Cuts for circular proofs: semantics and cut-elimination

Jérôme Fortier[1] and Luigi Santocanale[2]

1 **LaCIM, UQAM / LIF, AMU**
Montréal, Canada / Marseille, France
jerome.fortier@lif.univ-mrs.fr
2 **LIF, AMU**
Marseille, France
luigi.santocanale@lif.univ-mrs.fr

Abstract

One of the authors introduced in [16] a calculus of circular proofs for studying the computability arising from the following categorical operations: finite products, finite coproducts, initial algebras, final coalgebras. The calculus presented [16] is cut-free; even if sound and complete for provability, it lacked an important property for the semantics of proofs, namely fullness w.r.t. the class of intended categorical models (called μ-bicomplete categories in [18]).

In this paper we fix this problem by adding the cut rule to the calculus and by modifying accordingly the syntactical constraint ensuring soundness of proofs. The enhanced proof system fully represents arrows of the canonical model (a free μ-bicomplete category). We also describe a cut-elimination procedure as a a model of computation arising from the above mentioned categorical operations. The procedure constructs a cut-free proof-tree with possibly infinite branches out of a finite circular proof with cuts.

1998 ACM Subject Classification F.1.1 Models of Computation, F.4.1 Mathematical Logic

Keywords and phrases categorical proof-theory, fixpoints, initial and final (co)algebras, inductive and coinductive types

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.248

1 Introduction

Many researchers have studied fixed-point logics with, explicitly or implicitly, a proof-theoretic approach. Such a spread interest for the proof-theory of these logics stem from different fields of theoretical computer science: model-checking and the logics of computation such as modal μ-calculi [9, 12, 19, 20], logic programming and proof search [1, 3], computer aided verification via proof-assistants and its mathematical counterpart, mainly type theory with inductive and coinductive types [2, 7, 11], coalgebras [14] and categorical programming [6].

The calculus of circular proofs was introduced in [16] with, as main aim, that of lifting from provability to the level of proof-theory the game-theoretic machinery developed in the context of the lattice μ-calculus [17]. From a semantic and algebraic perspective, moving from provability to proof-theory meant sliding the focus from posetal structures to categorical structures; and as far as theoretical computer science is concerned, the reason for taking this step was to investigate fixed-point theory from the point of view of semantics of computation in the style of the Curry-Howard-Lawvere isomorphisms. We aim therefore with the present research at investigating the kind of computability arising from the following categorical operations: finite products and coproducts, initial algebras and final coalgebras. This can be

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca ; pp. 248–262

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

roughly rephrased in type theoretic terms, by saying that we aim at investigating product and sum types, as well as inductive and coinductive types.

In our first proposal [16] we exhibited a cut-free calculus. A circular proof was a finite pointed graph labeled by left or right introduction rules (the rules for additives of linear logic, as well as the fixed point rules, regenerating variables to their bindings) satisfying some constraints on cycles. The cut rule was no part of the calculus and the constraints on cycles were suggested from the theory of parity games in verification. A circular-proof could also be thought as a regular cut-free infinite proof-tree. The proposed calculus enjoyed some nice and non-obvious properties: a sound algebraic semantics and a way to compose two circular proofs into a new circular proof, a construction that could be assimilated to a cut-elimination procedure. However the calculus was not full—a fact of which we were conscious already in [16]—meaning that it was not expressive enough to denote all the arrows in the intended model, a free μ-bicomplete category, cf. [18]. Unfulness could be interpreted as evidence that we could not really dispense of the cut rule.

We fix, with the present work, the lack observed some years ago. We add the cut rule to the set of Gentzen-type rules; such a modification implies rethinking in some coherent way the condition on cycles for being a circular proof. A first main result presented here is the soundness of the proof-system: this amounts to rigorously define the semantics via a theorem asserting the existence and uniqueness of solutions for a certain class of systems of equations. We then prove the fullness of the refined calculus.

A second part of this work provides a cut-elimination procedure for circular proofs. The cut elimination procedure constructs a cut-free, infinite (not necessarily regular), finitely branching proof-tree, thus with infinite branches. This result can be read now as stating that we can actually dispense of the cut rule, but at the cost of giving away regularity of the infinite proof-tree.

2 Preliminaries and notation

Transition systems. A *transition system* is a tuple $G = \langle V, A, \varsigma \rangle$ where V is a set called the *support of* G, A is a set called the *alphabet* and $\varsigma \subseteq V \times A \times V$ is called the *transition relation*. By abuse language, the support V is thereby named G like the transition system itself and its elements are called *vertices*.

Transition systems are often seen as labelled oriented graphs. The notation $u \xrightarrow{a} v$ means $(u, a, v) \in \varsigma$. The *out-degree* of $u \in G$, denoted $\deg(u)$, is the cardinality of the set $\{v \in G : \exists a \in A \text{ such that } u \xrightarrow{a} v\}$. We say that G is *deterministic* if $u \xrightarrow{a} v$ and $u \xrightarrow{a} v'$ implies $v = v'$. In that case, we can also write $v = \varsigma_a u$. If, moreover, $\deg(u) = 1$, then we shall write $u \to v$ and $v = \varsigma u$ without ambiguity. Paths and cycles are defined in the obvious way and the composition of two paths Γ_0, Γ_1, when defined, is denoted $\Gamma_0 \cdot \Gamma_1$. For $u \in G$, let V_u denote the set of all targets of some path from u in G and let $\overline{G, u} = \langle V_u, A, \varsigma_{\restriction V_u \times A \times V_u} \rangle$. $\overline{G, u}$ is called the *reachable graph* from u.

Categories. The reader might consult [10] for basic notions about categories. For $f : a \to b$ and $g : b \to c$ arrows of some category, we shall mostly use $f \cdot g$ to denote their composition; notice however that, with respect to usual notation, we have $f \cdot g = g \circ f$.

3 The calculus of circular proofs

Terms are constructed from a fixed set of variables $\mathbb{V}$ using the binary function symbols $\times, +$ and the constants $1, 0$; the set of terms will be denoted by TERMS. The set of

Identity, Cut, Assumption	$\dfrac{}{t \vdash t}\,\mathtt{Id}$ $\qquad$ $\dfrac{s \vdash u \quad u \vdash t}{s \vdash t}\,\mathtt{Cut}$ $\qquad$ $\dfrac{}{s \vdash t}\,\mathtt{A}$	
	$\mathfrak{L}$	$\mathfrak{R}$
Products	$\dfrac{s_i \vdash t}{s_0 \times s_1 \vdash t}\,\mathtt{L}\times_i \;\; i = 0,1$	$\dfrac{}{t \vdash 1}\,\mathtt{RAx}$ $\dfrac{s \vdash t_0 \quad s \vdash t_1}{s \vdash t_0 \times t_1}\,\mathtt{R}\times$
Coproducts	$\dfrac{}{0 \vdash t}\,\mathtt{LAx}$ $\dfrac{s_0 \vdash t \quad s_1 \vdash t}{s_0 + s_1 \vdash t}\,\mathtt{L}+$	$\dfrac{s \vdash t_i}{s \vdash t_0 + t_1}\,\mathtt{R}+_i \;\; i = 0,1$
Fixpoints	$\dfrac{\tau(x) \vdash t}{x \vdash t}\,\mathtt{L}\mu x$ $\dfrac{\tau(x) \vdash t}{x \vdash t}\,\mathtt{L}\nu x$	$\dfrac{s \vdash \tau(x)}{s \vdash x}\,\mathtt{R}\mu x$ $\dfrac{s \vdash \tau(x)}{s \vdash x}\,\mathtt{R}\nu x$

Figure 1 Inference rules over a system S.

subterms of (resp. variables appearing in) a term t will be denoted $\mathsf{ST}(t)$ (resp. $\mathsf{VAR}(t)$). A *directed systems of equations* is a tuple $S = \langle X, \tau, \pi \rangle$, where X is a finite subset of $\mathbb{V}$, $\tau : X \to \mathsf{TERMS}$, and $\pi : X \to \mathbb{N}$. For such a system S, we let $\mathsf{BD}(S) := X$ and $\mathsf{FV}(S) := \bigcup_{x \in X} \mathsf{VAR}(\tau(x)) \setminus \mathsf{BD}(S)$.

Intuitively, we think of the tuple S as the system of equations $\{\, x =_{\theta(\pi(x))} \tau(x) \mid x \in X \,\}$ where $\theta(n) = \mu$ (least solution) if n is odd and $\theta(n) = \nu$ (greatest solution) otherwise. The priority function π shall also specify the order by which we solve this system of equations (cf. Section 4).

A sequent is here a pair (s, t) of terms. As usual we shall use the turnstile symbol, that is we write the sequent as $s \vdash t$, to separate its left part s from its right part t. We shall use SEQ to denote the set of sequents. For a fixed directed system of equations S, the *inference rules* over S are (instances of) the formal expressions appearing in Figure 1. Notice that the rules in the column $\mathfrak{L}$ act on the left part of a sequent while those $\mathfrak{R}$ act on the right. Accordingly, we group (labels of) the rules in two families $\mathfrak{L}$ and $\mathfrak{R}$ and set $\Sigma := \mathfrak{L} \cup \mathfrak{R} \cup \{\mathtt{Id}, \mathtt{Cut}, \mathtt{A}\}$.

▶ **Definition 1.** A *pre-proof* over S is a tuple $\Pi = \langle G, \rho, \sigma \rangle$ where G is a deterministic transition system over the alphabet $\{0, 1\}$, $\rho : G \to \Sigma$, and $\sigma = (\sigma_{\mathtt{L}}, \sigma_{\mathtt{R}}) : G \to \mathsf{SEQ}$; moreover, for each $v \in G$, $\deg(v) \leq 2$ and the expression (1) is an inference rule over S.

$$\frac{\sigma(\varsigma_0 v) \quad \dots \quad \sigma(\varsigma_{\deg(v)-1} v)}{\sigma(v)}\,\rho(v) \tag{1}$$

In order to be called a *proof*, a pre-proof must satisfy an additional syntactic constraint. The constraint is originally inspired by the theory of parity games: it actually codes, in a proof-theoretic setting, the winning condition on infinite paths. Scientists with a background in type theory might perceive the similarity with the *productivity* constraint of [7, §2.3]. The syntactic constraint turns out to be the key ingredient to ensure soundness of the proof system and local termination of the cut elimination procedure.

Let $\Pi = \langle G, \rho, \sigma \rangle$ be a pre-proof. We say that a path Γ in G is *left-traceable* if, for all n, if $\rho(\Gamma(n)) = \mathtt{Cut}$, then $\Gamma(n+1) = \varsigma_0(\Gamma(n))$. Similarly, Γ is *right-traceable* if, for all n, if $\rho(\Gamma(n)) = \mathtt{Cut}$, then $\Gamma(n+1) = \varsigma_1(\Gamma(n))$. We say that Γ has a *left μ-trace* if Γ is left-traceable, it contains a left fixpoint rule, and the highest priority of its left fixpoint rules is odd. Similarly, we say that Γ has a *right ν-trace* if Γ is right-traceable, it contains a right fixpoint rule, and the highest priority of its right fixpoint rules is even.

▶ **Condition 2** (The Guard condition on cycles)**.** Every cycle in G either has a left μ-trace or a right ν-trace.

▶ **Condition 3** (The Guard condition on infinite paths)**.** Every infinite path Γ in G can be written $\Gamma = \Gamma_0 \cdot \Gamma_1$ where Γ_0 is finite, Γ_1 either has a left μ-trace or a right ν-trace and every fixpoint rule in Γ_1 occurs infinitely often.

It is easily seen that if G is a finite graph, then conditions 2 and 3 are equivalent.

▶ **Definition 4.** A *circular proof* is a pre-proof $\Pi = \langle G, \rho, \sigma \rangle$ satisfying the guard conditions, with G a finite graph.

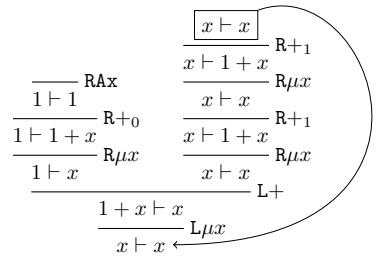
The assumption rule $\mathtt{A}$ is a technical tool that is needed to prove soundness of the system—the reader might have noticed that any sequent can be justified using this rule. Therfore, let $A_\Pi := \{v \in G : \rho(v) = \mathtt{A}\}$ and $C_\Pi := G \setminus A_\Pi$: A_Π is the set of assumptions of Π, while C_Π is the set of its conclusions. A circular proof is *ground* if $A_\Pi = \varnothing$. Even if we often draw a circular proof in the form of a tree with back-edges having a specified root (cf. Figure 3), we consider all the vertexes of the proof as potential conclusions. On the other hand, we can also easily define circular proofs with a root: a *rooted circular proof* is a pair $\langle \Pi, v \rangle$ where Π is a ground circular proof and $v \in G$.

4 Semantics of the calculus

The intended use of circular proofs is to describe functions between (possibly nested) inductive and coinductive types. Before going into the technical details, let us give a few examples.

Recall that a *natural numbers object* is the object part of an initial algebra of the functor $F(x) = 1 + x$ (such an object can be seen as the least categorical solution of the functorial equation $x = 1 + x$); of course, in Set, such an initial algebra is given by the usual data, $1 + \mathbb{N} \xrightarrow{\{0,\mathtt{suc}\}} \mathbb{N}$. The function $\mathtt{double} : \mathbb{N} \to \mathbb{N}$ that sends n to $2n$ is represented as the root of the proof in Figure 2. The $\mathtt{L}+$ instruction can be understood as destructively reading an input and branching according to its constructor (0 or $\mathtt{suc}$). The right rules, on the other hand, represent the choices of constructors for the output. The back edge marks the recursive call of the function to itself.

The fact that reading the input is destructive is a problem that can be partially dismissed with the cut rule. For instance, the interpretation of the circular proof in Figure 3 in Set is the diagonal mapping $\Delta : \mathbb{N} \to \mathbb{N}^2$ defined by $\Delta(n) = (n, n)$. It was constructed by the method given below (see *Fullness* and Figure 5) since Δ is the initial algebra morphism to the algebra $\{(0,0), \mathtt{suc} \times \mathtt{suc}\} : 1 + \mathbb{N}^2 \to \mathbb{N}^2$. It was mentioned in [15] that there is no cut-free circular proof with this interpretation.



$$\texttt{double}(n) = \begin{cases} 0 \text{ if } n = 0; \\ \texttt{suc}(\texttt{suc}(\texttt{double}(n'))) \text{ if } n = \texttt{suc}(n'). \end{cases}$$

Figure 2 Rooted circular proof denoting the function $\texttt{double} : \mathbb{N} \to \mathbb{N}$.

$$S := \left\{ \begin{array}{l} y =_1 x + y \\ x =_1 1 + x \end{array} \right\}$$

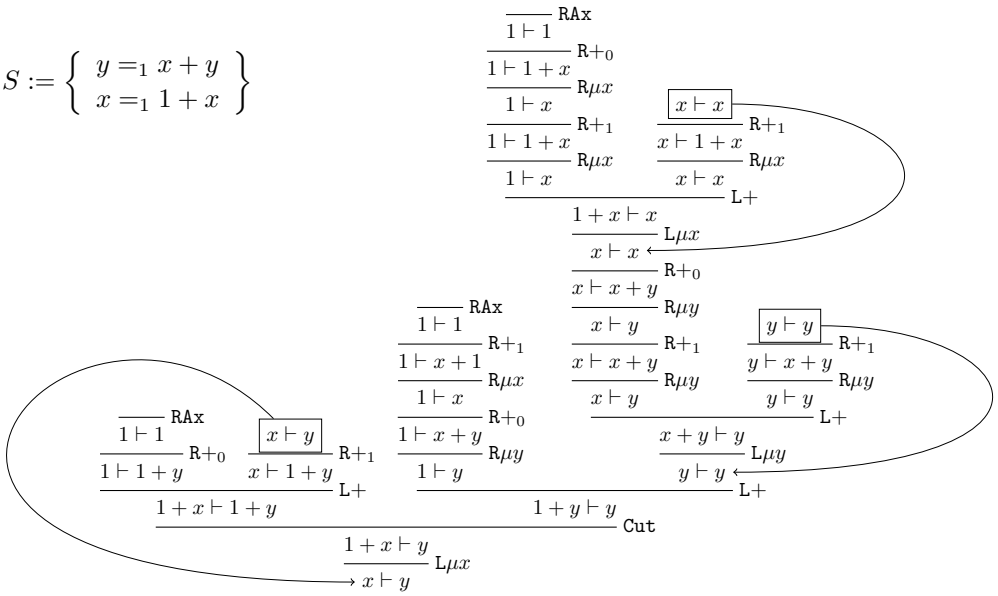


Figure 3 Rooted circular proof denoting the diagonal mapping $\Delta : \mathbb{N} \to \mathbb{N}^2$.

Interpreting the terms. μ-bicomplete categories were defined in previous work on the subject, mainly in [18]. In this paper it was also argued about the equivalence among possible definitions of this notion, via a scalar μ-calculus or via a vectorial one. Roughly speaking μ-bicomplete categories are categories with finite products and finite coproducts (i.e. bicartesian categories) with enough initial algebras and final coalgebras to solve directed systems of equations. Examples of μ-bicomplete categories include the locally presentable categories such as the category of sets, categories of algebras for a finitary signature, and categories of presheaves and sheaves.

Let $\mathcal{M}$ be a μ-bicomplete category and S be a directed system of equations. The definition of the semantics is achieved in three steps: first we define the obvious functorial semantics of terms; then we define the semantics of S as a canonical solution to the system of equations it represents; finally, we evalute terms in $\mathcal{M}$ by means of the bound variables of S.

Given $t \in \mathsf{TERMS}$ and a finite subset X with $\mathsf{VAR}(t) \subseteq X$, the *semantics of* t, denoted $|t|_X$, is a functor from $\mathcal{M}^X$ to $\mathcal{M}$. Let us denote by $\mathcal{M}_X$ the category of functors from $\mathcal{M}^X$ to $\mathcal{M}$ with natural transformations as arrows. Recall that this is a bicartesian category, limits being computed pointwise, see [10, Chapter V]. The formal definition of $|t|_X$ is by induction on the structure of t as follows:

- if $t = x \in X$, then $|x|_X$ is the projection functor on the x component (thus $|x|_X = \mathtt{pr}_x^X$);
- $|0|_X$ (resp. $|1|_X$) is the initial (resp. terminal) object in the category $\mathcal{M}_X$;

- if $t = t_1 \times t_2$ (resp. $t = t_1 + t_2$), then $|t|_X$ is the product $|t_1|_X \times |t_2|_X$ (resp. coproduct $|t_1|_X + |t_2|_X$) in the category $\mathcal{M}_X$.

Given $n \geq 0$ and a system S, let $X_n = \{ x \in \mathsf{BD}(S) \mid \pi(x) \leq n \}$ and let S_n be the restriction of S to X_n, namely $S_n = \langle X_n, \tau_{\restriction X_n}, \pi_{\restriction X_n} \rangle$. In particular, if $M = \max\{ \pi(x) \mid x \in \mathsf{BD}(S) \}$, then we define $\mathsf{MAX}(S) := \{ x \in \mathsf{BD}(S) \mid \pi(x) = M \}$, $\mathsf{LOW}(S) := X_{M-1}$, and let $P(S)$, the *predecessor system*, be S_{M-1}.

Assuming that X is finite and that $\mathsf{FV}(S) \subseteq X$ and $\mathsf{BD}(S) \cap X = \varnothing$, the *semantics of* S, noted by $[\![S]\!]_X$, is a functor from $\mathcal{M}^X$ to $\mathcal{M}^{\mathsf{BD}(S)}$. The definition is as follows:

▶ **Definition 5.** If $\mathsf{BD}(S) = \varnothing$, then $\mathcal{M}^{\mathsf{BD}(S)}$ is the terminal category (with just one object and its identity arrow), so that we let $[\![S]\!]_X$ be the unique functor from $\mathcal{M}^X$ to the terminal category. Otherwise, consider the predecessor system $P(S)$ and observe that $\mathsf{FV}(P(S)) \subseteq \mathsf{MAX}(S) \cup \mathsf{FV}(S) \subseteq \mathsf{MAX}(S) \cup X$ and $(\mathsf{MAX}(S) \cup X) \cap \mathsf{BD}(P(S)) = \varnothing$; as $\mathrm{card}(\mathsf{BD}(P(S))) < \mathrm{card}(\mathsf{BD}(S))$, $[\![P(S)]\!]_{X \cup \mathsf{MAX}(S)}$ is defined as a functor from $\mathcal{M}^{X \cup \mathsf{MAX}(S)}$ to $\mathcal{M}^{\mathsf{BD}(P(S))}$. Let G and H be the functors so defined:

$$G := \langle |\tau(x)|^{\mathsf{BD}(S) \cup X} \mid x \in \mathsf{MAX}(S) \rangle : \mathcal{M}^{\mathsf{BD}(S) \cup X} \to \mathcal{M}^{\mathsf{MAX}(S)},$$

$$H := \langle\, G\,,\ [\![P(S)]\!]_{\mathsf{MAX}(S) \cup X} \circ \mathtt{pr}^{\mathsf{BD}(S) \cup X}_{\mathsf{MAX}(S) \cup X} \,\rangle :$$

$$\mathcal{M}^{\mathsf{BD}(S)} \times \mathcal{M}^X = \mathcal{M}^{\mathsf{BD}(S) \cup X} \longrightarrow \mathcal{M}^{\mathsf{MAX}(S)} \times \mathcal{M}^{\mathsf{BD}(P(S))} = \mathcal{M}^{\mathsf{BD}(S)}.$$

If $\pi(\mathsf{MAX}(S))$ is odd, then we let $[\![S]\!]_X$ be the parametrized initial algebra of H; and if $\pi(\mathsf{MAX}(S))$ is even, then we let $[\![S]\!]_X$ be the parametrized final coalgebra of H.

▶ Remark. The existence of an initial algebra and of a final coalgebra in the previous definition is ensured by the assumption that $\mathcal{M}$ is a μ-bicomplete category.

Finally, given a system S, a term t, and a finite subset X with $\mathsf{FV}(S) \cup (\mathsf{VAR}(t) \setminus \mathsf{BD}(S)) \subseteq X$, the *value of* t *w.r.t.* S, noted $[\![t]\!]_X$, is the functor from $\mathcal{M}^X$ to $\mathcal{M}$ defined below:

$$[\![t]\!]_X := \left(\mathcal{M}^X \xrightarrow{\langle \mathtt{id}, [\![S]\!]_X \rangle} \mathcal{M}^X \times \mathcal{M}^{\mathsf{BD}(S)} = \mathcal{M}^{X \cup \mathsf{BD}(S)} \xrightarrow{|t|_{X \cup \mathsf{BD}(S)}} \mathcal{M} \right).$$

We leave the reader to verify that if $X \subseteq Y$, then $|t|_Y$ (resp. $[\![S]\!]_Y$, $[\![t]\!]_Y$) is obtained from $|t|_X$ (resp. $[\![S]\!]_X$, $[\![t]\!]_X$) by precomposing the latter with the projection from $\mathcal{M}^Y$ to $\mathcal{M}^X$. The previous observation allows us to be sloppy with the notation and to omit the subscript X, which shall be understood from the context as the least set of variables satisfying some required constraints. For example, if we are considering a set of terms $E = \{ t_1, \ldots, t_n \}$ with $t \in E$, then we shall have $[\![t]\!] := [\![t]\!]_X$ with $X = \mathsf{FV}(S) \cup (\bigcup_{i=1,\ldots,n} \mathsf{VAR}(t_i) \setminus \mathsf{BD}(S))$. It might be necessary, on the other hand, to evaluate a term with respect to different systems S and T; we shall then write the system in superscript, so to have $[\![t]\!]^S$ and $[\![t]\!]^T$.

Let $M = \pi(\mathsf{MAX}(S))$; let us remark that if M is odd, then for all $x \in \mathsf{BD}(S)$ there is (by definition of $[\![S]\!]$) a canonical invertible arrow $\zeta_x : [\![\tau(x)]\!] \to [\![x]\!]$; however, if $\pi(x) < M$, then we can assume that ζ_x is the identity while $[\![x]\!](Y) = [\![x]\!]^{S \restriction \pi(x)}([\![Z]\!], Y)$ where $Y = \mathsf{FV}(S)$ and $Z = \{ z \in \mathsf{BD}(S) \mid \pi(z) > \pi(z) \}$ (see Proposition 2.2 in [18] with $F := [\![P(S)]\!]$, $G := G$, $\mathsf{C} := \mathcal{M}^{MAX(S)}$, and $\mathsf{D} := \mathcal{M}^{\mathsf{LOW}(S)}$). Similarly, if $x \in \mathsf{BD}(S)$ and $\pi(\mathsf{MAX}(S))$ is odd, then there exists a canonical invertible arrow $\xi_x : [\![x]\!] \to [\![\tau(x)]\!]$; if $\pi(x) < M$, then we can assume that ξ_x is the identity and that $[\![x]\!](Y) = [\![x]\!]^{S \restriction \pi(x)}([\![Z]\!], Y)$. An easy induction shall therefore prove the following statement:

▶ **Proposition 6.** *For each $x \in \mathsf{BD}(S)$, if $\pi(x)$ is odd, then there exists a canonical invertible arrow $\zeta_x : [\![\tau(x)]\!] \to [\![x]\!]$; if $\pi(x)$ is even, then there exists a canonical invertible arrow $\xi_x : [\![x]\!] \to [\![\tau(x)]\!]$.*

Identity, Cut	$\dfrac{}{[\![t]\!] \xrightarrow{id_{[\![t]\!]}} [\![t]\!]}\,\mathtt{Id}$ $\qquad$ $\dfrac{[\![s]\!] \xrightarrow{f} [\![u]\!] \quad [\![u]\!] \xrightarrow{g} [\![t]\!]}{[\![s]\!] \xrightarrow{f\cdot g} [\![t]\!]}\,\mathtt{Cut}$	
Products	$\dfrac{[\![s_i]\!] \xrightarrow{f} [\![t]\!]}{[\![s_0 \times s_1]\!] \xrightarrow{\mathtt{pr}_i\cdot f} [\![t]\!]}\,\mathtt{L}\times_i \ i=0,1$	$\dfrac{}{[\![t]\!] \xrightarrow{!_{[\![t]\!]}} [\![1]\!]}\,\mathtt{RAx}$ $\quad$ $\dfrac{[\![s]\!] \xrightarrow{f} [\![t_0]\!] \quad [\![s]\!] \xrightarrow{g} [\![t_1]\!]}{[\![s]\!] \xrightarrow{\langle f,g\rangle} [\![t_0 \times t_1]\!]}\,\mathtt{R}\times$
Coproducts	$\dfrac{}{[\![0]\!] \xrightarrow{?_{[\![t]\!]}} [\![t]\!]}\,\mathtt{LAx}$ $\quad$ $\dfrac{[\![s_0]\!] \xrightarrow{f} [\![t]\!] \quad [\![s_1]\!] \xrightarrow{g} [\![t]\!]}{[\![s_0 + s_1]\!] \xrightarrow{\{f,g\}} [\![t]\!]}\,\mathtt{L}+$	$\dfrac{[\![s]\!] \xrightarrow{f} [\![t_i]\!]}{[\![s]\!] \xrightarrow{f\cdot \mathtt{in}_i} [\![t_0 + t_1]\!]}\,\mathtt{R}+_i \ i=0,1$
Fixpoints	$\dfrac{[\![\tau(x)]\!] \xrightarrow{f} [\![t]\!]}{[\![x]\!] \xrightarrow{\zeta_x^{-1}\cdot f} [\![t]\!]}\,\mathtt{L}\mu x$ $\quad$ $\dfrac{[\![\tau(x)]\!] \xrightarrow{f} [\![t]\!]}{[\![x]\!] \xrightarrow{\xi_x\cdot f} [\![t]\!]}\,\mathtt{L}\nu x$	$\dfrac{[\![s]\!] \xrightarrow{f} [\![\tau(x)]\!]}{[\![s]\!] \xrightarrow{f\cdot\zeta_x} [\![x]\!]}\,\mathtt{R}\mu x$ $\quad$ $\dfrac{[\![s]\!] \xrightarrow{f} [\![\tau(x)]\!]}{[\![s]\!] \xrightarrow{f\cdot\xi_x^{-1}} [\![x]\!]}\,\mathtt{R}\nu x$

Figure 4 Semantics of rules.

Interpreting the rules. From now on our goal shall be that of associating to a circular proof Π over a system S its semantics; this shall be a collection of arrows (one for each conclusion of Π) in some μ-bicomplete category. If Π is ground and does not have cycles, then this task is easily achieved using induction; namely, we start from the leaves and, by interpreting the rules of the calculus as specifying how to construct new arrows from given ones via the categorical operations, we build more complex arrows. Such interpretation of rules is given in Figure 4. For each vertex $v \in \Pi$ with $\sigma(v) = s \vdash t$, the construcion gives an arrow f_v from $[\![s]\!]_X$ to $[\![t]\!]_X$ in the category $\mathcal{M}_X$; that is, f_v is a natural transformation from $[\![s]\!]_X$ to the functor $[\![t]\!]_X$.

Thus, if we write $\mathcal{M}_X(F,G)$ for the set of arrows from F to G in the category $\mathcal{M}_X$, each rule `Rule`, with assumptions $s_i \vdash t_i$ and conclusion $s \vdash t$, can be intertpreted as a function from $\prod_{i=1,\dots,n} \mathcal{M}_X([\![s_i]\!],[\![t_i]\!])$ to $\mathcal{M}_X([\![s]\!],[\![t]\!])$. We notice, however, that for $F, G \in \mathcal{M}_X$, the similar expression $\mathcal{M}(F,G)$ denotes the following functor:

$$(\mathcal{M}^X)^{op} \times \mathcal{M}^X \xrightarrow{F^{op}\times G} \mathcal{M}^{op} \times \mathcal{M} \xrightarrow{\mathcal{M}(\_,\_)} \mathrm{Set}\,.$$

With exception of `Id` and `Cut`, all the rules can also be interpreted as defining natural transformations of these generalized hom-functors:

$$\lceil \mathtt{Rule} \rceil_{X,X'} : \prod_{i=1,\dots,n} \mathcal{M}([\![s_i]\!],[\![t_i]\!]) \to \mathcal{M}([\![s]\!],[\![t]\!]) : (\mathcal{M}^X)^{op} \times \mathcal{M}^X \to \mathrm{Set}\,. \tag{2}$$

The following Lemma relates the two possible semantical interpretations of rules.

▶ **Lemma 7.** *Let $\alpha : \prod_{i=1,\dots,n} \mathcal{M}(F_i, G_i) \to \mathcal{M}(F, G)$ be a natural transformation and suppose that, for each $i = 1, \dots n$, we are given a natural transformation $\beta^i \in \mathcal{M}_X(F_i, G_i)$. Then the collection of arrows $\alpha_{c,c}(\beta^1_c, \dots, \beta^n_c) : Fc \to Gc$, c an object of $\mathcal{M}^X$, is a natural transformation from F to G.*

We notice next that even the cut rule can be given a semantics as a natural transformation of hom-functors. Namely, *given a natural transformation $\beta : [\![u]\!] \to [\![t]\!]$*, we can define the semantics of a cut as the natural transformation

$$\lceil \mathtt{Cut}, \beta \rceil : \mathcal{M}([\![s]\!], [\![u]\!]) \to \mathcal{M}([\![s]\!], [\![t]\!]) \tag{3}$$

sending $f : [\![s]\!](c) \to [\![u]\!](d)$ to $f \cdot \beta_d : [\![s]\!](c) \to [\![t]\!](d)$. Similarly, given $\gamma : [\![s]\!] \to [\![s]\!]$, we can define the semantics of a cut as follows:

$$\lceil \gamma, \mathtt{Cut} \rceil : \mathcal{M}([\![u]\!], [\![t]\!]) \to \mathcal{M}([\![s]\!], [\![t]\!])\,, \qquad \lceil \gamma, \mathtt{Cut} \rceil (f) = \gamma_c \cdot f : [\![s]\!](c) \to [\![t]\!](d)\,. \tag{4}$$

Interpreting some derived inference rules. Motivated by the previous observations about the semantics of certain rules, we shall make sense of a large collection of circular-proofs as derived inference rules whose interpretation is a natural transformation between hom-set functors.

▶ **Definition 8.** A circular proof Π is *homogeneous* if it does not contain the rule $\mathtt{Id}$ and, for each $v \in \Pi$ with $\rho(v) = \mathtt{Cut}$, at least one among $\varsigma_0 v$ and $\varsigma_1 v$ is an assumption of Π.

It was argued in [16] that we can associate to an identity-free and cut-free circular proof Π a system of equations $\lceil \Pi \rceil$; it was then shown that Π has a unique solution $\lceil \Pi \rceil_\dagger$, thus defining the semantics of Π via this unique solution. We generalize here this result to homogeneous circular proofs. For each $v \in \Pi$ with $\rho(v) = \mathtt{Cut}$, let $\chi(v) \in \{ 0, 1 \}$ such that $\varsigma_{\chi(v)} v \in A_\Pi$; let therefore $A^{\mathtt{c}}_\Pi := \{ \varsigma_{\chi(v)} v \in A_\Pi \mid \rho(v) = \mathtt{Cut} \}$ and $A^{\mathtt{s}}_\Pi := A_\Pi \setminus A^{\mathtt{c}}_\Pi$ (w.l.o.g., we assume that if $v \in A_\Pi$, then v has just one predecessor in G). Given a collection of natural transformations $\beta = \{ \beta^v : [\![\sigma_{\mathtt{L}}(v)]\!] \to [\![\sigma_{\mathtt{R}}(v)]\!] \mid v \in A^{\mathtt{c}}_\Pi \}$, the system $\lceil \Pi_\beta \rceil$ is defined as

$$\lceil \Pi_\beta \rceil := \left\{ \; v \; = \; \lceil \rho(v)_\beta \rceil (C_\Pi, A^{\mathtt{s}}_\Pi) \; \right\}_{v \in C_\Pi}\,. \tag{5}$$

In the definition of $\lceil \Pi_\beta \rceil$ above, if $\rho(v) \neq \mathtt{Cut}$, then $\lceil \rho(v)_\beta \rceil := \lceil \rho(v) \rceil$ is as in (2); if $\rho(v) = \mathtt{Cut}$, then: if $\varsigma_1 v \in A_\Pi$, then $\lceil \rho(v)_\beta \rceil = \lceil \mathtt{Cut}, \beta_{\varsigma_1 v} \rceil$ as in (3); if $\varsigma_0 v \in A_\Pi$, then $\lceil \rho(v)_\beta \rceil = \lceil \beta_{\varsigma_0 v}, \mathtt{Cut} \rceil$ as in (4). Furthermore the defining equation (5) emphasizes that each $\lceil \rho(v)_\beta \rceil$ depends on two kinds of variables, those coming from C_Π and those coming from $A^{\mathtt{s}}_\Pi$. Therefore, the system has the conclusions of Π as bound variables and depends on parameters coming from $A^{\mathtt{s}}_\Pi$. We identify such a system with the natural transformation

$$\lceil \Pi_\beta \rceil : \prod_{v \in C_\Pi} \mathcal{M}([\![\sigma_{\mathtt{L}}(v)]\!], [\![\sigma_{\mathtt{R}}(v)]\!]) \times \prod_{v \in A^{\mathtt{s}}_\Pi} \mathcal{M}([\![\sigma_{\mathtt{L}}(v)]\!], [\![\sigma_{\mathtt{R}}(v)]\!]) \to \prod_{v \in C_\Pi} \mathcal{M}([\![\sigma_{\mathtt{L}}(v)]\!], [\![\sigma_{\mathtt{R}}(v)]\!])\,,$$

which is an arrow of the category of functors from $(\mathcal{M}^{op})^X \times \mathcal{M}^X$ to Set. Notice that, to a certain degree, we are abusing of language, as some circular proof might be considered to be over different systems S and T. If it we need to specify the system S, we can write the more explicit $\lceil \Pi \rceil^S$ (and $\lceil \Pi \rceil^S_\dagger$) in place of $\lceil \Pi \rceil$.

▶ **Theorem 9.** *For each homogeneous circular proof* Π *and each collection of natural transformations* $\{\, \beta_v : [\![\sigma_{\mathtt{L}}(v)]\!] \to [\![\sigma_{\mathtt{R}}(v)]\!] \mid v \in A^{\mathtt{c}}_{\Pi} \,\}$*, the system* $\lceil \Pi_\beta \rceil$ *admits a unique natural solution*

$$\lceil \Pi_\beta \rceil_\dagger : \prod_{v \in A^{\mathtt{s}}_{\Pi}} \mathcal{M}([\![\sigma_{\mathtt{L}}(v)]\!], [\![\sigma_{\mathtt{R}}(v)]\!]) \longrightarrow \prod_{v \in C_\Pi} \mathcal{M}([\![\sigma_{\mathtt{L}}(v)]\!], [\![\sigma_{\mathtt{R}}(v)]\!])\,.$$

The proof of the Theorem mostly depends on the Bekić Lemma, as well as on the following kind of categorical fixed point Lemma. The Lemma can be understood as giving a categorical interpretation to Mendler's style recursion, see [11, §2].

▶ **Lemma 10.** *Let* $\mathsf{W}, \mathsf{C}, \mathsf{D}$ *be three categories, of which* C *has products; let* $F : \mathsf{C} \times \mathsf{W} \to \mathsf{C}$*,* $G : \mathsf{D} \to \mathsf{C}$*,* $Q : \mathsf{C}^{op} \times \mathsf{W}^{op} \times \mathsf{D} \to \mathrm{Set}$ *be functors; let* $\zeta_w : F(\boldsymbol{x}_w, w) \to \boldsymbol{x}_w$ *be a parametrized initial algebra of* F*. Consider an arbitrary natural transformation*

$$\alpha : \mathsf{C}(\_, G) \times Q \to \mathsf{C}(F, G) : \mathsf{C}^{op} \times \mathsf{W}^{op} \times \mathsf{D} \to \mathrm{Set}\,.$$

For each $w \in W$*,* $d \in \mathsf{D}$ *and* $q \in Q(\boldsymbol{x}_w, w, d)$*, there exists a unique* $f_{w,d} : \boldsymbol{x}_w \to Gd$ *which is a solution of the equation*

$$f = \zeta_w^{-1} \cdot \alpha_{\boldsymbol{x}_w, w, d}(f, q)\,.$$

Moreover, the map sending $q \in Q(\boldsymbol{x}_w, w, d)$ *to* $f_{w,d} \in \mathsf{C}(F(\boldsymbol{x}_w, w), Gd)$ *is natural in* w *and* d*.*

Semantics of rooted circular proofs (soundness). Let $\langle \Pi, v \rangle$ be a rooted circular proof with $\sigma(v) = s \vdash t$; we can now define $[\![\Pi, v]\!] : [\![s]\!] \to [\![t]\!]$, the *natural transformation interpreting* $\langle \Pi, v \rangle$ (with respect to a system S), by induction, almost as usual. The induction is now on the well-founded structure of maximal strongly connected components of the underlying graph of Π. The key observation is that if $\mathcal{C}$ is such a non trivial component of Π (i.e. there exists $v, u \in \mathcal{C}$ and a non-null path from v to u), then the restriction of Π to $\mathcal{C}$ can be made into an homogeneous circular proof. More formally, we can define $\Pi \restriction \mathcal{C}$ by choosing $v_0 \in \mathcal{C}$ and putting

$$A_{\mathcal{C}} := \{\, \varsigma_i v \mid v \in \mathcal{C}, \varsigma_i v \notin \mathcal{C} \,\}\,, \qquad \Pi \restriction \mathcal{C} := \overline{\overline{\Pi, v_0}^{A_{\mathcal{C}}}, v_0}\,.$$

The inductive definition is as follows. If v belongs to a trivial component, then we can define the semantics of $\langle \Pi, v \rangle$ as in the non-circular case. Otherwise, we dispose by induction of two collections $\beta = \{\, [\![\Pi, v]\!] \mid v \in A^{\mathtt{c}}_{\Pi \restriction \mathcal{C}} \,\}$ and $\gamma = \{\, [\![\Pi, v]\!] \mid v \in A^{\mathtt{s}}_{\Pi \restriction \mathcal{C}} \,\}$; by Theorem 9 we dispose of a natural transformation $\lceil (\Pi \restriction \mathcal{C})_\beta \rceil_\dagger$ between the appropriate hom-functors; Lemma 7 ensure that we can pointwise apply $\lceil (\Pi \restriction \mathcal{C})_\beta \rceil_\dagger$ to the natural transformations in γ to obtain a new collection of natural transformations indexed by elements of $\mathcal{C}$. We define therefore

$$[\![\Pi, v]\!] := \big(\lceil (\Pi \restriction \mathcal{C})_\beta \rceil_\dagger(\gamma) \big) \cdot \mathtt{pr}_v\,.$$

Fullness. We show that this interpretation of rooted circular proods is *full*. That means that if $\mathcal{M}$ is a free μ-bicomplete category over a set of generators, then, for every arrow f of $\mathcal{M}$, there is a rooted circular proof $\langle \Pi, v \rangle$ such that $[\![\Pi, v]\!] = f$.

The proof of this fact is a lengthy induction. However, the only non-trivial case is the closure of the class of definable arrows under canonical maps from initial algebras (and dually, behaviour maps to final coalgebras); let us exemplify this point. Suppose S is a system with just one bound variable x of maximal priority; if this priority is odd, then

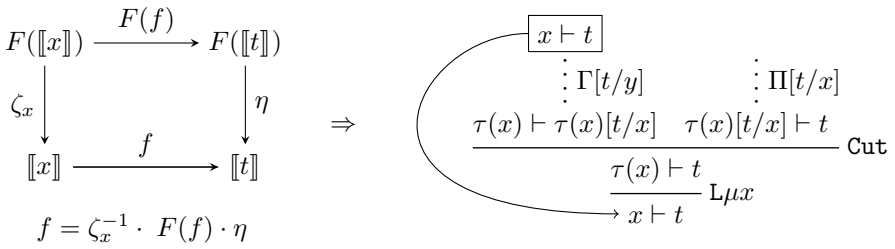


Figure 5 Construction of maps from the initial algebra.

$F = [\![\tau(x)]\!]^{P(S)} : \mathcal{M} \to \mathcal{M}$ while $[\![x]\!]^S$ is the initial F-algebra. Suppose now that we are given a system T which contains an exact copy of $P(S)$, with the exception that the variable x is bound to some term t. If we dispose of a rooted circular proof $\langle \Pi, v\rangle$ on T with $\sigma(v) = \tau(x) \vdash t$, then $[\![\tau(x)]\!]^T = [\![\tau(x)]\!]^{P(S)}([\![t]\!]^T) = F([\![t]\!]^T)$, that is $\eta := [\![\Pi, v]\!] : F([\![t]\!]^T) \to [\![t]\!]^T$ is an F-algebra. We notice that the canonical natural transformation $F_{x,y} : \mathcal{M}(x,y) \to \mathcal{M}(Fx, Fy)$ is definable via a cut-free circular proof Γ (using a language of game theory, Γ is the copycat strategy). Let T' be a system which is a disjoint copy of S and T, with the variable x of T being renamed to y. The circular proof on the right of Figure 5, on the system T', shall then denote the unique algebra morphism from the initial one.

5 Cut elimination

We give in this section an algorithm that, given as input a pointed circular proof $\langle \Pi, v\rangle$, outputs a cut-free pre-proof with possibly infinite branches (yet, a finitely branching tree). A refinement of the technique used to prove Theorem 12 below can also be used to prove that the output tree satisfies the Guard condition 3. This justifies saying that the output tree is an *infinite proof-tree*.

Just like in the classical case for Gentzen's system (see [8], for instance) the procedure consists in "pushing" every cut away from the root. However, in our case, the output tree must be computed with a lazy (outermost) rather than eager (innermost) strategy; this is because not every path in Π leads to a leaf, so that we have to eliminate cuts by performing a breadth-first search of Π from the root. A problem that arises by using this strategy is that we might need to permute a cut with another cut. We temporarily dismiss the problem by merging consecutive cuts together into a sort of n-ary cut.

$$\frac{t_0 \vdash t_1 \quad t_1 \vdash t_2 \quad \dots \quad t_{n-1} \vdash t_n}{t_0 \vdash t_n}\,\texttt{Cut}$$

Therefore, the algorithm grows an output tree whose pending leaves contain objects that can be thought of as n-ary cuts between vertices of Π, waiting to be pushed forward. We call these objects *tapes*.

▶ **Definition 11.** A *tape* is a finite list $M := [u_1, \dots, u_n]$ of vertices of Π such that for all $i = 1 \dots n-1$, $\sigma_{\mathtt{R}}(u_i) = \sigma_{\mathtt{L}}(u_{i+1})$.

Analogously to the behaviour of higher order pushdown automata, the algorithm can also be understood as a non-deterministic automaton disposing of a tape as its internal data structure; the tape can be thought of as a generalized stack. The automaton tries to build up a branch of the proof-tree; when the proof-tree branches, the automaton forks into several automata so to construct all the branches; equivalently, we can think that the automaton

undeterministically chooses which branch to continue constructing. The automaton grows up the branch by means of commutative cut reductions (called here *flips*) at the extremities of the tape; if all the cuts in the tape are principal, the automaton undeterministically chooses one and reduces it, without constructing a new node on the branch. While in principle the construction of a complete branch might fail, due to the fact that we cannot operate flips, we shall see that this does not actually happen.

5.1 Primitive operations

Internal operations. What we call *internal operations* on tapes are functions that take (M, i) as input (with some assumptions on M and i) and return a new tape.

- **Elimination of identities.** The first thing we can do is to eliminate identities since they bring nothing to the semantics. Thus, if $\rho(u_i) = \mathtt{Id}$, we define $\textsc{IdElim}(M, i)$ as the tape obtained by removing u_i from M.

$$\dfrac{\dots \quad t_{i-1} \vdash s \qquad \dfrac{}{s \vdash s}\,\mathtt{Id} \qquad s \vdash t_{i+2} \quad \dots}{t_0 \vdash t_n}\,\mathtt{Cut} \overset{\textsc{IdElim}}{\Longrightarrow} \dfrac{\dots \quad t_{i-1} \vdash s \quad s \vdash t_{i+2} \quad \dots}{t_0 \vdash t_n}\,\mathtt{Cut}$$

- **Merging cuts.** Since the tape represents the fusion of some consecutive cuts, we need an operation for merging new cuts into the tape, thus expanding its size. So if $M = [\dots, u_i, \dots]$ and $\rho(u_i) = \mathtt{Cut}$, we define $\textsc{Merge}(M, i) = [\dots, \varsigma_0 u_i, \varsigma_1 u_i, \dots]$. Schematically:

$$\dfrac{\dots \qquad \dfrac{t_i \vdash s \quad s \vdash t_{i+1}}{t_i \vdash t_{i+1}}\,\mathtt{Cut} \qquad \dots}{t_0 \vdash t_n}\,\mathtt{Cut} \overset{\textsc{Merge}}{\Longrightarrow} \dfrac{\dots \quad t_i \vdash s \quad s \vdash t_{i+1} \quad \dots}{t_0 \vdash t_n}\,\mathtt{Cut}$$

- **Essential reductions.** The last internal operation is a bit more subtle. Suppose $\rho(u_i) \in \mathfrak{R}$ and $\rho(u_{i+1}) \in \mathfrak{L}$ for some i. Note that $t_i := \sigma_{\mathtt{R}}(u_i) = \sigma_{\mathtt{L}}(u_{i+1})$ and that this common term is either a product, a sum, a bound variable or a constant. But $\rho(u_i) \in \mathfrak{R}$ implies $t_i \neq 0$ and $\rho(u_i) \in \mathfrak{R}$ implies $t_i \neq 1$, so t_i is not a constant. Hence there are actually three possible scenarios for the values of $\rho(u_i)$ and $\rho(u_{i+1})$: they can be both product rules, both coproduct rules or both fixpoint rules of the same variable. In each case, we can find successors of u_i and u_{i+1} with compatible sequents. We can then *reduce* u_i with u_{i+1} and substitute them in M with their appropriate successors. More precisely:
  - If $\rho(u_i) = \mathtt{R}\times$, $\rho(u_{i+1}) = \mathtt{L}\times_k$, $k \in \{0, 1\}$, then $\textsc{Reduce}(M, i) = [\dots, \varsigma u_i, \varsigma_k u_{i+1}, \dots]$.

$$\dfrac{\dfrac{t_{i-1} \vdash s_0 \quad t_{i-1} \vdash s_1}{\dots \quad t_{i-1} \vdash s_0 \times s_1}\,\mathtt{R}\times \qquad \dfrac{s_k \vdash t_{i+1}}{s_0 \times s_1 \vdash t_{i+1} \quad \dots}\,\mathtt{L}\times_k}{t_0 \vdash t_n}\,\mathtt{Cut} \overset{\textsc{Reduce}}{\Longrightarrow} \dfrac{\dots \; t_{i-1} \vdash s_k \quad s_k \vdash t_{i+1} \; \dots}{t_0 \vdash t_n}\,\mathtt{Cut}$$

  - If $\rho(u_i) = \mathtt{R}+_k$, $\rho(u_{i+1}) = \mathtt{L}+$, $k \in \{0, 1\}$, then $\textsc{Reduce}(M, i) = [\dots, \varsigma_k u_i, \varsigma u_{i+1}, \dots]$.

$$\dfrac{\dfrac{t_{i-1} \vdash s_k}{\dots \quad t_{i-1} \vdash s_0 + s_1}\,\mathtt{R}+_k \qquad \dfrac{s_0 \vdash t_{i+1} \quad s_1 \vdash t_{i+1}}{s_0 + s_1 \vdash t_{i+1} \quad \dots}\,\mathtt{L}+}{t_0 \vdash t_n}\,\mathtt{Cut} \overset{\textsc{Reduce}}{\Longrightarrow} \dfrac{\dots \; t_{i-1} \vdash s_k \quad s_k \vdash t_{i+1} \; \dots}{t_0 \vdash t_n}\,\mathtt{Cut}$$

  - If $\rho(u_i) = \mathtt{R}\theta x$, $\rho(u_{i+1}) = \mathtt{L}\theta x$, $x \in \mathsf{BD}(S)$, then $\textsc{Reduce}(M, i) = [\dots, \varsigma u_i, \varsigma u_{i+1}, \dots]$.

$$\dfrac{\dfrac{t_{i-1} \vdash \tau(x)}{\dots \quad t_{i-1} \vdash x}\,\mathtt{R}\theta x \qquad \dfrac{\tau(x) \vdash t_{i+1}}{x \vdash t_{i+1} \quad \dots}\,\mathtt{L}\theta x}{t_1 \vdash t_n}\,\mathtt{Cut} \overset{\textsc{Reduce}}{\Longrightarrow} \dfrac{\dots \; t_{i-1} \vdash \tau(x) \quad \tau(x) \vdash t_{i+1} \; \dots}{t_0 \vdash t_n}\,\mathtt{Cut}$$

Productions (external operations, flips). We call *productions* functions that take a tape M as input and return a tuple (r, s, L) where r is a rule name, s is a sequent that will be used to create a new vertex of the output tree, and L is a list of tapes that will be the successors of that new vertex.

The simplest case is when $M = [u]$ with $\rho(u) = \mathtt{Id}$. In that case, let $\textsc{IdOut}(M) = (\mathtt{Id}, \sigma(u), [])$. Otherwise, productions can only happen when there is a left rule on the left of M or a right rule on its right. In these cases, we can perform a *commutative reduction*, or *flip* just like in the classical case.

- If $\rho(u_0) = \mathtt{LAx}$, then $\textsc{LFlip}(M) = (\mathtt{LAx}, 0 \vdash t_n, [])$.

$$\dfrac{\dfrac{}{0 \vdash t_1}\,\mathtt{LAx} \qquad t_1 \vdash t_2 \quad \dots}{0 \vdash t_n}\,\mathtt{Cut} \quad \overset{\textsc{LFlip}}{\Longrightarrow} \quad \dfrac{}{0 \vdash t_n}\,\mathtt{LAx}$$

- If $\rho(u_0) = \mathtt{L}\times_k$ for $k \in \{0, 1\}$, then $\textsc{LFlip}(M) = (\mathtt{L}\times_k, t_0 \vdash t_n, [[\varsigma u_0, u_1 \dots]])$.

$$\dfrac{\dfrac{s_k \vdash t_1}{s_0 \times s_1 \vdash t_1}\,\mathtt{L}\times_k \qquad t_1 \vdash t_2 \quad \dots}{s_0 \times s_1 \vdash t_n}\,\mathtt{Cut} \quad \overset{\textsc{LFlip}}{\Longrightarrow} \quad \dfrac{s_k \vdash t_1 \quad t_1 \vdash t_2 \quad \dots}{\dfrac{s_k \vdash t_n}{s_0 \times s_1 \vdash t_n}\,\mathtt{L}\times_k}\,\mathtt{Cut}$$

- If $\rho(u_0) = \mathtt{L}+$, then $\textsc{LFlip}(M) = (\mathtt{L}+, t_0 \vdash t_n, [[\varsigma_0 u_0, u_1, \dots], [\varsigma_1 u_0, u_1 \dots]])$.

$$\dfrac{\dfrac{s_0 \vdash t_1 \;\; s_1 \vdash t_1}{s_0 + s_1 \vdash t_1}\,\mathtt{L}+ \qquad t_1 \vdash t_2 \;\dots}{s_0 + s_1 \vdash t_n}\,\mathtt{Cut} \quad \overset{\textsc{LFlip}}{\Longrightarrow} \quad \dfrac{s_0 \vdash t_1 \;\; t_1 \vdash t_2 \;\dots}{s_0 \vdash t_n}\,\mathtt{Cut} \quad \dfrac{s_1 \vdash t_1 \;\; t_1 \vdash t_2 \;\dots}{s_1 \vdash t_n}\,\mathtt{Cut}$$
$$\dfrac{s_0 \vdash t_n \qquad\qquad s_1 \vdash t_n}{s_0 + s_1 \vdash t_n}\,\mathtt{L}+$$

- If $\rho(u_0) = \mathtt{L}\theta x$, $x \in \mathsf{BD}(S)$, $\theta \in \{\mu, \nu\}$, then $\textsc{LFlip}(M) = (\mathtt{L}\theta x, x \vdash t_n, [[\varsigma u_0, u_1 \dots]])$.

$$\dfrac{\dfrac{\tau(x) \vdash t_1}{x \vdash t_1}\,\mathtt{L}\theta x \qquad t_1 \vdash t_2 \quad \dots}{x \vdash t_n}\,\mathtt{Cut} \quad \overset{\textsc{LFlip}}{\Longrightarrow} \quad \dfrac{\tau(x) \vdash t_1 \quad t_1 \vdash t_2 \quad \dots}{\dfrac{\tau(x) \vdash t_n}{x \vdash t_n}\,\mathtt{L}\theta x}\,\mathtt{Cut}$$

Right flips are defined dually to left flips, so we present them without the schemas.

- If $\rho(u_n) = \mathtt{RAx}$, then $\textsc{RFlip}(M) = (\mathtt{RAx}, t_0 \vdash 1, [])$.
- If $\rho(u_n) = \mathtt{R}\times$, then $\textsc{RFlip}(M) = (\mathtt{R}\times, t_0 \vdash t_n, [[\dots, u_{n-1}, \varsigma_0 u_n], [\dots u_{n-1}, \varsigma_1 u_n]])$.
- If $\rho(u_n) = \mathtt{R}+_k$ for $k \in \{0, 1\}$, then $\textsc{RFlip}(M) = (\mathtt{R}+_k, t_0 \vdash t_n, [[\dots u_{n-1}, \varsigma u_n]])$.
- If $\rho(u_n) = \mathtt{R}\theta x$, $x \in \mathsf{BD}(S)$, $\theta \in \{\mu, \nu\}$, then $\textsc{RFlip}(M) = (\mathtt{R}\theta x, t_0 \vdash x, [[\dots u_{n-1}, \varsigma u_n]])$.

5.2 The cut-elimination algorithm

In order to produce a segment of the output tree, we need a tape with a left rule on the left or a right rule on the right (call such a tape *reduced*). The *treatment* phase of a tape M consists in executing internal operations until the tape is reduced.

Algorithm 1 defines a tree with transitions labelled by $\{0, 1\}$ along with two mappings $\rho : \Lambda \to \Sigma$, and $\sigma = \Lambda \to \mathsf{SEQ}$ that make it into a possibly infinite proof-tree. In the algorithm, $v \in \Pi$ is fixed and Q is a queue whose elements are of the form (w, M) where $w \in \Lambda$ was previously computed and M is a tape.

It may not be clear that the computation of $\textsc{Treat}(M)$ always halts. After all, once a pair of consecutive nodes in the tape is reduced, it is replaced by another pair of nodes that could, in principle, still be labelled by right and left rules, respectively. Even worse: when a new cut is encountered, the tape grows, thus the number of pairs left to be reduced may enlarge. So why does it stop? It is a consequence of the fact that Π satisfies the Guard condition.

Function $\textsc{Treat}(M)$

while $\rho(\mathrm{fst}(M)) \notin \mathfrak{L}$ ***and*** $\rho(\mathrm{lst}(M)) \notin \mathfrak{R}$ **do**
- **if** $|M| > 1$ ***and*** $\exists i : \rho(u_i) = \texttt{Id}$ **then**
  - $M \leftarrow \textsc{IdElim}(M, i)$;
- **else if** $\exists i : \rho(u_i) = \texttt{Cut}$ **then**
  - $M \leftarrow \textsc{Merge}(M, i)$;
- **else if** $\exists i : \rho(u_{i-1}) \in \mathfrak{R}$ ***and*** $\rho(u_i) \in \mathfrak{L}$ **then**
  - $M \leftarrow \textsc{Reduce}(M, i)$;

return M;

Algorithm 1: Cut-elimination

Initialization: $\Lambda \leftarrow \varnothing$; $Q \leftarrow [(\varepsilon, [v])]$;
while $Q \neq []$ **do**
- $(w, M) \leftarrow \mathrm{pull}(Q)$;
- $\Lambda \leftarrow \Lambda \cup \{w\}$;
- $M \leftarrow \textsc{Treat}(M)$;
- **if** $|M| = 1$ ***and*** $\rho(\mathrm{fst}(M)) = \texttt{Id}$ **then**
  - $(\rho(w), \sigma(w), L) \leftarrow \textsc{IdOut}(M)$;
- **else if** $\rho(\mathrm{fst}(M)) \in \mathfrak{L}$ **then**
  - $(\rho(w), \sigma(w), L) \leftarrow \textsc{LFlip}(M)$;
- **else if** $\rho(\mathrm{lst}(M)) \in \mathfrak{R}$ **then**
  - $(\rho(w), \sigma(w), L) \leftarrow \textsc{RFlip}(M)$;
- **if** $L = [M']$ **then**
  - $\mathrm{push}((w0, M'), Q)$;
- **else if** $L = [M'_0, M'_1]$ **then**
  - $\mathrm{push}((w0, M'_0), Q)$;
  - $\mathrm{push}((w1, M'_1), Q)$;

Figure 6 The cut-elimination algorithm.

▶ **Theorem 12.** *For every input tape M, the computation of* $\textsc{Treat}(M)$ *halts.*

Proof. We suppose, for a contradiction, that there is an entry tape M on which the computation of $\textsc{Treat}(M)$ loops forever. For all $i \geq 1$, let M_i be the tape in memory before the i-th turn of the loop (so that $M = M_1$). Consider that tapes are words and that they are generated from one another according to some context dependent grammar; we wish therefore to consider a sort of infinite parse tree. To that end, we define the *full trace* of the algorithm as the reachable graph $T = \overline{T', (0,0)}$, where T' is a transition system over the alphabet $\mathbb{N} \cup \{\bot\}$ with support $\mathbb{N} \times \mathbb{N}$ and the following transitions:

- For $1 \leq i \leq |M_1|$, $(0,0) \xrightarrow{i} (1,i)$.
- If $M_{n+1} = \textsc{IdElim}(M_n, i)$, then for $k < i$, $(n,k) \xrightarrow{\bot} (n+1,k)$ and for $k > i$, $(n,k) \xrightarrow{\bot} (n+1,k-1)$.
- If $M_{n+1} = \textsc{Merge}(M_n, i)$, then for $k < i$, $(n,k) \xrightarrow{\bot} (n+1,k)$ and for $k > i$, $(n,k) \xrightarrow{\bot} (n+1,k+1)$. Moreover $(n,i) \xrightarrow{1} (n+1,i)$ and $(n,i) \xrightarrow{2} (n+1,i+1)$.
- If $M_{n+1} = \textsc{Reduce}(M_n, i)$, then for $k \in \{i, i+1\}$, $(n,k) \xrightarrow{0} (n+1,k)$ and otherwise $(n,k) \xrightarrow{\bot} (n+1,k)$.

For $(n,k) \in T \setminus (0,0)$, let $g(n,k) \in \Pi$ denote the k-th element of M_n. Basically, the paths in T represent the history of the vertices of Π that occur in the tapes. Transitions labelled by $\bot$ mean that such a vertex has not evolved at a given stage, while the other labels encode the operation that made them evolve. In order to exploit the Guard condition, we shall collapse the transitions labelled by $\bot$ to get a correspondence with paths in Π. We then get the *real trace* Ψ of the algorithm.

It should be clear that Ψ is an infinite, finitely branching labelled tree. The prefix order on such a tree is denoted $\sqsubseteq$ and the lexicographical order is denoted $\preceq$ (see [13]). A maximal (finite or infinite) path in Ψ is called a *branch* and it can be shown that the set of branches

of Ψ ordered lexicographically is a complete lattice. Note that by Kőnig's lemma, the set of infinite branches is nonempty and it is easy to see that its infimum is an infinite branch itself.

Given an infinite branch β, we say that β is a μ-branch (resp. *ν-branch*) if the path Γ in Π formed by the transitions between vertices of β can be written $\Gamma = \Gamma_0 \cdot \Gamma_1$ where Γ_0 is finite, Γ_1 has a left μ-trace (resp. right ν-trace) and every fixpoint rule in Γ_1 occurs infinitely often. Since Π satisfies the Guard condition 3, it follows that every infinite branch β is either a μ-branch or a ν-branch. We shall use the lexicographical order to compare them. Note that by the Guard condition, a μ-branch (resp. ν-branch) can only admit finitely many right (resp. left) cuts.

▶ **Lemma 13.**

1. *The least infinite branch of Ψ is a ν-branch.*
2. *Let E be a nonempty collection of ν-branches and let $\gamma = \bigvee E$. Then γ is a ν-branch.*
3. *If β is a ν-branch, then there exists another ν-branch $\beta' \succ \beta$.*

The key observation to prove Lemma 13 is the following. Recall that each time $M_{n+1} = \textsc{Reduce}(M_n, i)$, a pair (u, v) of elements of Ψ is created, such that $u \prec v$ and $\rho(g(u)) \in \mathfrak{R}$, $\rho(g(v)) \in \mathfrak{L}$ are rules of the same nature (product, coproduct or fixpoint on the same variable). u and v are then called *twins* of each other. Conversely, every $u \in \Psi$ that is not the root, a leaf or a cut occurs in such a pair.

Now, let β_0 be the least infinite branch of Ψ. If β_0 were a μ-branch, then it would admit infinitely many left rules. The twins of those rules would then generate an infinite subtree on the left β_0, contradicting the minimality and proving part 1. For part 2, the case $\gamma \in E$ is trivial, and otherwise, one must analyze how the supremum is computed to observe that infinitely many right cuts are necessary. Therefore, γ must be a ν-branch. Finally, for part 3, it suffices to prove that if β is a ν-branch and β' is a μ-branch, such that $\beta \prec \beta'$ are consecutive, then after a finite time, all the forementioned pairs (u, v) are such that $u \sqsubset \beta$ if and only if $v \sqsubset \beta'$. Therefore, the variables $x \in \mathsf{BD}(S)$ for which the fixpoint rule is applied infinitely often in those two branches are the same. But then, the highest priority of such a variable should be both even and odd, a contradiction.

To conclude, we reach the fundamental contradiction of the proof. Let E be the collection of all the ν-branches. By part 1 of Lemma 13, E is nonempty. Let $\gamma = \bigvee E$. By part 2 of the same Lemma, γ is a ν-branch. Hence by part 3 of the Lemma, there is another ν-branch $\gamma' \succ \gamma$. But then, by definition of E, we should have $\gamma' \in E$ and therefore $\gamma' \preceq \bigvee E = \gamma$. ◀

6 Conclusions and perspectives

We consider this work as a starting point for future research—the more we develop it, the more are the questions. A main motivation for this work was the following problem about the computability by means of initial and final coalgebras: since that all the primitive recursive functions are definable by circular proofs [5] and some function space can be constructed via final coalgebras, can we also define more complex set-theoretic functions such as the Ackermann function? The growing knowledge about hierarchies of infinite trees [4] suggested a concrete way to tackle the problem and encouraged us to pursue this work. A concrete step to be taken is now to compare the expressive power of the calculus with respect to these existing hierarchies. For example, can we simulate higher order pushdown automata by means of our tape automaton?

References

1 D. Baelde. Least and greatest fixed points in linear logic. *ACM Trans. Comput. Log.*, 13(1):2, 2012.

2 Y. Bertot and E. Komendantskaya. Inductive and coinductive components of corecursive functions in Coq. *Electr. Notes Theor. Comput. Sci.*, 203(5):25–47, 2008.

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

4 D. Caucal. On infinite transition graphs having a decidable monadic theory. *Theor. Comput. Sci.*, 290(1):79–115, 2003.

5 J. R. B. Cockett and L. Santocanale. Induction, coinduction, and adjoints. *Electr. Notes Theor. Comput. Sci.*, 69:101–119, 2002.

6 J. R. B. Cockett and D. Spencer. Strong categorical datatypes II: A term logic for categorical programming. *Theor. Comput. Sci.*, 139(1&2):69–113, 1995.

7 T. Coquand. Infinite objects in type theory. In H. Barendregt and T. Nipkow, editors, *TYPES*, volume 806 of *Lecture Notes in Computer Science*, pages 62–78. Springer, 1993.

8 R. David, K. Nour, and C. Raffalli. *Introduction à la logique*. Dunod, 2nd edition, 2004.

9 C. Dax, M. Hofmann, and M. Lange. A proof system for the linear time μ-calculus. In S. Arun-Kumar and N. Garg, editors, *FSTTCS*, volume 4337 of *Lecture Notes in Computer Science*, pages 273–284. Springer, 2006.

10 S. Mac Lane. *Categories for the working mathematician*. Springer-Verlag, New York, second edition, 1998.

11 N. P. Mendler. Inductive types and type constraints in the second-order lambda calculus. *Ann. Pure Appl. Logic*, 51(1-2):159–172, 1991.

12 D. Niwinski and I. Walukiewicz. Games for the mu-calculus. *Theor. Comput. Sci.*, 163(1&2):99–116, 1996.

13 D. Perrin and J.-E. Pin. *Infinite Words, Automata, Semigroups, Logic and Games*, volume 141. Elsevier, 2004.

14 G. Rosu and D. Lucanu. Circular coinduction: A proof theoretical foundation. In A. Kurz, M. Lenisa, and A. Tarlecki, editors, *CALCO*, volume 5728 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 2009.

15 L. Santocanale. A calculus of circular proofs and its categorical semantics. Technical Report RS-01-15, BRICS, daimi, May 2001. 30 pp.

16 L. Santocanale. A calculus of circular proofs and its categorical semantics. In M. Nielsen and U. Engberg, editors, *FoSSaCS*, volume 2303 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 2002.

17 L. Santocanale. Free μ-lattices. *J. of Pure and Appl. Algebra*, 168(2-3):227–264, Mar. 2002.

18 L. Santocanale. μ-bicomplete categories and parity games. *Theoretical Informatics and Applications*, 36:195–227, Sept. 2002.

19 T. Studer. On the proof theory of the modal mu-calculus. *Studia Logica*, 89(3):343–363, 2008.

20 I. Walukiewicz. Completeness of Kozen's Axiomatisation of the Propositional μ-Calculus. *Inf. Comput.*, 157(1-2):142–182, 2000.