume	CR INRIA	WP 1, 3	4,8
LS2N	Pédrot	Pierre-Marie	CR INRIA	WP 6	9,6
LS2N	Sozeau	Matthieu	CR INRIA	WP 6	7,2
LIP	Kuperberg	Denis	CR CNRS	Local coord. Lyon WP 2,3,5	24
LIP	Pous	Damien	DR CNRS	WP 3,4,6	9,6
LIP	Clairambault	Pierre	CR CNRS	WP 1,2	4,8
LIP	Laurent	Olivier	DR CNRS	WP 4, 6	4,8
LIP	Mio	Matteo	CR CNRS	WP 4	4,8
LIP	Riba	Colin	MCF, ENS Lyon	WP 1,5	4,8
LIP	Zakowski	Yannick	CR INRIA	WP 5,6	7,2
LIS	Santocanale	Luigi	Professor, Aix-Marseille Univ. (LIS)	Local coord. Marseille WP 1,3,4,5 & 6	19,2
LIS	Grellois	Charles	MCF, Aix-Mars. Univ. (LIS)	WP 1,2 & 3	9,6
LIS	Vaux	Lionel	MCF, Aix-Mars. Univ. (I2M)	WP 1	9,6
			PhD Students & Post-docs		
IRIF	Chardonnet	Kostia	PhD Student	WP 1& 3	12
IRIF	De	Abhishek	PhD Student	WP 2& 5	6
IRIF	Jafarrahmani	Farzad	PhD Student	WP 1 & 3	12
LS2N	Maillard	Kenji	Postdoc	WP 3 & 6	9.6
LIP	Hazard	Emile	PhD Student	WP 5	24
			ANR funded participants		
IRIF	Postdoc A	-	Postdoc Paris	WP 2 & 3	12
LIP	Postdoc B	-	Postdoc Lyon	WP 5	12
LIS	PhD Stud.	-	PhD student	WP 4, 5 & 6	36
LS2N	Postdoc C	-	Postdoc Nantes	WP 6	12

Contents

1 Proposal's context, positioning and objectives

1.1 Objectives and research hypotheses

On the unusual effectiveness of (co)induction in computer science. Given the pervasive nature of inductively defined data in computing (integers, finite lists, finite trees, *etc.*), induction is one of the most elementary concepts of computer science and induction principles are probably its most important reasoning principles.

Coinduction can arguably be considered of the same basic nature as induction, even though it is not as widespread and well-known as induction. As a dual concept to induction, coinduction allows to model and reason about infinite data and infinite behaviours (finite or infinite lists, infinite words, streams, trees or execution traces, bisimulations, *etc.*) and provides its own proof principles. Still, the focus on coinduction is much more recent and CS community is not yet as familiar with coinduction than it is with induction. Reading 20 years later the inspiring paper by Halpern *et al. On the unusual effectiveness of logic in computer science* [45], it is striking to note that, while induction is – explicitly or implicitly – a key concept of almost every section of this paper which emphasizes the central role of logic for computer science, one cannot find a single reference to coinduction.

Coinduction and programming. While the ability to define inductive and coinductive types gives the user a great flexibility in specifying data together with a high level of abstraction, coinductive types are in general not as well-handled as their inductive counterpart, notably in total programming languages and proof assistants. This can be explained on the base of younger age of coinduction compared to that of induction. We believe, however, that this is most importantly due to the sharp discrepancy between the infinite nature of coinductive objects and the termination requirements of some of those languages.

If we come to mixed inductive and coinductive types, then very few languages correctly handle them, see e.g. [43, 26, 25], and this is an active research topic for proof assistants [1, 14, 15, 27, 13, 47].

Coinductive types in Coq. The Coq proof assistant [83] allows us to define coinductive types using cofixpoints [44]. Such coinductive reasoning is widely used in large scale formalized developments, such as CompCert [63], which relies on (bi)simulation arguments to express the semantic preservation of several compilation phases. Yet, both the foundational and practical aspects of coinduction in type theory are still unsatisfactory.

On a foundational side, critical meta-theoretical problems arise because of coinductive predicates: coinduction in Coq breaks type preservation [65]. Moreover, the interaction between inductive and coinductive types suffers severe limitations. From the point of view of usability, coinduction is also largely ill-handled, particularly when compared to induction (e.g.Coq does not provide any coinduction principle). Moreover, to preserve the strong normalization property, a corecursive definition must fulfill a syntactic criterion of productivity: every corecursive call must appear as an argument of a constructor of the coinductive type, this is the *guard condition*, constraining the writing of coinductive definitions. In the example of Figure 1, `filter1everyk` and `faultyfilter` are rejected, only `gfilter1everyk` satisfies the guard condition. Even if the last example is rightly rejected (the term `faultyfilter (fun _ ⇒ false)` cannot be applied to an infinite stream without diverging), the programmer should be able to add an extra logical argument to `faultyfilter` to justify, for instance, that there is only a finite number of steps before the predicate `cond` is verified by an element of the stream. Such a logical argument can be expressed in Coq by nesting an induction inside the coinduction [30, 12] which requires a significant reformulation of the program. More recent work [47] introduces a semantically justified productivity argument based on lattice theory formalized inside Coq itself, resulting in the Paco library. Still, such issues should instead be solved at the language level.

Fixed point logics and circular proofs. (Co)induction proof principles are derived from fixed-point theorems, that ensure the existence of least and greatest fixed points. Witnessing the importance of those forms of reasoning, logics featuring least and greatest fixed points have been developed since the early eighties [58]. These logics are named *μ-calculi*, and were very successful as specification languages, e.g. to specify temporal or spatial properties of program executions. More recently, the structural proof theory of those logics with least and greatest fixed point received a growing attention from the community especially with the research line on circular proof theory. Unsurprisingly, such proof systems can themselves be derived from Knaster-Tarski's characterization of least fixed points as least pre-fixed points (and dually for gfp). However, when building proofs using such inference rules, one has to guess (co)invariants which may be more complex than the property to establish.

```
CoInductive Stream := Cons : nat → Stream → Stream.

CoFixpoint filter1everyk (m n : nat) (s : Stream) : Stream := match (m,s) with
  | (0, Cons a s') ⇒ Cons a (filter1everyk n n s')
  | (S m', Cons a s') ⇒ (filter1everyk m' n s') end.
Fixpoint afterk (n : nat) (s: Stream) : nat * Stream := match (n,s) with
  | (0, Cons a s') ⇒ (a, s')
  | (S m, Cons a s') ⇒ afterk m s' end.
CoFixpoint gfilter1everyk (n : nat) (s : Stream) : Stream := match (afterk n s) with
| (a, s') ⇒ Cons a (gfilter1everyk n s') end.
CoFixpoint faultyfilter (cond : nat → bool) (s : Stream) : Stream := match s with
| Cons a s' ⇒ if cond a then Cons a (faultyfilter cond s') else faultyfilter cond s' end.
```

Figure 1: Guarded and non-guarded examples of coinductive predicates.

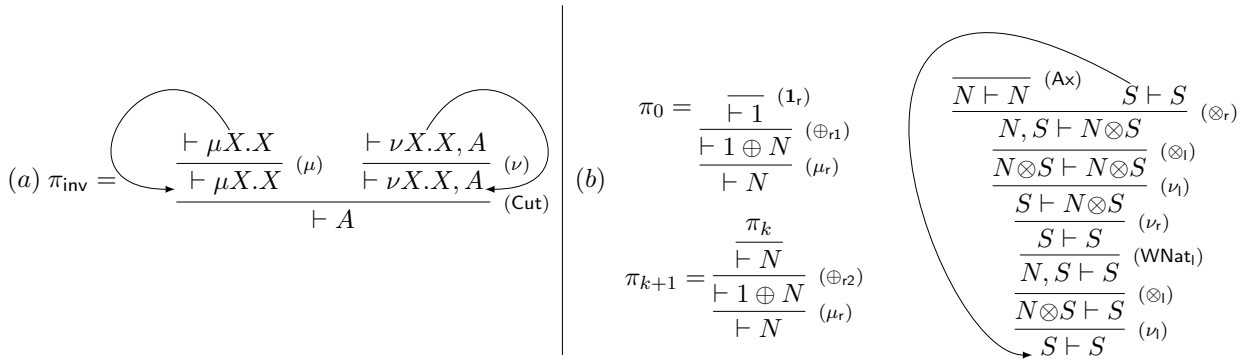

Figure 2: (a) Example of a non-valid circular derivation, illustrating inconsistency of non-valid derivations. (b) Examples of μMALL finite and circular derivations of sequents $\vdash N$ and $S \vdash S$.

Infinite (aka. non-wellfounded) and circular proofs were introduced as alternatives to these explicit forms of (co)induction rules. Considering infinite (or *regular* in the case of circular proofs) derivation trees makes it possible to reason on fixed points by simply unfolding them, possibly infinitely often. There is of course a price to pay: the soundness of such systems requires a *validity* condition which is global: it ensures that some progress is made along *every infinite branch*. Indeed, in the particular case of circular proofs, one can view a circular proof as reducing a statement to be established to the statement itself, under the condition that the structure of the loop prevents "vicious" circular reasoning: circular *invalid* derivations may conclude any formula as shown in Figure 2.(a). For instance, in the framework of linear logic with fixed points, one can consider formulas $N = \mu X.1 \oplus X$ and $S = \nu Y.N \otimes Y$ which are respectively the least and greatest fixed points of operators $X \mapsto 1 \oplus X$ and $Y \mapsto N \otimes Y$. These formulas provide encodings for the (co)inductive types of the natural numbers and streams of natural numbers in μMALL and one can consider several examples of circular proofs on logical judgements (or sequents) involving those formulas in Figure 2.(b).

Towards a Curry-Howard correspondence between circular proofs and recursive programs. Circular proofs [16, 19, 20, 77] were successful both at a foundational and at a more applied level. Foundationally, the meta-theory of circular proofs is important and well-behaved compared to finitary proof systems for fixed point logics: cut-elimination is notably better-behaved in non-wellfounded proof systems [77, 42, 9] than with finitary proofs [7]. From a more practical point of view, circular proofs happened to be particularly useful for automated proof construction in fixed point logics, with automated theorem provers now relying on circular proofs [18, 41]. A natural question arises, underlying our project:

Can circular proofs have a similar impact on programming with coinductive types?

In the long-standing tradition of the Curry-Howard correspondence [46] associating logical formulas to data types, logical proofs to programs and cut-elimination (or proof normalization) to program execution, it is indeed natural to expect that the success of circular proofs can have an impact on programming with inductive and

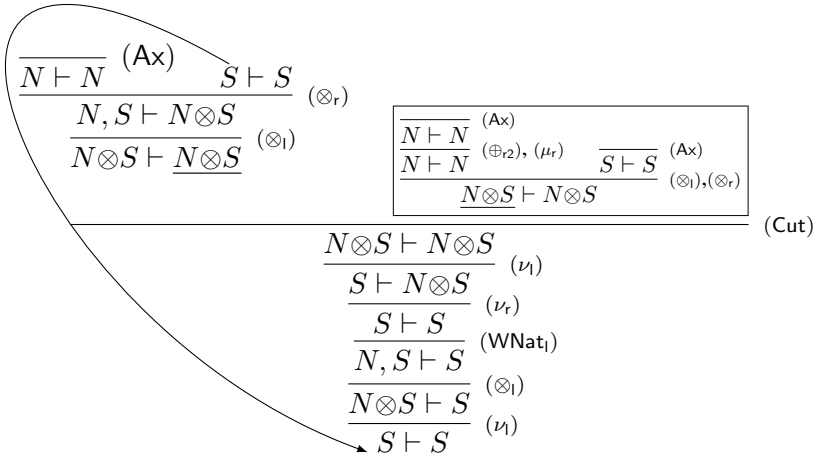

Figure 3: Examplifying the lack of compositionality of circular proofs.

coinductive types. Still, in the current state of the art, circular proofs suffer a lack of compositionality preventing them to be directly transferable, in the Curry-Howard perspective, to the programming side.

ReCiProg proposes to improve the compositionality of circular proofs and their meta-theory (providing them, notably, with well-established proof-semantics) to unlock their use in a Curry-Howard correspondence with inductive and coinductive types. We put a specific focus on the Coq proof assistant with the aim at improving its coinductive types and its support for coinductive reasoning.

In the course of our **ReCiProg** project, we consider the Coq proof assistant as a central object of research as it will be both a particular software to which our findings will find a concrete application, and a tool to formalize our results as well as to test the ability to formalize coinductive reasoning. This twofold role of Coq is best described in WP 6 below.

Lack of compositionality. The lack of compositionality of circular proofs can be described in two ways: on the one hand, those proof objects shall be equipped with a global validity condition and on the other hand, this validity condition is only compositional in a weak sense that we describe now.

While cut-elimination holds for circular proofs of linear logic, cuts – when introduced in cycles along an infinite branch – may badly interact with the validity condition and result in invalidating a proof and in the failure of cut-elimination: this is examplified in Figure 3. This phenomenon can be produced on even simpler situations, for instance applying an identity program (that is building a trivial cut with an identity axiom) to a recursive call (that is along an cycle) may render it invalid.

Indeed, the validity condition requires every infinite branch to be inhabited by a thread which witnesses that an inductive object is consumed or a coinductive object is produced. This is a strong restriction, and rules out some natural proofs that could be considered correct on other grounds: they derive valid judgements, cut-elimination produces a valid proof, etc.

A condition for using circular proofs as a primitive construct for programming and for building coinductive reasonings in proof assistants such as Coq is to provide a solution to their lack of compositionality.

Lack of proof semantics. While the μ-calculus [58] has a well-established truth semantics, the denotational semantics of logics with least and greatest fixed points is yet underdeveloped, especially when it comes to non-wellfounded and circular proofs. Regarding finitary proofs for linear or intuitionistic logics with least and greatest fixed points, one can essentially build on the following prior works: originally, Santocanale's categorical [77] and game-theoretical[76, 78] semantics, Clairambault's game semantics for μLJ[24], interactive semantics of $\mu MALL$ by Baelde et al. in the framework of Ludics [10] as well as Ehrhard and Jafarramahni's recent work on a coherent semantics of μLL. In a slightly different though related perspective, there are also a bunch of categorical semantics developed on the emerging topic of the Curry-Howard correspondence for functional reactive programming where reactive types are in correspondence with LTL formulas [22, 52, 53, 54, 55, 56].

When turning to circular proofs, even less is known: Santocanale defined a categorical semantics for circular proofs of the additive fragment of linear logic with fixed points [77] later extended with Fortier to properly treat

the cut-rule [42] but no denotational semantics is yet known for the full fragment of μLL non-wellfounded (or even circular) proofs nor for other logics.

Expanding the realm of circular proofs. After the first seminal logical frameworks expressing inductive and coinductive statements were equipped with circular proofs, namely Martin Löf's inductive predicates [17, 19, 21] and variant of the μ-calculus (linear-time [33, 38, 36] or linear logical [77, 42, 37, 9]), circular proofs found to be successfully applied on a variety of other logical languages such as separation logic, Kleene algebra or transitive closure logics [72, 31, 28].

A global objective will be to investigate applicability and benefits of circular proofs to various frameworks, from logics with quantitative semantics (see below) to λ-calculi with dependent types and Coq.

Let us illustrate this question on an example from fixed–point logics based on *quantitative* semantics – where formulas are interpreted as real numbers rather than booleans. Those logics indeed emerged as useful specification formalisms to express properties of programs exhibiting quantitative (e.g., probabilistic) behaviours (see, e.g., [68, 67, 6]). One interesting line of research is to advance the theory of circular proofs in the context of quantitative semantics. For a simple example, consider the following least fixed–point equation $E \overset{\mu}{=} (\frac{1}{2} \cdot E) + 0$ evaluated in $\mathbb{R}$. The least (and in this case only) solution is $E = 0$. Suppose we attempt a bottom-up proof-search for $0 \leq E$, shown on Figure 4. As usual, in the machinery of circular proofs, we have reduced the problem of proving $0 \leq E$ to itself, and we would like to conclude by observing that the "reasoning loop" is valid. However this type of loop would be deemed not valid in the usual framework of circular proofs, technically because the expression being unfolded in the cycle is a least fixed–point expression and it appears on the right side of the inequality, violating the usual thread condition. The loop is indeed valid but the justification for its validity comes from the real–valued semantics under consideration and the Archimedean property of $\mathbb{R}$: appropriate conditions for circular reasoning in such a quantitative framework shall be established.

$$\dfrac{\dfrac{\dfrac{0 \leq E}{0 \leq (\frac{1}{2} \cdot E)}\ (*)\ \text{if } 0 \leq x \text{ then } 0 \leq \frac{1}{2} \cdot x}{0 \leq (\frac{1}{2} \cdot E) + 0}\ x + 0 = x}{0 \leq E}\ \text{by def. } E = (\frac{1}{2} \cdot E) + 0$$

Figure 4: A proof-search attempt for $0 \leq E$

We believe there is a lot of potential in this approach, on the theoretical side (e.g. understand the link with metric space) and on the practical side (verifying quantitative properties of systems).

Verifying coinductive systems with circular proofs. A further step, once circular proofs have been developed for those more application-oriented logics, will be to attack several research directions more directly connected with verification, investigating more advanced uses of circular proofs in this direction. Indeed, since many natural objects of interest in computer science are defined by coinduction, it is natural to use circular proofs to verify the correctness of these objects, for instance their compliance with a specification. The typical example is the model of infinite words: it is one of the most standard framework for modelling the behaviour of systems, and it is the simplest example of coinduction. Circular proofs can therefore be of use to verify specification on infinite words, such as inclusion of ω-regular languages defined by logics or expressions using coinduction. Given the nature and composition of our consortium:

Research directions connected with verification will be investigated with a great attention to the nature and structure of the proofs being built and in particular to their computational content.

1.2 Position of the project as it relates to the state of the art

The project aims at building on recent developments in structural proof theory which have an already recognized strong potential for application in the design of (typed) programming languages and proof assistants in particular. The international community of researchers in computational logic and theoretical computer science that is

getting interested in circular proofs has been increasing in the past ten years and our consortium reflects the variety of such researchers, from logic and category theory to the theory and implementation of typed programming languages and proof assistants.

The project also builds on the recent ANR JCJC RAPIDO project and ERC CoVeCe which provided major advances in the meta-theory of circular proofs and their use in axiomatizing various logics with induction and coinduction. More specifically, the main advances of RAPIDO directly related with **ReCiProg** are (i) the definition, for a wide range of logics, of a general notion of circular proof with a productive cut-elimination procedure (ii) the investigation of the validity condition from a complexity point of view as well as from the relation with finitary logics (iii) the design of a first proposal for a notion of non-wellfounded proof nets.

The discussion of the state of the art has been mostly done in the presentation of the context. Let us add here some remarks relevant to the positioning of the project community-wise. Since the seminal work by Rutten [73, 74, 51], many scientists working on foundations of computer science converged into a community developing the field of coalgebra theory, see e.g. the scientific meetings CMCS and CALCO. While only part of this community shares our interest in coinduction for programming, the proof theory of fixed point logic, therefore circular proof systems, is presently intensively investigated by some of these researchers, see e.g. [4, 40] to which members of the project are already related. We plan to develop and make formal connections with international groups (in Italy, Japan, the Netherlands, Slovenia, Sweden, UK and USA) which gather the top researchers in those topics and among which are natural researchers to invite for visits during the project. In connection with this, one should mention projects managed by Anupam Das, in the UK, or Bahareh Afshari, in Sweden, who pursue research on circular proofs, with different aims but quite complementary and also targeting to structure the scientific community. More details on this are provided in Section 3.

1.3 Methodology and risk management

In order to achieve the goals presented in Section 1.1, our project is organized around six work-packages (WP) presented in detail below, preceded by a general coordination work package (WP 0):

Summary of the work packages

WP 1: Denotational semantics of fixed-point logics and circular proofs. This will address the lack of proof semantics for circular proofs and design denotational guides for the associated programming languages of the following work packages.

WP 2: Modularity and compositionality of circular proofs. This will address the default of compositionality of circular proofs by designing alternative and more expressive validity conditions as well as alternative proof representations for circular proofs.

WP 3: Circular proofs and Computational Calculi. This work package will consist in conceiving the theoretical grounds for designing programming languages with inductive and coinductive types taking advantage of circular proofs.

WP 4: Circularity and constructivity issues. This will address several directions related with constructivity issues in fixed-point logics: how to turn a circular proof into a finite proof à la Park (that is, address Brotherston-Simpson's conjecture for μ-calculi)? How to formalize circular proofs and their validity and extract validity witnesses? Which proof systems admit a circular proof presentation?

WP 5: Circular Program analysis and verification of infinite systems. This is dedicated to developing program analysis tools as well as verification techniques deriving from circular proof systems.

WP 6: Enhancing coinductive reasoning in Coq. Benefiting from the previous work packages, this aims at improving the treatment of coinductive types in Coq proof assistant in order to ease coinductive reasoning which is currently not satisfactory, both foundationally and practically.

WORK PACKAGE 0: **National and Local Coordination Tasks.** We start by describing an initial WP detailing coordination tasks that we view as a condition of success for the project, due to the fact that our consortium is large and spread over France.

The national and local coordination duties are naturally assigned to the coordinators PI and site coordinators, they will be in regular contact in managing the project.

They will be in charge of organizing the general meetings of the project to be held mainly in Lyon, taking the opportunity of the already existing Chocola monthly seminar, and alternatively in other sites of the project. The actual implementation of those meetings will depend on the conditions of normalization after the Covid crisis: currently, Chocola seminar is held twice a month online and should return to a physical gathering early in the academic year 2021-2022. We plan to organize general meetings twice a year, the day before of after Chocola meetings, plus an annual general meeting in another site of a project.

In addition to the regular meetings of the project, we plan several actions to foster discussion and collaboration in the project: First, we will set up a dedicated forum for communication on the project, most probably adopting the Zulip chat platform which is used in the Coq community. This will permit scientific discussion, reference communication and also internal announcements in the project. Second, we plan to set up a regular internal seminar that we consider keeping online, the purpose of which being mainly to make young researchers more visible in our group: indeed, we noticed that the covid-crisis had a huge impact on how young researchers developed their scientific network and we want to compensate for this, both during the general meetings and with this online seminar. This seminar will be complemented with a reading group that we plan to organize.

The project plans on hiring for several positions. We plan to have a general steering committee for hiring postdoctoral researchers and identifying potential valuable candidates for the various sites of the project. In doing so, we plan to have a particular care in gender equality in the recruitment process from position announcement to candidate selection by using tools for raising awareness about gender biases. We will be using existing tools such as video[1] or documents elaborated in our lab (for instance in IRIF gender-equality committee to which the PI participates). Considering the very low number of female participants to the project, we are considering involving some external members (in particular Assia Mahboubi who had to leave the project to take the lead of her ERC project) in the hiring process.

In the third year of the project, we plan to organize a spring/summer school. An organizing and scientific committee for the school will be set up soon after the start of the project to foresee the school program and its concrete organization (choice of location, format of the school, program, speakers...).

Who does what. Alexis Saurin is the main investigator of this task, and will coordinate goals. Guilhem Jaber, Denis Kuperberg and Luigi Santocanale together with Alexis Saurin are site coordinators. We plan to involve PhD students in organizing the regular seminars and reading groups. Both the hiring committee of the project and the scientific committee of the school will be constituted partly of members of the consortium and partly of external members.

Work Package 1: **Denotational Semantics of Fixed-point Logics and Circular Proofs.** While the μ-calculus [58] has a well-established truth semantics, the project's interest in proofs and programs leads us to shift our attention to proof semantics and denotational semantics. Indeed, in such a setting, semantics shall model formulas altogether with proofs as they shall respectively stand for (coinductive) datatypes and programs.

While for finite proofs, approaches to the semantics of fixed points might be easily developed (for example, via initial or final (co)algebras), the lack of such semantics and the limits of those approaches in the case of non-wellfounded and circular proofs are obvious. In this WP, we plan to develop various denotational semantics for logics with circular proofs and for computational calculi with inductive and coinductive types.

Task 1.1: Totality and Coherence Semantics of Circular Proofs. We will extend the coherence space model developed by Ehrhard and Jafarrahmani for finitary μLL [39] to the circular and non-wellfounded setting. Their model comes with two instances: a relational semantics in which least and greatest fixed points are interpreted in the same way and a more refined interpretation in which sets are equipped with a notion of totality (non-uniform totality spaces), the relations preserving the totality.

A crucial point will be to ensure that infinitary proofs can be interpreted as total morphisms in the model of non-uniform totality spaces. On the categorical side [76] (see next point), thread-validity is intimately related with the existence (and uniqueness) of certain systems of equations that automatically ensures totality.

Another direction will consist in refining models that are naturally equiped with unique fixed points (*eg.* in domains or the topos of trees) in order to distinguish, as above, least and greatest fixed points.

Task 1.2: Categorical and Interactive Semantics of Circular Proofs. The semantics of circular proofs [77, 42] relies on a few elementary concepts from category theory (naturality, the Yoneda Lemma, initial algebras and final coalgebras of functors) and fixed-point theory. We shall investigate how the recent proposals from the syntactic side (notably extension of thread-validity to the multiplicatives) relate to these concepts and whether

[1] https://www.youtube.com/watch?v=g978T58gELo

these concepts may provide a categorical semantics for different systems of circular proofs, especially in the linear and intuitionistic setting.

In addition, we shall build on Clairambault's game-model of μLJ as well as on Baelde, Doumane and Saurin's Ludics model of μMALL [24, 10] in order to design game-theoretical interpretations of non-wellfounded proofs in both linear and intuitionistic logics with least and greatest fixed-points.

Task 1.3: Denotational Semantics for Reactive, Probabilistic and Quantum Languages with (Co)Inductive Types. As an outcome of WP 3, we will design foundational probabilistic and quantum programming languages with inductive and coinductive types and revisit the models of FRP. This goal consists in providing those languages with denotational semantics on the ground of the previous items. In particular, we shall make use of probabilistic coherence spaces and the recent work by Ehrhard, Pagani and Tasson on Probabilistic PCF and on Valiron's expertise on the denotational semantics of quantum programming. Moreover, we plan to use the close relationship of Zamdzhiev's recent categorical constructions [86] with Munch-Maccagnoni's models of linear call-by-push-value in order to apply the latter to model quantum languages.

Who does what. Alexis Saurin is the main investigator of this task, and will coordinate goals. Pierre Clairambault, Thomas Ehrhard, Charles Grellois, Adrien Guatto, Paul-Andre Melliès, Daniela Petrişan, Luigi Santocanale and Lionel Vaux will bring their expertise in denotational semantics and coalgebraic methods and contribute to this task. Task 1.1 is currently investigated by Ehrhard and Saurin with their first-year PhD student Jafarrahmani. Task 1.3 on the quantum programming semantics will be investigated by Valiron, Saurin, Chardonnet and Munch-Maccagnoni while the reactive aspects will be investigated by Guatto and Saurin and a PhD student to start (hopefully) in September 2021.

WORK PACKAGE 2: **Modularity and compositionality of circular proofs.** Even though circular proofs are promising in many respects, their main current shortcoming lies in their lack of compositionality and modularity which has been discussed on page 4, a serious restriction discussed above preventing them from being directly usable for modelling programming. The task at hand is to allow for more compositionality, and to render the criterion as permissive as possible while still guaranteeing termination.

Task 2.1: Relaxing the Validity Condition of Circular Proofs. In the case of cut-free circular proofs, the validity condition is widely accepted as robust and satisfying. It is shown to be compositional in some sense, as it ensures productivity of infinitary cut-elimination and validity is preserved by cut-elimination. Still, the interplay between cuts and validity is not fully understood: as discussed in the context section, cuts may badly interact with the validity condition, and the current validity condition is too restricted. In this task, we shall develop more relaxed validity conditions that provide more compositionality for circular proofs while preserving productivity. Some preliminary results have been recently obtained [8, 66], and the semantical results of Task 1 (notably the game and coherent semantics) shall be useful in this respect. For instance Baelde et al. consider threads endowed with a richer geometry, allowing them to bounce on cuts and axioms [8], in the spirit of Girard's Geometry of Interaction. This yields a more relaxed validity condition where some branches can be validated by threads bouncing on cuts even in the absence of validating straight threads, thereby providing more compositionality for circular proofs. A strong limitation for this approach is the undecidability of the general productivity problem: given an arbitrary circular proof, it is undecidable to recognize whether the program associated to the proof will terminate, as this boils down to solving the halting problem. Therefore, we can only hope to accept as many natural proofs as possible, via a criterion that can be verified algorithmically. It is particularly interesting here to understand which features of circular proofs bring undecidability, and compare them to the features we want to allow for programming purposes: we are interested in designing more expressive validity conditions (decidable and ensuring productivity) and at the same time obtaining some negative results, in order to understand in a fine-grained way the undecidability barrier that we are facing.

Task 2.2: Canonical Circular Proof Objects. It is well-known that sequent calculus proofs suffer non-canonicity (sometimes called "bureaucracy"): many proofs differ only by an irrelevant permutation of their inferences, which is denotationally trivial. Proof-nets were developed as a canonical alternative proofs of the sequent calculus. We shall develop in this task a theory of *circular proofs-nets*. In previous works, De and Saurin introduced *Infinets* [34], a framework extending traditional linear proof-nets to allow for least and greatest fixed-points together with non-wellfounded branches later extended with Pellissier to account for an adequate treatment of the cut inference. Still, the current framework suffers two limitations: (i) it only captures straight-thread validity, not the more flexible validity conditions (ii) no notion of circular (*ie.* finitely presentable) infinet is available yet. We will develop the theory of regular infinets and, once Task 2.1 is achieved, generalize the cut-elimination theorem to more expressive and flexible notions of validity.

Task 2.3: Enriching the Shape of Proof Trees. Proof trees with non well-founded branches also appear naturally when considering transfinite expressions, see Task 5.2. We hope that investigating validity conditions for proof trees in such frameworks can shed a light on dealing with nested fixed points in general.

In another direction, insisting that the number of subtrees is finite, as in regular trees, can be too strong a constraint. Indeed, we could allow a more general class of trees, that still admit finite representations. This would include for instance trees generated by context-free grammars, or by higher-order grammars. Investigating how robust the properties of circular proofs are in these new contexts would likely yield interesting insights, and could expand our understanding of what should be considered a valid proof.

Who does what. Alexis Saurin is the main investigator of this task, and will coordinate goals. PostDoc A, Charles Grellois, Adrien Guatto, Pierre Hyvernat, Guilhem Jaber, Denis Kuperberg, Paul-Andre Melliès, and Alexis Saurin will contribute to Task 2.1. Task 2.2 is currently investigated by Alexis Saurin and his PhD student Abhishek De. Pierre Clairambault shall also work on Task 2.2. Task 2.3 is currently investigated by Denis Kuperberg and his PhD student Emile Hazard and finitely representable non-regular proofs will be investigated by Abhishek De.

WORK PACKAGE 3: **Circular proofs and Computational Calculi.** Circular proofs were originally defined for intuitionistic, modal, and linear logic. A number of alternative logics have been provided with circular proof systems (Kleene algebras, temporal logics, ...) in the recent literature. Following the Curry-Howard correspondence, we plan to cover a systematic investigation to provide circular systems together with their computational interpretation for a wide range of logical systems.

Task 3.1: Circular Typing Derivation. Many systems, like Charity [43] or COQ are based on typed λ-calculi with inductive and coinductive types. Our goal in this task will be to design circular representation of the typing derivation for such calculi. It would enforce termination of programs over inductive types, and productivity of programs over coinductive types. The distinction between inductive and coinductive types follows their representation respectively as smallest and greatest fixed-points.

We first plan to explore a system with only coinductive types, using their encoding as guarded recursive types [69]. Indeed, guarded recursive types have been embedded within rich type systems with dependent types, polymorphism, universes, as can be found in the Calculus of Inductive Constructions, the type system behind COQ. Guarded recursive types rely on the so-called Löb rule as a general coinductive principle. We plan to investigate the representation of this rule using a circular representation and a validity criterion expressed in terms of threads, as explored in Task 5.4. Using the validity criterion that encompass both smallest and greatest fixed points developed in that Task, we then plan to incorporate inductive types.

This system will be designed to handle mutually inductive definitions, and interactions between inductive and coinductive types. Moreover, a comparison with COQ's inductive definitions will be performed, in order to check that elimination schemes that are generated by inductive definitions can indeed be represented in our system. We also plan to investigate relationships with size-change termination techniques [62, 48, 49], as developed in Task 5.1.

Task 3.2: Probabilistic and Quantum programming with (co)inductive types. Probabilistic and quantum programming languages are active research topics in need of higher-level programming features. In this respect, it is important to provide the ability for (co)inductive types in such systems. We plan to design probabilistic and quantum programming languages with inductive and coinductive types, for the probabilistic effect. we shall build on the recent development of probabilistic PCF and its fully abstract semantics.

Regarding quantum programming, we plan to build on preliminary results by Chardonnet, Saurin and Valiron [23] who extended a typed reversible language to inductive and coinductive datatypes, following [75] and are currently studying the type isomorphisms in finitary and infinitary μMALL and their proof-theoretic properties, using [7, 9]. Then we shall extend the logic for supporting superposition of states and unitary operations, building on [75] and compare with alternative recent approaches [70]. This investigation should open the route towards a "fully quantum" logic, suitable for reasoning about general quantum datatypes and quantum algorithms.

Task 3.3: Automata and Kleene Algebras. Denis Kuperberg, Laureline Pinault and Damien Pous established links between cyclic proof systems for Kleene algebra and automata models as well as complexity classes, via the Curry-Howard correspondence. The cut-free system characterizes regular languages (without contraction) and LOGSPACE (with contraction) [59], while the system with cut characterizes primitive recursive functions (without contraction) and Gödel's System T (with contraction) [60]. These approaches can later be generalized to other systems using less constrained forms of fixed points (the Kleene star being a particular case of least fixed point).

The knowledge acquired during the study of Kleene algebra is likely to be useful for a general understanding of circular proofs. For instance in [60], in order to obtain the characterization of primitive recursion functions, an algorithm was built to assign ranks to simultaneous cycles in a circular proof. This allows to explicit a nesting hierarchy between these cycles, and to turn the cyclic proof into a λ-term with nested recursion operators. We hope this approach can adapted for a better understanding of circular proofs in a general setting.

Who does what. PostDoc A, Guilhem Jaber, Kenji Maillard, Pierre-Marie Pédrot, Luigi Santocanale and Alexis Saurin will work on Task 3.1. Benoit Valiron, Kostia Chardonnet, Farzad Jafarrahmani, Alexis Saurin, Charles Grellois and Guillaume Munch-Maccagnoni will work on Task 3.2. Denis Kuperberg, and Damien Pous will work on Task 3.3.

WORK PACKAGE 4: **Circularity and constructivity issues.** Constructivity issues arise when studying circular proofs: (i) Towards having decision procedures or even a feasible syntax, Brotherston and Simpson firstly asked when a given infinitary circular proof system is equivalent to the corresponding finitary one. It is almost straightforward that finitary proofs, circular or not, can be embedded in infinitary circular proofs. However, the converse inclusions are open for many logical frameworks (with the notable exception of circular arithmetic [11, 80]). We shall investigate finitization of circular proofs in variants of the μ-calculus. (ii) Another problem that we aim to address is the extraction of validity-witnesses from circular proofs. This might be tackled via the formalization in COQ (thus in a constructive setting) of these systems and of algorithms for deciding the validity conditions. (iii) Finally, we aim at systematically relating circular proof systems to proof systems based on ordinals and ordinal notations. While circular proof systems are tailored on Tarski's impredicative definition of least fixed points, proof systems based on ordinal notations reflect our understanding of least fixed points as built via successive applications of a function, that is, by Kleene's iteration.

Task 4.1: Finitization of Circular Proof Systems. We shall tackle the finitization problem for variants of the μ-calculus in the context of provability logic. While tackling a given variant of the μ-calculus might be of interest, the most ambitious goal is to identify uniform criteria ensuring finitization.

A related direction will consist in investigating implications of finitization for proof-theory; here, proof equivalence is determined either by a denotational semantics, or by the equivalences induced by the cut-elimination procedure. We shall rely on the work achieved in WP 1 to develop this second step.

Task 4.2: Formalization of Soundness. We shall formalize in COQ circular proofs and their validity conditions, the standard ones as well as the criteria designed in Task 2.1. We also aim at formalizing some of the meta-theory of circular proof, most notably the decidability of the validity condition. To this end, we will extend the Yalla library (`https://perso.ens-lyon.fr/olivier.laurent/yalla/`, that provides tools for the syntactic study of linear logic variants in COQ. The current release is restricted to finitary sequent calculus systems. We want to extend it to more general notions of proofs, notably circular proofs. Some experiments in this direction have already been performed by Saurin and students. More graphical representations of proofs such as proof nets also have to be considered, finally leading to a formalization of appropriate notions of circular proof nets. Formalization of the validity condition and its validity will rely on Work package 6, and more specifically Task 6.4, in which we shall develop a library for ω-automata. This formalization could then be used to get certified algorithms that automatically check validity of proofs in these logical systems, using the extraction feature of COQ to programming languages like OCaml.

Task 4.3: Ordinal-based Representations. We shall investigate relations between circular proof systems and logical systems based on ordinals. While the two kinds of proof systems have similarities, it is not obvious to develop uniform translations, since on the semantic side the least fixed-points characterizations of each system (Tarski vs. Kleene) coincide only on complete lattices and in a classical meta-theory. This identity falls apart as soon as: (i) the poset considered are not complete, see [79], (ii) the meta-theory is intuitionistic, see [57, 82], (iii) the focus is shifted from provability to proofs and their categorical models.We will approach the problem by developing translations for some selected simple systems of circular proofs and systems with ordinals and then generalize and develop a uniform approach to inter-translating between these two kinds of systems. In particular, we will formalize the size-change principle as an ordinal-based logical system and as a system of circular proofs.

Task 4.4: Fixed-Points Logics. We shall try to answer some challenging concrete questions, such as: (i) does any fixed point logic have a presentation by means of circular proofs? (ii) if a fixed-point logical system has such a circular presentation, is it the case that least-fixed points are constructible by Kleene's iteration in the free fixed-point algebras presented by circular proof systems? (iii) otherwise, what are the conditions needed for this? (iv) do finite circular proof systems ensure general completeness theorems? (v) what is the metatheory necessary

to achieve any of the previous results? (vi) use the knowledge achieved so far to develop an intuitionistic theory of ordinals based on circular proof systems.

Quantitative reasoning, as discussed along with Fig. 4, p. 5 will be a natural framework for this program: how to advance the theory of circular proofs in the context of quantitative semantics? As shown in the example, this type of loop is not valid in the usual framework of circular proofs. What validates this reasoning comes from the real–valued semantics under consideration and the Archimedean property of $\mathbb{R}$. This leads to some interesting lines of research: identifying the additional reasoning principles (e.g., the Archimedean property) available in quantitative semantics, identifying syntactic conditions on cycles ensuring their validity, exploring natural theoretical questions such as, e.g., completeness of these syntactic conditions and develop the corresponding proof theory (e.g., cut–elimination, *etc.*). Understanding the smooth connection relating 2-valued logic to multivalued logics and metric spaces, see e.g. [61, 71], will allow to find the appropriate ideas to generalize circular proof systems to quantitative logics and to exploit them in this context.

Who does what. Luigi Santocanale is the main investigator of this task, and will coordinate goals. Luigi Santocanale and Alexis Saurin shall work on goal Goal 4.1. Olivier Laurent and Damien Pous shall work on Goal 4.2. Luigi Santocanale, Pierre Hyvernat, and the recruited PhD student shall work on Goals 4.3 and 4.4. Matteo Mio and Daniela Petrişan will work on Task 4.4.

WORK PACKAGE 5: **Circular Program analysis and verification of infinite systems.** Circular proofs offer a powerful tool to perform verification tasks, i.e. guarantee that a system behaves according to a specification. These specification can range for simple termination to complex behaviours specified by different formalisms, such as regular expressions or logics. We are also interested in testing program equivalence, for instance to guarantee that a new implementation behaves in the same way as the previous one in any context. We shall build on preliminary works by the partners to develop tools and techniques for program analysis and verification.

Task 5.1: Program Termination. Correctness of circular proofs relate to program termination, and it is therefore very natural to use circular proofs as a certificate for program termination. Preliminary work on this approach has shown how to use the size-change principle [62] to check validity of recursive functions with coinductive types [49]. One issue encountered is that the problem of verifying coinductive types is subtler than plain termination as there exist invalid well-typed definitions that terminate [5]. Extending this work to more expressive settings (e.g. dependent types) is a natural next step, that could use work done in the Task 3.1.

Task 5.2: Regular Expression Inclusion. Circular proofs offer a promising framework for checking inclusion of various kinds of regular expressions. While, this approach is well-studied for regular expressions [32], we plan to design proof-based algorithms for inclusion of ω-regular expressions (which describe languages of infinite words, a standard formalism in verification). This approach could be further extended to transfinite expressions, to encompass more complex specifications. Apart from its direct applications, we hope that this project will allow us to better understand how nesting fixed points can be dealt with in circular proofs. Indeed, regular expressions correspond to the case where only least fixed points are present, while transfinite expressions are allowed to nest least fixed points ($*$) and greatest fixed points (ω). An example of such an expression is $(a+b)^\omega(b^*a^\omega + a)^\omega$, which describes a language of transfinite words, containing for instance the word $(ab)^\omega bbbba^\omega bba^\omega a^\omega$. Generalizing circular proof systems for regular expressions such as the ones from [32] to ω-regular expressions, describing words of length ω, is already a challenging problem. Going even further, the model of words of ordinal length can be seen as a toy model for the nesting of fixed points. In the realm of inclusion checking algorithm for expressions, our algorithms building circular proofs would be more compositional than classical automata-based algorithms, as the cut is a versatile way to combine inclusion certificates. This corresponds to potential applications of the proof systems we aim at designing in Task 2.3. It should be noted that the theoretical complexity for inclusion of ω-regular expressions is known to be PSPACE-complete. Although this approach with circular proofs cannot improve on this, it can offer a formalism which is more convenient as a certificate of correctness. in particular, compositionality properties will allow to compose certificates for different components of a system into a certificate for the whole system. This operation typically cannot be done with the usual automata-based approach to verification, where an algorithm has to be run on the automaton modelling the system, and the algorithm has to be run again when the system is modified.

Task 5.3: Automating Circular Proofs and Program Equivalence. Circular proofs have been particularly useful to design automated theorem provers, notably in the setting of separation logic with inductive predicates [18]. Following this line of work, we will explore the design of automated tools based on the circular systems designed in the Work Packages 2 and 3.

As a first step, we will explore the decidability of program equivalence for the basic calculus of circular proofs given in [77]. Going further, we will explore the possibility of transferring these techniques to the Coq proof assistant by proving automatically the equivalence of recursively-defined programs encoded as interaction trees [84]. The equational theory of interaction trees has been developed up-to a notion of weak bisimulations defined and reasoned on via parameterized coinduction [85], formalized in Coq using Paco. We plan more specifically to use circular representations of parameterized coinduction, as developed in Task 6.2, to explore how to automate the construction of bisimulation proofs between interaction trees.

A more fundamental line on automated circular proofs will consist in developing proof-search methods for the circular proof nets of Task 2.2, thus quotienting the proof-search space by permutative equivalence.

Task 5.4: Temporal Refinements for Guarded Recursive Types. Guilhem Jaber and Colin Riba have introduced in [50] a framework for verifying functional reactive programs, based on a type system that combines guarded recursive types, to enforce productivity, with temporal refinements, for specifying safety and liveness properties. These refinements are given as coalgebraic μ-calculus formulas.

We plan to develop a new version of this system that would be based on a circular representation of derivations. It would combine the circular representation of guarded recursive types, as developed in Task 3.1, together with the circular proof systems for the μ-calculus, as developed in [38, 4]. Using a circular representation for both types and refinements would allow for a transfer of information from refinements to types. This would permit to use refinements in order to type more programs, whose productivity depends on the refinement. Moreover, the design on an embedding of this system in Coq is planned, that would use Task 6.2.

Who does what. Pierre Hyvernat and the recruited PhD student will contribute on Tasks 5.1 and 5.3. Denis Kuperberg and Emile Hazard shall work on Task 5.2. Guilhem Jaber will work on Task 5.3 and 5.4, Abhishek De, Luigi Santocanale, Alexis Saurin and Yannick Zakowski on Task 5.3 and Colin Riba on Task 5.4.

Work Package 6: **Enhancing coinductive reasoning in Coq.** Coq is a widely used proof assistant based on the *Calculus of Inductive Constructions*. As the name implies, it prominently features inductive types, together with a heavy support from the tactic language that comes bundled with it. While it also features coinductive types, both their foundational properties and practical tooling are deficient. This task aims at enhancing the situation on both sides.

Task 6.1: Foundation of Coinductive Types. Coinductive types as they were introduced in Coq are well-known to introduce a breakage of subject reduction. On foundational grounds, this is highly problematic from the programming language point of view, and this subtask is dedicated to solving this issue. There are various potential ways out. First, we shall work at the integration of copatterns [2, 3], a language construct that can be used to cleanly define computations over infinite structures through their *observations*. Copatterns rely on a negative formulation of coinductive types, already implemented in Coq, which enjoy subject reduction but have little high-level support compared to inductive types. By integrating copattern-matching in the system, we hope to provide a solution that unifies induction and coinduction, compatible with dependent types and that keeps the good meta-theoretical properties of the system. We will in particular work on extending the Equations compiler [81] which already includes a high-level syntax for dependent pattern-matching to handle copatterns. For backwards compatibility, we shall also consider other solutions, based on syntactic criteria that are closer to the usual treatment of coinduction in Coq.

Task 6.2: Parameterized Coinduction. As a mean to provide practical tools to users of Coq, we notably plan to develop further the Paco library. Paco provides a compositional way to perform coinductive proofs in Coq, building incrementally the coinductive predicate that one traditionally must exhibit upfront. This work has been a major step toward making coinductive proofs in Coq as easy to perform as their inductive counterpart, with similar tactic support. Paco is however currently restricted to coinductive predicates (living in the Prop universe), and is hence of no help to build coinductive objects living in other universes, in Type in particular. We wish to address this shortcoming. In another direction, we also plan to connect Paco to foundational works done on the circular representation of type systems, as developed in Task 3.1, in order to explore automation of reasoning with Paco.

Task 6.3: Built-in Compositional Coinductive Reasoning. The syntactic guard condition used in Coq to check for (co)fixed-point productivity is a well-known source of problems, including lack of modularity, brittleness and poor foundational justifications. We will study the extension of this guard condition with techniques coming from the world of circular proofs. We aim at providing a comprehensive theoretical treatment of the interaction between inductive and coinductive types in dependent type theory. Surprisingly, this is still missing, despite

incremental results, notably from the Agda community. Moreover, mixing inductive and coinductive types in Coq typically incurs a high expressivity limitation, which requires intricate and fragile workarounds. This is especially problematic in so far as on paper it is standard to mix inductive steps of reasoning with coinductive steps, e.g. when considering fairness predicates. We consider this as a barrier to proof mechanization which should be tackled theoretically, and implemented in practice in Coq.

This subgoal depends on Tasks 6.1, and would benefit from the results obtained in Tasks 3.1,3.2 and 5.1.

Task 6.4: Formalization using Circular Reasoning. Along with the contributions of the previous three tasks, we will also pursue a formalization effort of computational and mathematical results related to the project. A natural direction will consist in formalizing several classical mathematical results using *infinite descent*: one such result would be Fermat's last theorem for $n = 3$ the proof of which can be done by infinite descent but is much more involved than for $n = 4$ [35]. More related to the results of the project, in direct connection with Task 4.2, we will develop a Coq library to formalize and reason about various classes of ω-automata, building on preexisting formalizations of Büchi automata in Coq [64] but targeting better-behaved classes of automata on infinite words, such as parity automata, which have not yet been formalized nor their rich meta-theory. Still in connection with Task 4.2, we will extend the Yalla library with a support for fixed-points and circular proofs, formalizing their validity conditions.

Who does what. Pierre-Marie Pédrot, Matthieu Sozeau and the recruited post-doctorate in Nantes will work on Tasks 6.1-3. Damien Pous will contribute on Tasks 6.2-4. Guilhem Jaber and Yannick Zakowski will work on Tasks 6.2. The recruited PhD student, Pierre Hyvernat, Luigi Santocanale and Kenji Maillard will contribute on Tasks 6.1 and 6.3. Alexis Saurin will contribute to Task 6.4

Methodology and Risks of the Proposal The methodology of the project (and its broad spectrum, ranging from the most abstract semantical contributions to the principled implementation of coinduction in proof assistants) results from our analysis (that we partly build from the experience and result of the previous ANR JCJC project RAPIDO concluded in September 2019) that circular proofs reached a degree of maturity such that we hope to apply them in the short term to the design and analysis of programs even if some limitations are still to be unlocked, notably a compositionality issue. The organization of the tasks (as well as composition of the consortium, see below) reflects this analysis: the first three tasks deal with questions of a more fundamental nature while the last three are more focused towards applications of circular proofs. Wrt. risks, we identified the following two main risks: (i) achieving enough compositionality or lifting circular proofs to dependently-typed setting might be out of reach and (ii) other groups may obtain results we expect prior to our project (on some aspects of Task 4, there is an intense research effort in Italy, Slovenia, Sweden, etc.). Risk (i) is quite low (our preliminary investigations on bouncing threads, semantics and on mixing circular proofs systems with guarded recursive types are promising) and it would not prevent us from achieving major contributions in each and every tasks of the project; risk (ii) is inherent to active research and we are quite confident as the consortium is ahead (most notably on Tasks 1, 2, 5 and 6); the project will also allow to strengthen international collaborations through invitations of colleagues.

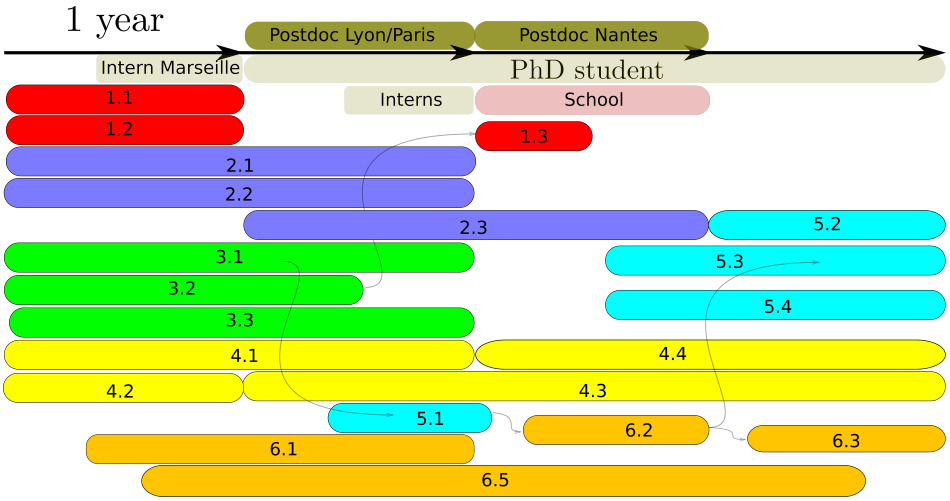

Figure 5: Organization in time and dependencies between tasks

2 Organization and implementation of the project

2.1 Scientific coordinator and its consortium

Scientific coordination. The scientific coordinator of the project will be **Alexis Saurin**, who will also coordinate the site of Paris (IRIF). He is CR CNRS at IRIF and team leader of the INRIA team $\pi.r^2$ (INRIA Paris-Rocquencourt)[29] since October 2019.

Alexis Saurin is an expert in computational logic and the theory of programming languages: his field ranges from the computational interpretations of proof theory to interactive semantics of programming languages and logics, with a specific focus on the the Curry-Howard correspondence, classical and linear proof theory and on circular and non-wellfounded proofs for fixed-point logics. His current research effort mainly focuses on unveiling the computational content of circular proofs and transferring this logical artifact to programming.

In addition to the INRIA team management, his experience is complemented with the scientific coordination of the INRIA associated team SEMACODE (2011-2013, France-Serbia-USA) and various collective responsibilities (GT-Scalp of GDR-IM, ...) and, more importantly, the scientific coordination of the ANR JCJC project RAPIDO (from jan. 2015 to oct. 2019) in which a significant part of the meta-theory of circular proofs that will be used in **ReCiProg** has been developed. In 2018, he chaired a workshop on the topics of circular proofs and coinduction, organized in Oxford as part of FLoC 2018; he was also involved in a follow-up edition organized late 2019 in Sweden by Afshari, while a forthcoming edition is planned in the future, see below.Saurin will be mainly involved in Tasks 1, 2, 3, 4 and 6 and three PhD students is currently (co-)advises (Chardonnet, De and Jafarrahmani) who will be involved in **ReCiProg**. He will devote about 65% of his research time to the project.

More details on scientific coordination of the project are provided as Work-Package 0.

Description of the consortium. The consortium consists in researchers gathered in four sites, located in Lyon (LIP), Marseille (LIS), Nantes (LS2N) in addition to Paris (IRIF).

The site of Lyon (LIP and LAMA) will be coordinated by **Denis Kuperberg** (CR CNRS LIP, 50%, Tasks 2,3 & 5). He is specialized in automata theory and its links with verification, logics, proof theory and algebra. The site of Marseille (LIS and I2M) will be coordinated by **Luigi Santocanale** (Prof. Univ Aix-Marseille, 55%, Tasks 1, 4, and 6). He is expert on fixed point theory of monotone functions, both at the order theoretic level (provability) and at the category-theoretic level (proof theory). The site of Nantes (LS2N) will be coordinated by **Guilhem Jaber** (MCF, univ Nantes, 33%, Tasks 2,3 & 5). He is specialized in semantics and analysis of programming languages, and type theory (including guarded recursive types). Besides the four coordinators, the other participants of the projects are (by sites – PhD students are not listed at that stage): Lyon (Pierre Clairambault, Olivier Laurent, Matteo Mio, Damien Pous, Colin Riba & Yannick Zakowski), Marseille (Charles Grellois & Lionel Vaux), Nantes (Kenji Maillard, Guillaume Munch-Maccagnoni, Pierre-Marie Pédrot & Matthieu Sozeau), and Paris (Thomas Ehrhard, Adrien Guatto, Hugo Herbelin, Pierre Hyvernat, Paul-André Melliès, Daniela Petrişan & Benoît Valiron).

Originality, complementarity and strength of the consortium. **ReCiProg** proposes to build quite a large consortium, for two main reasons: the main goal of the project is to put circular proofs to use in the Curry-Howard perspective targeting notably coinduction in Coq; this requires both theoretical developments on the meta-theory of circular proofs as well as software developments of Coq. Additionally a research community is currently growing and spreading in France with an interest in circular proofs that the project will contribute to structure. The main strength of the consortium is actually in that it gathers top researchers involved in denotational semantics, (circular) proof theory, program analysis and verification, type theory and proof assistants (members of the Coq development team are part of the project, including Sozeau and Herbelin, respectively the current and previous coordinators of Coq development). About half of the members of the project are Coq developers or Coq users. Most of the project members are actually involved in at least two of the above scientific categories, allowing to foresee a dense network of potential collaborations within the project. Also, the project involves senior researchers as well as young researchers (recently hired ones as well as post-docs).

Collaborations between members on the consortium on the topic have already been started. Alexis Saurin and Denis Kuperberg, together with David Baelde and Amina Doumane, are investigating an extended criterion for circular proofs in μMALL [8]. Denis Kuperberg and Damien Pous, together with their Phd student Laureline Pinault, are studying the Curry-Howard correspondence in a context of Kleene Algebra [59, 60]. Matteo Mio and Denis Kuperberg have also each recently hired a PhD student (Christophe Lucas and Emile Hazard respectively) to work on related topics. Paul-André Melliès and Charles Grellois have a long-standing collaboration on proof

systems for linear logics with fixed points. The expertise of Damien Pous, Denis Kuperberg, Matteo Mio and Daniela Petrişan in automata theory and Kleene algebra offers a valuable complement, and expands the variety of tools at the disposal of the team. Hugo Herbelin, Pierre-Marie Pédrot, Matthieu Sozeau are core member of the development team of COQ and are experts on the type theory on which this proof assistant is based on. This expertise will be crucial in order to design and implement the modifications of COQ needed to support circular reasoning. Olivier Laurent, Kenji Maillard and Damien Pous have a recognized expertise in formalization in proof assistants, that would be needed to design the precise expectations of the extension of COQ with circular reasoning, to be useful in such formalizations. Guillaume Munch-Maccagnoni is an expert on calculi for proofs/programs with resources and effects, and their algebraic models. Colin Riba is a specialist of Proof Theory and in particular of the Curry-Howard "proofs-as-programs" correspondence, for MSO logics and automata theory. Lionel Vaux is expert in denotational semantics of various calculi.

2.2 Brief description of objectives for the non-permanent researchers to be hired by the project

2.2.1 Partner 1 : IRIF – Paris

The Post-doc will work on WP 2 and 3, mainly on the issue of compositionality of circular proof and the design of computational systems based on circular proofs as well as the cross-fertilization between guarded recursive types and circular proofs. The post-doc will be supervised by Alexis Saurin and will benefit from the environment of IRIF site members, notably Guatto, Herbelin, Melliès for his/her purpose.

2.2.2 Partner 2: LS2N – Nantes

The postdoc would work on WP 6, in coordination with Guilhem Jaber, Pierre-Marie Pédrot and Matthieu Sozeau. His goal would be to pursue the theoretical results of Work Package 3, in order to incorporate circular reasoning into COQ.

2.2.3 Partner 3 : LIP – Lyon

The postdoc shall work on different aspects of WP 3, in coordination with Denis Kuperberg, Damien Pous and Matteo Mio. One of the objectives is to succeed Laureline Pinault currently finishing her thesis which brought first contributions to Goal 3.5, and to extend the techniques developed in her thesis to more general frameworks. Depending on her/his expertise in COQ and software developments, he/she may also contribute to WP 4 and 6.

2.2.4 Partner 4 : LIS – Marseille

Staff Expenses The two internships we have asked for shall attract young researchers towards the subject of the project during its first year. At the end of the internships we plan to hire as a PhD the student of the internships that has shown the better aptitude to research. We do not exclude, however, to announce the PhD position on various mailing lists and hire more qualified candidates if needed.

The PhD student shall be supervised by Luigi Santocanale (50 %) and Pierre Hyvernat (50 %). The exchanges among the members of the project have emphasized that the need to recruit good PhD students is more important for the Marseille partner and at the Université Savoie Mont Blanc to which Pierre Hyvernat is affiliated. Other sites acknowledge their capacity to attract qualified students. Pierre Hyvernat and Luigi Santocanale know each other very well, since Hyvernat doctoral studies in Marseille, and look forward to this collaboration on co-tutoring the PhD student. This co-tutoring will be the occasion for Pierre Hyvernat to complete his training as a research director and obtain his Habilitation Thesis. In order to foster this collaboration and the co-tutoring, a special budget has been asked for to allow longer visits (mostly of the PhD student) between Marseille and Chambéry.

Themes, methodology, and risks of the PhD tutoring. The PhD student shall develop his research along three axes that might be tackled in a chronological order—even if this is more a pedagogic approach than a necessity. The axes are: (a) comparison of circular proof systems with other formalisms for fixed-points and (co)induction (systems based on ordinals, size-change principle, ...); this will include coverage of existing literature and devising formal translations between the systems; (b) semantic study of guard conditions w.r.t. standard semantics of programming languages and proof assistants (PER and realisability models); (c) implementation within COQ of algorithms for termination verification based on established validity of guard conditions. These axes are related to many of our tasks (1.1, 1.2, **4.2**, **4.3**, **5.1**, 5.3, 6.1, **6.3**). Many elementary questions may

be raised w.r.t. these axes, since they can be answered simply by making precise the context of the questions. Other goals and questions are more challenging, but they can be approached by systematically producing partial results. Therefore, we believe that the PhD student will have no difficulties advancing with his research.

3 Impact and benefits of the projects

Our project is mostly a fundamental research project, so our first and main impact will consist in developing *new fundamental results*. Still, **ReCiProg** also has a strong ambition to have an impact on *Software development* and, first of all, on the treatment of coinduction in Coq proof assistant. Besides those expected impacts from the project, a major benefit will also consist in *strengthening and shaping the young but growing scientific "circular community"* of french researchers interested in circular proofs. Last, the project aims at *disseminating the results*, both in direction of experts and colleagues from the field (through seminars, conferences and publications) but also to researchers who are less focused on circular proofs but could benefit from them and apply those techniques to their problems. We also aim at a dissemination to a general public, as we think it is important that researchers are involved in popularisation activities, and we believe this topic could sparkle the interest of people that do not have formal mathematical training.

Fundamental results First of all, we aim at contributing to the meta-theory of circular proof and their theoretical applications which is an active research topic at the international level. The impact of understanding these objects will likely reach other fields of theoretical computer science and mathematics, and can irrigate theoretical research in connected fields: automata theory, verification, complexity theory, measure theory. We plan that our results will be published in top international conferences and journals.

Software Unveiling the concrete Curry-Howard correspondence between circular proofs and coinductive programming will allow us to treat coinductive constructions in Coq. This can be instantiated by additions to the Coq kernel allowing more flexible syntax, or in more specialized libraries to verify the validity of circular proofs, or to extend current Coq tactics to coinductive data. (The project gathers researchers from two of the Inria teams mainly involved in the development of Coq as well as the current main developers of Coq proof assistant.) Other software developments include verification and program analysis tools: their purposes will be to verify the correctness of various components (hardware, software), using the new model-checking algorithms brought by circular proofs.

Community building This project will strengthen and shape a french community working on circular proofs, allowing more communication and making explicit the relationships between different projects. This goal is in particular at the center of the organization of a summer school affiliated with the project to bring young researchers as well as interested colleagues to the project's topics. This will in turn ease the process of connecting with international researchers working on similar topics, notably in United Kingdom, Sweden, Italy, Netherlands, USA as well as Japan. Also, our young "circular community" is having a growing number of PhD students graduating and the post-doc positions offered in the project will allow us to attract promising PhD who recently graduated in France as well as international PhDs from the teams contributing to the field. Visits to and invitations of foreign colleagues will be an important part of this international community-building. To name a few people we consider inviting or visiting (but also involving in the spring school, see below), one can think of Andreas Abel, Bahareh Afshari, Stefano Berardi, James Brotherston, Anupam Das, Liron Cohen, Graham Leigh, Reuben Rowe, Alexandra Silva, Alex Simpson, Makoto Tatsuta or Yde Venema, *etc.*

In the longer term, the shaping of such a community, at the french and international level, could result in an international research network. Early initiatives are already taking place: for instance a series of workshops started in Oxford in july 2018 (funded by ANR as part of RAPIDO project) continued in Sweden in november 2019 while a forthcoming edition originally planned in the UK in 2021 was postponed due to the pandemics.

Dissemination By expanding the existing body of knowledge on circular proofs, and giving talks about this research to a larger community, our goal is to spread the idea of using circular proofs instead of well-founded ones to researchers from other subfields. Indeed, this concept is interesting beyond the areas of proof theory and verification. Researchers from any field of mathematics or computer science can benefit from knowing the theory of circular proofs, but also getting more familiar with coinduction and mechanized proofs. To this aim, the school will be an important landmark of the project. Popularisation talks are also part of this goal, as the idea of circular proof is suitable to awaken the interest and curiosity of a general public. This can be done for

instance via the Pint of Science festival occurring each May, in which Denis Kuperberg and Pierre Clairambault were already involved several times as organizers and speakers, or during the “Fête de la Science" each October.

References

[1] Andreas Abel. Type-based termination, inflationary fixed-points, and mixed inductive-coinductive types. In Dale Miller and Zoltán Ésik, editors, *Proc. of FICS 2012*, volume 77 of *EPTCS*, pages 1–11, 2012.

[2] Andreas Abel and Brigitte Pientka. Wellfounded recursion with copatterns: a unified approach to termination and productivity. In *ICFP*, pages 185–196. ACM, 2013.

[3] Andreas Abel, Brigitte Pientka, David Thibodeau, and Anton Setzer. Copatterns: programming infinite structures by observations. In *POPL*, pages 27–38. ACM, 2013.

[4] Bahareh Afshari and Graham E. Leigh. Cut-free completeness for modal mu-calculus. In *LICS 2017*, pages 1–12. IEEE, 2017.

[5] Thorsten Altenkirch and Nils Anders Danielsson. Termination checking in the presence of nested inductive and coinductive types. In Ekaterina Komendantskaya, Ana Bove, and Milad Niqui, editors, *PAR-10. Partiality and Recursion in ITP*, volume 5 of *EasyChair Proc. in Computing*, pages 101–106. EasyChair, 2012.

[6] Carroll Morgan Annabelle McIver. Results on the quantitative μ-calculus qmμ. *ACM Trans. Comput. Log.*, 9(1), 2007.

[7] David Baelde. Least and greatest fixed points in linear logic. *ACM Transactions on Computational Logic*, 13(1), January 2012.

[8] David Baelde, Amina Doumane, Denis Kuperberg, and Alexis Saurin. Bouncing threads for infinitary and circular proofs. preprint, available at `http://arxiv.org/abs/2005.08257`, 2020.

[9] David Baelde, Amina Doumane, and Alexis Saurin. Infinitary proof theory: the multiplicative additive case. In *CSL 2016*.

[10] David Baelde, Amina Doumane, and Alexis Saurin. Least and greatest fixed points in ludics. In *CSL 2015*, pages 549–566, 2015.

[11] Stefano Berardi and Makoto Tatsuta. Equivalence of inductive definitions and cyclic proofs under arithmetic. In *LICS 2017*, pages 1–12. IEEE Computer Society, 2017.

[12] Yves Bertot. Filters on coinductive streams, an application to eratosthenes' sieve. In Pawel Urzyczyn, editor, *TLCA*, volume 3461 of *LNCS*, pages 102–115. Springer, 2005.

[13] Yves Bertot and Ekaterina Komendantskaya. Inductive and coinductive components of corecursive functions in coq. In Jirí Adámek and Clemens Kupke, editors, *Proceedings of CMCS 2008*, volume 203 of *ENTCS*, pages 25–47. Elsevier, 2008.

[14] Julian Biendarra, Jasmin Christian Blanchette, Aymeric Bouzy, Martin Desharnais, Mathias Fleury, Johannes Hölzl, Ondrej Kuncar, Andreas Lochbihler, Fabian Meier, Lorenz Panny, Andrei Popescu, Christian Sternagel, René Thiemann, and Dmitriy Traytel. Foundational (co)datatypes and (co)recursion for higher-order logic. In *FroCoS 2017*, pages 3–21, 2017.

[15] Jasmin Christian Blanchette, Aymeric Bouzy, Andreas Lochbihler, Andrei Popescu, and Dmitriy Traytel. Friends with benefits - implementing corecursion in foundational proof assistants. In *ESOP 2017*, pages 111–140, 2017.

[16] James Brotherston. Cyclic proofs for first-order logic with inductive definitions. In Bernhard Beckert, editor, *TABLEAUX*, volume 3702 of *LNCS*, pages 78–92. Springer, 2005.

[17] James Brotherston. *Sequent Calculus Proof Systems for Inductive Definitions*. PhD thesis, University of Edinburgh, November 2006.

[18] James Brotherston, Nikos Gorogiannis, and Rasmus L. Petersen. A generic cyclic theorem prover. In *Proceedings of APLAS-10*, LNCS, pages 350–367. Springer, 2012.

[19] James Brotherston and Alex Simpson. Complete sequent calculi for induction and infinite descent. In *LICS*, pages 51–62. IEEE, 2007.

[20] James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 21(6):1177–1216, December 2011.

[21] James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *J. Log. Comput.*, 21(6):1177–1216, 2011.

[22] Andrew Cave, Francisco Ferreira, Prakash Panangaden, and Brigitte Pientka. Fair reactive programming. In *POPL '14*, pages 361–372. ACM, 2014.

[23] Kostia Chardonnet, Alexis Saurin, and Benoît Valiron. Toward a curry-howard equivalence for linear, reversible computation - work-in-progress. In *RC*, volume 12227 of *Lecture Notes in Computer Science*, pages 144–152. Springer, 2020.

[24] Pierre Clairambault. Least and greatest fixpoints in game semantics. In *FOSSACS*, pages 16–31, 2009.

[25] J. Robin B. Cockett and Dwight Spencer. Strong categorical datatypes II: A term logic for categorical programming. *Theor. Comput. Sci.*, 139(1&2):69–113, 1995.

[26] Robin Cockett and Dwight Spencer. Strong categorical datatypes I. In R. A. G. Seely, editor, *International Meeting on Category Theory 1991*, Canadian Mathematical Society Proceedings. AMS, 1992.

[27] Jesper Cockx and Andreas Abel. Elaborating dependent (co)pattern matching: No pattern left behind. *J. Funct. Program.*, 30:e2, 2020.

[28] Liron Cohen and Reuben N. S. Rowe. Non-well-founded proof theory of transitive closure logic. *ACM Trans. Comput. Log.*, 21(4):31:1–31:31, 2020.

[29] collective. Pi.r2 activity report 2020. Available at `https://raweb.inria.fr/rapportsactivite/RA2020/pi.r2/pi.r2.pdf`, 2021.

[30] Thierry Coquand. Infinite objects in type theory. In *TYPES'93*, volume 806 of *LNCS*, pages 62–78. Springer, 1994.

[31] Anupam Das, Amina Doumane, and Damien Pous. Left-handed completeness for kleene algebra, via cyclic proofs. In *LPAR*, volume 57 of *EPiC Series in Comp.*, pages 271–289. EasyChair, 2018.

[32] Anupam Das and Damien Pous. Non-wellfounded proof theory for (kleene+action)(algebras+lattices). In *27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4-7, 2018, Birmingham, UK*, pages 19:1–19:18, 2018.

[33] Christian Dax, Martin Hofmann, and Martin Lange. A proof system for the linear time μ-calculus. In S. Arun-Kumar and Naveen Garg, editors, *FSTTCS 2006*, volume 4337 of *LNCS*, pages 273–284. Springer, 2006.

[34] Abhishek De and Alexis Saurin. Infinets: the parallel syntax for non-wellfounded proof-theory. In Serenella Cerrito and Andrei Popescu, editors, *TABLEAUX 2019*, volume 11714 of *Lecture Notes in Computer Science*, pages 297–316. Springer, 2019.

[35] David Delahaye and Micaela Mayero. Diophantus' 20th Problem and Fermat's Last Theorem for n=4: Formalization of Fermat's Proofs in the Coq Proof Assistant. 16 pages, October 2005.

[36] Amina Doumane. Constructive completeness for the linear-time μ-calculus. In *LICS*, pages 1–12. IEEE Computer Society, 2017.

[37] Amina Doumane. *On the infinitary proof theory of logics with fixed points. (Théorie de la démonstration infinitaire pour les logiques à points fixes)*. PhD thesis, Paris Diderot University, France, 2017.

[38] Amina Doumane, David Baelde, Lucca Hirschi, and Alexis Saurin. Towards completeness via proof search in the linear time μ-calculus: The case of Büchi inclusions. In *LICS '16*, pages 377–386, 2016.

[39] Thomas Ehrhard and Farzad Jafarrahmani. Categorical models of Linear Logic with fixed points of formulas. to appear in LICS 2021, February 2021.

[40] Sebastian Enqvist, Fatemeh Seifan, and Yde Venema. Completeness for the modal μ-calculus: Separating the combinatorics from the dynamics. *Theor. Comput. Sci.*, 727:37–100, 2018.

[41] Gadi Tellez Espinosa and James Brotherston. Automatically verifying temporal properties of programs with cyclic proof. In *Proceedings of CADE-26*, 2017.

[42] Jérôme Fortier and Luigi Santocanale. Cuts for circular proofs: semantics and cut-elimination. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013*, volume 23 of *LIPIcs*, pages 248–262. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[43] Tom Fukushima and Charles Tuckey. *Charity User Manual*, January 1996. preprint, `http://www.cpsc.ucalgary.ca/projects/charity/home.html`.

[44] Eduardo Giménez and Pierre Castéran. A tutorial on [co-] inductive types in coq.

[45] Joseph Y. Halpern, Robert Harper, Neil Immerman, Phokion G. Kolaitis, Moshe Y. Vardi, and Victor Vianu. On the unusual effectiveness of logic in computer science. *Bulletin of Symbolic Logic*, 7(2):213–236, 2001.

[46] William A. Howard. The formulae-as-type notion of construction, 1969. In J. P. Seldin and R. Hindley, editors, *To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–490. Academic Press, New York, 1980.

[47] Chung-Kil Hur, Georg Neis, Derek Dreyer, and Viktor Vafeiadis. The power of parameterization in coinductive proof. In R. Giacobazzi and R. Cousot, editors, *POPL*, pages 193–206. ACM, 2013.

[48] Pierre Hyvernat. The size-change termination principle for constructor based languages. *Logical Methods in CS*, 10(1), 2014.

[49] Pierre Hyvernat. The Size-Change Principle for Mixed Inductive and Coinductive Types. submitted to Logical Methods in CS, 2019.

[50] G Jaber and C Riba. Temporal refinements for guarded recursive types. In *Proc. of ESOP 2021*, volume 12648, pages 548–578, 2021.

[51] B. Jacobs and J.J.M.M. Rutten. An introduction to (co)algebras and (co)induction. In *Advanced Topics in Bisimulation and Coinduction*, pages 38–99. Cambridge University Press, 2011.

[52] Alan Jeffrey. LTL types FRP: linear-time temporal logic propositions as types, proofs as functional reactive programs. In Koen Claessen and Nikhil Swamy, editors, *PLPV*, pages 49–60. ACM, 2012.

[53] Alan Jeffrey. Functional reactive programming with liveness guarantees. In *Proc. of the 18th ICFP*, pages 233–244. ACM, 2013.

[54] Alan Jeffrey. Functional reactive types. In *LICS*, 2014.

[55] Wolfgang Jeltsch. Temporal logic with "until", functional reactive programming with processes, and concrete process categories. In *PLPV*, pages 69–78. ACM, 2013.

[56] Wolfgang Jeltsch. An abstract categorical semantics for functional reactive programming with processes. In Nils Anders Danielsson and Bart Jacobs, editors, *PLPV*, pages 47–58. ACM, 2014.

[57] A. Joyal and I. Moerdijk. *Algebraic set theory*, volume 220 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1995.

[58] Dexter Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[59] Denis Kuperberg, Laureline Pinault, and Damien Pous. Cyclic proofs and jumping automata. In *FSTTCS 2019*, volume 150 of *LIPIcs*, pages 45:1–45:14. Schloss Dagstuhl, 2019.

[60] Denis Kuperberg, Laureline Pinault, and Damien Pous. Cyclic proofs, system t, and the power of contraction. *Proc. ACM Program. Lang.*, 5(POPL):1–28, 2021.

[61] F. W. Lawvere. Metric spaces, generalized logic and closed categories. *Rendiconti del Seminario Matematico e Fisico di Milano*, XLIII:135–166, 1973.

[62] Chin Soon Lee, Neil D. Jones, and Amir Ben-Amram. The size-change principle for program termination. In *Symposium on Principles of Programming Languages*, volume 28, pages 81–92. ACM press, january 2001.

[63] Xavier Leroy. Formal verification of a realistic compiler. *Commun. ACM*, 52(7):107–115, July 2009.

[64] Moritz Lichter and Gert Smolka. Constructive analysis of s1s and büchi automata. *arXiv:1804.04967*, 2018.

[65] Conor McBride. Let's see how things unfold: Reconciling the infinite with the intensional (extended abstract). In Alexander Kurz, Marina Lenisa, and Andrzej Tarlecki, editors, *CALCO*, volume 5728 of *LNCS*, pages 113–126. Springer, 2009.

[66] Paul-André Melliès. Higher-order parity automata. In *ACM/IEEE Symposium on Logic in Computer Science, LICS 2017*, pages 1–12. IEEE Computer Society, 2017.

[67] Marta Z. Kwiatkowska Michael Huth. Quantitative analysis and model checking. *Proc. of LICS conference*, pages 111–122, 1997.

[68] Matteo Mio and Alex Simpson. Łukasiewicz mu-calculus. *Fundam. Inform.*, 150(03-04):317–346, 2017.

[69] Hiroshi Nakano. A modality for recursion. In *IEEE Symposium on Logic in Computer Science*, LICS '00, pages 255–. IEEE, 2000.

[70] Romain Péchoux, Simon Perdrix, Mathys Rennela, and Vladimir Zamdzhiev. Quantum programming with inductive datatypes: Causality and affine type theory. In *FoSSaCS*, volume 12077 of *Lecture Notes in Computer Science*, pages 562–581. Springer, 2020.

[71] Maurice Pouzet and Ivo G. Rosenberg. General metrics and contracting operations. *Discret. Math.*, 130(1-3):103–169, 1994.

[72] Reuben N. S. Rowe and James Brotherston. Automatic cyclic termination proofs for recursive procedures in separation logic. In *Proceedings of CPP-6*, pages 53–65. ACM, 2016.

[73] Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000.

[74] Jan J. M. M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theor. Comput. Sci.*, 308(1-3):1–53, 2003.

[75] Amr Sabry, Benoît Valiron, and Juliana Kaizer Vizzotto. From symmetric pattern-matching to quantum control. In Christel Baier and Ugo Dal Lago, editors, *FOSSACS 2018*, volume 10803 of *LNCS*, pages 348–364. Springer, 2018.

[76] Luigi Santocanale. μ-bicomplete categories and parity games. *ITA*, 36(2):195–227, 2002.

[77] Luigi Santocanale. A calculus of circular proofs and its categorical semantics. In Mogens Nielsen and Uffe Engberg, editors, *FOSSACS '02*, volume 2303 of *LNCS*, pages 357–371. Springer, 2002.

[78] Luigi Santocanale. Free μ-lattices. *Journal of Pure and Applied Algebra*, 168(2–3):227–264, March 2002.

[79] Luigi Santocanale. Completions of μ-algebras. *Annals of Pure and Applied Logic*, 154(1):27–50, May 2008.

[80] Alex Simpson. Cyclic arithmetic is equivalent to peano arithmetic. In Javier Esparza and Andrzej S. Murawski, editors, *FOSSACS 2017*, volume 10203 of *LNCS*, pages 283–300, 2017.

[81] Matthieu Sozeau and Cyprien Mangin. Equations Reloaded: High-Level Dependently-Typed Programming and Proving in Coq. *PACMPL*, 3(ICFP):86–115, 2019.

[82] Paul Taylor. Intuitionistic sets and ordinals. *J. Symbolic Logic*, 61(3):705–744, 1996.

[83] The Coq Development Team. *The Coq Reference Manual (release 8.9.0)*. INRIA, 2019.

[84] Li-yao Xia, Yannick Zakowski, Paul He, Chung-Kil Hur, Gregory Malecha, Benjamin C Pierce, and Steve Zdancewic. Interaction trees: representing recursive and impure programs in coq. *Proceedings of the ACM on Programming Languages*, 4(POPL):1–32, 2019.

[85] Yannick Zakowski, Paul He, Chung-Kil Hur, and Steve Zdancewic. An equational theory for weak bisimulation via generalized parameterized coinduction. In *Proc. of the 9th ACM SIGPLAN International Conf. on Certified Programs and Proofs*, pages 71–84, 2020.

[86] Vladimir Zamdzhiev. Computational adequacy for substructural lambda calculi. In *ACT*, volume 333 of *EPTCS*, pages 322–334, 2020.