2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) | 979-8-3503-3587-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/LICS56636.2023.10175772

Computational expressivity of (circular) proofs with fixed points

Gianluca Curzi
University of Birmingham
Birmingham, UK
g.curzi@bham.ac.uk

Anupam Das
University of Birmingham
Birmingham, UK
a.das@bham.ac.uk

***Abstract*—We study the computational expressivity of proof systems with fixed point operators, within the 'proofs-as-programs' paradigm. We start with a calculus $\mu\mathsf{LJ}$ (due to Clairambault) that extends intuitionistic logic by least and greatest positive fixed points. Based in the sequent calculus, $\mu\mathsf{LJ}$ admits a standard extension to a 'circular' calculus $\mathsf{C}\mu\mathsf{LJ}$.**

Our main result is that, perhaps surprisingly, both $\mu\mathsf{LJ}$ and $\mathsf{C}\mu\mathsf{LJ}$ represent the same first-order functions: those provably total in Π^1_2-CA_0, a subsystem of second-order arithmetic beyond the 'big five' of reverse mathematics and one of the strongest theories for which we have an ordinal analysis (due to Rathjen). This solves various questions in the literature on the computational strength of (circular) proof systems with fixed points.

For the lower bound we give a realisability interpretation from an extension of Peano Arithmetic by fixed points that has been shown to be arithmetically equivalent to Π^1_2-CA_0 (due to Möllerfeld). For the upper bound we construct a novel computability model in order to give a totality argument for circular proofs with fixed points. In fact we formalise this argument itself within Π^1_2-CA_0 in order to obtain the tight bounds we are after. Along the way we develop some novel reverse mathematics for the Knaster-Tarski fixed point theorem.

***Index Terms*—Proof theory, Fixed points, Curry-Howard, Circular proofs, Reverse mathematics, Higher-order computability, Realisability**

I. Introduction

Fixed points abound in mathematics and computer science. In logic we may enrich languages by 'positive' fixed points to perform (co)inductive reasoning, while in programming languages positive fixed points in type systems are used to represent (co)datatypes and carry out (co)recursion. In both settings the underlying systems may be construed as fragments of their second-order counterparts.

In this work we investigate the computational expressivity of type systems with least and greatest (positive) fixed points. We pay particular attention to *circular proof systems*, where typing derivations are possibly non-well-founded (but regular), equipped with an ω-regular 'correctness criterion' at the level of infinite branches. Such systems have their origins in modal fixed point logics, notably the seminal work of Niwiński and Walukiewicz [28]. Viewed as type systems under the 'Curry-Howard' correspondence, circular proofs have received significant attention in recent years, notably based in systems of *linear logic* [3], [4], [14]–[16], [18], [19] after foundational work on related finitary systems in [2], [5]. In these settings circular proofs are known to be (at least) as expressive as their finitary counterparts, but classifying the exact expressivity of both systems has remained an open problem. This motivates the main question of the present work:

Question 1: What functions do (circular) proof systems with fixed points represent?

Circular type systems with fixed points were arguably pre-empted by foundational work of Clairambault [8], who introduced an extension $\mu\mathsf{LJ}$ of Gentzen's sequent calculus LJ for intuitionistic propositional logic by least and greatest positive fixed points. $\mu\mathsf{LJ}$ forms the starting point of our work and, using standard methods, admits an extension into a circular calculus, here called $\mathsf{C}\mu\mathsf{LJ}$, whose computational content we also investigate.

In parallel lines of research, fixed points have historically received considerable attention within mathematical logic. The ordinal analysis of extensions of Peano Arithmetic (PA) by inductive definitions has played a crucial role in giving proof theoretic treatments to (impredicative) second-order theories (see, e.g., [31]). More recently, inspired by Lubarsky's work on 'μ-definable sets' [24], Möllerfeld has notably classified the proof theoretic strength of extensions of PA by general inductive definitions in [27].

In this work we somewhat bridge these two traditions, in computational logic and in mathematical logic, in order to answer our main question. In particular we apply proof theoretic and metamathematical techniques to show that both $\mu\mathsf{LJ}$ and $\mathsf{C}\mu\mathsf{LJ}$ represent just the functions provably recursive in the subsystem Π^1_2-CA_0 of second-order arithmetic. This theory is far beyond the 'big five' of reverse mathematics, and is among the strongest theories for which we have an ordinal analysis (see [30]). In contrast, the best known lower bound for $\mu\mathsf{LJ}$ before was Gödel's T, i.e. recursion up to ε_0 [8].

Finally let us point out that our characterisation also applies to aforementioned (circular) systems of linear logic, namely $\mu\mathsf{MALL}$ and its circular counterpart, here called $\mathsf{C}\mu\mathsf{MALL}$, from [2]–[5], [14]–[16], [18], [19], by adapting structural proof theoretic interpretations between the systems. Again, the

The authors would like to thank Pierre Clairambault and Colin Riba for several illuminating conversations about fixed point types. and This work was supported by a UKRI Future Leaders Fellowship, 'Structure vs Invariants in Proofs', project reference MR/S035540/1.

previously best known lower bounds for these systems were forms of (higher-order) primitive recursion, cf. [2], [17], [18]. Thus this work resolves our main Question 1.

A. *Outline and contribution*

The structure of our overall argument is visualised in Figure 1. (1) is a standard embedding of finitary proofs into circular proofs (Proposition 21). (2) reduces $\mathsf{C}\mu\mathsf{LJ}$ to its 'negative fragment' $\mathsf{C}\mu\mathsf{LJ}^-$, in particular free of greatest fixed points (ν), via a double negation translation (Proposition 24).

(3) is one of our main contributions: we build a higher-order computability model $|\cdot|$ that interprets $\mathsf{C}\mu\mathsf{LJ}^-$ (Theorem 39), and moreover formalise this construction itself within $\Pi^1_2\text{-}\mathsf{CA}_0$ to obtain our upper bound (Theorem 62). The domain of this model a priori is an (untyped) term extension of $\mathsf{C}\mu\mathsf{LJ}^-$. It is important for logical complexity here that we interpret fixed points semantically as bona fide fixed points, rather than via encoding into a second-order system. Along the way we must also establish some novel reverse mathematics of the Knaster-Tarski fixed point theorem (Theorem 59).

(4) is an intricate and nontrivial result established by Möllerfeld in [27], which we use as a 'black box'. (5) is again a double negation translation and can be seen as a specialisation of the Π^0_2-conservativity of full second-order arithmetic $\mathsf{PA2}$ over its intuitionistic counterpart $\mathsf{HA2}$ (Theorem 65). (6) is our second main contribution: we provide a realisability interpretation from $\mu\mathsf{HA}$ into $\mu\mathsf{LJ}$ (Theorem 68), morally by considerable specialisation of the analogous interpretation from $\mathsf{HA2}$ into Girard-Reynolds' system F [20], [32]. Our domain of realisers is a (typed) term extension of $\mu\mathsf{LJ}^-$ (the negative fragment of $\mu\mathsf{LJ}$), which is itself interpretable within $\mu\mathsf{LJ}$ (Proposition 24).

Finally, the application of our results to characterise the representable functions of (circular) $\mu\mathsf{MALL}$ is given in Section VIII (Theorem 73).

B. *Related work*

Fixed points have been studied extensively in type systems for programming languages. In particular foundational work by Mendler in the late '80s [25], [26] already cast inductive type systems as fragments of second-order ones such as Girard-Reynolds' F [20], [32]. Aside from works we have already mentioned, (a variant of) (5) has already been obtained by Tupailo in [38]. Berger and Tsuiki have also obtained a similar result to (6) in a related setting [6], for *strictly* positive fixed points.[1] Their interpretation of fixed points is more akin to that in our type structure $|\cdot|$ than our realisability model.

Finally the structure of our argument, cf. Figure 1, is inspired by recent works in cyclic proof theory, notably [12], [34] for (cyclic) (fragments of) PA and [11], [13], [23] for (circular) (fragments of) Gödel's system T.

C. *Full version and notation*

Due to space constraints, most of our proofs and technical development are omitted, but a full version is available [10].

[1]Bound variables may never occur under the left of an arrow.

$$\begin{tikzcd}
\text{Theories of arithmetic:} & \mu\mathsf{HA} \arrow[d, "(6)"'] & \mu\mathsf{PA} \arrow[l, "(5)"'] & \Pi^1_2\text{-}\mathsf{CA}_0 \arrow[l, "(4)"'] \\
\text{Models of computation:} & \mu\mathsf{LJ}^{(-)} \arrow[r, "(1)"'] & \mathsf{C}\mu\mathsf{LJ} \arrow[r, "(2)"'] & \mathsf{C}\mu\mathsf{LJ}^- \arrow[u, "(3)"']
\end{tikzcd}$$

Fig. 1. Summary of the main 'grand tour' of this work. All arrows indicate inclusions of representable functions.

Throughout this work we employ standard rewriting theoretic notation. Namely for a relation $\rightsquigarrow_a$, we denote by $\rightsquigarrow_a^*$ the reflexive and transitive closure of $\rightsquigarrow_a$, and by $=_a$ the symmetric closure of $\rightsquigarrow_a^*$.

We shall make use of *(first-order) variables*, written x, y etc., and *(second-order) variables*, written X, Y etc. throughout. We shall use these both in the setting of type systems and arithmetic theories, as a convenient abuse of notation.

II. Simple types with fixed points: system $\mu\mathsf{LJ}$

In this section we recall the system $\mu\mathsf{LJ}$ from [8], [9].[2]

A. *The sequent calculus* $\mu\mathsf{LJ}$

Pretypes, written σ, τ etc., are generated by the following grammar:

$$\sigma, \tau \ ::= \ X \mid 1 \mid \sigma + \tau \mid \sigma \times \tau \mid \sigma \to \tau \mid \mu X \sigma \mid \nu X \sigma$$

Free (second-order) variables of a pretype are defined as expected, construing μ and ν as binders. A pretype is *closed* if it has no free variables (otherwise it is *open*).

Definition 2 (Types and polarity): Positive and *negative* variables in a pretype are defined as expected. A pretype is a *type* (or even *formula*) if, for any subexpression $\kappa X \sigma$, σ is positive in X. The notions of *(type) context* and *substitution* are defined as usual.

Remark 3 (Positivity vs strict positivity): Many authors require variables bound by fixed point operators to appear in *strictly* positive position, i.e. never under the left-scope of $\to$. Like Clairambault we do not impose this stronger requirement, requiring only positivity in the usual syntactic sense.

Definition 4 (System $\mu\mathsf{LJ}$): A *cedent*, written Σ, Γ etc., is just a list of types. A *sequent* is an expression $\Sigma \Rightarrow \sigma$.[3] The system $\mu\mathsf{LJ}$ is given by the rules of Figures 2, 3 and 4.[4] The notions of *derivation* (or *proof*) are defined as usual. We write $P : \Gamma \Rightarrow \tau$ if P is a derivation of the sequent $\Gamma \Rightarrow \tau$.

Remark 5 ($\mu\mathsf{LJ}$ as a fragment of second-order logic): We may regard $\mu\mathsf{LJ}$ properly as a fragment of Girard-Reynolds System F [20], [32], an extension of simple types to a second-order setting. In particular, (co)inductive types may be identified with second-order formulas by:

$$\begin{aligned} \mu X \sigma &= \forall X ((\sigma \to X) \to X) \\ \nu X \sigma &= \exists X (X \times (X \to \sigma)) \end{aligned}$$

[2]More precisely, we present the 'strong' version of $\mu\mathsf{LJ}$ from [9].

[3]The symbol $\Rightarrow$ is, formally, just a syntactic delimiter, but the arrow notation is suggestive.

[4]Colours may be ignored for now.

$$\mathsf{id}\,\frac{}{\sigma \Rightarrow \sigma} \qquad \mathsf{e}\,\frac{\Gamma, \sigma, \tau, \Delta \Rightarrow \gamma}{\Gamma, \tau, \sigma, \Delta \Rightarrow \gamma} \qquad \mathsf{cut}\,\frac{\Gamma \Rightarrow \sigma \quad \Delta, \sigma \Rightarrow \tau}{\Gamma, \Delta \Rightarrow \tau}$$

$$\mathsf{w}\,\frac{\Gamma \Rightarrow \tau}{\Gamma, \sigma \Rightarrow \tau} \qquad \mathsf{c}\,\frac{\Gamma, \sigma, \sigma \Rightarrow \tau}{\Gamma, \sigma \Rightarrow \tau} \qquad 1_r\,\frac{}{\Rightarrow 1} \qquad 1_l\,\frac{\Gamma \Rightarrow \sigma}{\Gamma, 1 \Rightarrow \sigma}$$

$$\to_r\,\frac{\Gamma, \sigma \Rightarrow \tau}{\Gamma \Rightarrow \sigma \to \tau} \qquad \to_l\,\frac{\Gamma \Rightarrow \sigma \quad \Delta, \tau \Rightarrow \gamma}{\Gamma, \Delta, \sigma \to \tau \Rightarrow \gamma}$$

$$\times_r\,\frac{\Gamma \Rightarrow \sigma \quad \Delta \Rightarrow \tau}{\Gamma, \Delta \Rightarrow \sigma \times \tau} \qquad \times_l\,\frac{\Gamma, \sigma, \tau \Rightarrow \gamma}{\Gamma, \sigma \times \tau \Rightarrow \gamma}$$

$$+_r^0\,\frac{\Gamma \Rightarrow \tau_0}{\Gamma \Rightarrow \tau_0 + \tau_1} \qquad +_r^1\,\frac{\Gamma \Rightarrow \tau_1}{\Gamma \Rightarrow \tau_0 + \tau_1} \qquad +_l\,\frac{\Gamma, \sigma \Rightarrow \gamma \quad \Gamma, \tau \Rightarrow \gamma}{\Gamma, \sigma + \tau \Rightarrow \gamma}$$

Fig. 2. Sequent calculus rules for LJ.

The rules for fixed points in μLJ are essentially inherited from this encoding. Later we shall use a different encoding of fixed point types into a second-order setting, namely in arithmetic, as bona fide fixed points, in order to better control logical complexity.

In proofs that follow, we shall frequently only consider the cases of *least* fixed points (μ-types) and not *greatest* fixed points (ν-types), due to space constraints, appealing to 'duality' for the latter. The cases for ν should be deemed analogous. As we shall soon see, in Subsection III-D, we can indeed reduce our consideration to ν-free types, without loss of generality in terms of representable functions.

Remark 6 (Why sequent calculus?): Using a sequent calculus as our underlying type system is by no means the only choice. However, since we shall soon consider non-wellfounded and circular typing derivations, it is important to have access to a well behaved notion of *formula ancestry*, in order to properly define the usual totality criterion that underlies them. This is why the sequent calculus is the formalism of choice in circular proof theory.

Remark 7 (Variations of the fixed point rules): It is common, e.g. for semantic considerations, to consider context-free (or 'weak') specialisations of the fixed point rules:[5]

$$\mathsf{iter}\,\frac{\sigma(\tau) \Rightarrow \tau}{\mu X \sigma(X) \Rightarrow \tau} \qquad \mathsf{coiter}\,\frac{\rho \Rightarrow \sigma(\rho)}{\rho \Rightarrow \nu X \sigma(X)} \tag{1}$$

In the presence of cut the '(co)iterator' rules above are equivalent to those of μLJ, but since the computational model we presume is cut-reduction, as we shall soon see, it is not appropriate to take them as first-class citizens. However, when giving a semantics that interprets cut directly, e.g. as we do for the term calculi in Section IV, it is often simpler to work with the (co)iterators above.

In the remainder of this work we shall freely use (variations of) the versions above in proofs.

Definition 8 (Functors): Let $\sigma(X)$ and $\rho(X)$ be (possibly open) types that are positive (and negative, respectively) in X.

[5]Note that we have simply omitted identity premisses, for legibility.

$$\mu_r\,\frac{\Gamma \Rightarrow \sigma(\mu X \sigma(X))}{\Gamma \Rightarrow \mu X \sigma(X)} \qquad \nu_l\,\frac{\Gamma, \sigma(\nu X \sigma(X)) \Rightarrow \tau}{\Gamma, \nu X \sigma(X) \Rightarrow \tau}$$

Fig. 3. Some unfolding rules for μ and ν.

$$\mu_l\,\frac{\Gamma, \sigma(\rho) \Rightarrow \rho \quad \Delta, \rho \Rightarrow \tau}{\Gamma, \Delta, \mu X \sigma(X) \Rightarrow \tau} \qquad \nu_r\,\frac{\Gamma \Rightarrow \tau \quad \Delta, \tau \Rightarrow \sigma(\tau)}{\Gamma, \Delta \Rightarrow \nu X \sigma(X)}$$

Fig. 4. '(Co)iteration' rules for μ and ν.

For a proof $P : \Gamma, \tau \Rightarrow \tau'$ we define $\sigma(P) : \Gamma, \sigma(\tau) \Rightarrow \sigma(\tau')$ and $\rho(P) : \Gamma, \rho(\tau') \Rightarrow \rho(\tau)$ by simultaneous induction. If σ and/or ρ are formed from smaller types by a connective of LJ, then the corresponding functors are defined as usual. If $\sigma(X) = \mu Y \sigma'(X, Y)$ then $\sigma(P)$ is,

$$\mu_l\,\cfrac{\mu_r\,\cfrac{\sigma'(P,\sigma(\tau'))\,\cfrac{\Gamma, \tau \Rightarrow \tau'}{\Gamma, \sigma'(\tau, \sigma(\tau')) \Rightarrow \sigma'(\tau', \sigma(\tau'))}}{\Gamma, \sigma'(\tau, \sigma(\tau')) \Rightarrow \sigma(\tau')}}{\Gamma, \sigma(\tau) \Rightarrow \sigma(\tau')}$$

where $\sigma'(P, \sigma(\tau'))$ is obtained from the inductive hypothesis for $\sigma'(P, Y)$ under substitution of $\sigma(\tau')$ for Y. Dually for $\rho(P)$, and similarly for greatest fixed points.

Example 9 (Post-fixed point): It is implicit in the rules of μLJ that $\mu X \sigma(X)$ may be seen as the *least* fixed point of $\sigma(\cdot)$, under a suitable semantics (e.g. later in Section V). The μ_r rule indicates that it is a pre-fixed point, while the μ_l rule indicates that it is least among them. To see that it is also a post-fixed point $\mu_l'\,\frac{\Gamma, \sigma(\mu X \sigma(X)) \Rightarrow \tau}{\Gamma, \mu X \sigma(X) \Rightarrow \tau}$ we may use a derivation that mimics standard textbook-style proofs of Knaster-Tarski:

$$\mu_l\,\cfrac{\sigma\,\cfrac{\mu_r\,\cfrac{\mathsf{id}\,\cfrac{}{\sigma(\mu X \sigma(X)) \Rightarrow \sigma(\mu X \sigma(X))}}{\sigma(\mu X \sigma(X)) \Rightarrow \mu X \sigma(X)}}{\sigma(\sigma(\mu X \sigma(X))) \Rightarrow \sigma(\mu X \sigma(X))} \quad \Gamma, \sigma(\mu X \sigma(X)) \Rightarrow \tau}{\Gamma, \mu X \sigma(X) \Rightarrow \tau}$$

Dually for ν-types as pre-fixed points.

B. Computing with derivations

The underlying computational model for sequent calculi, with respect to the 'proofs-as-programs' paradigm, is *cut-reduction*. In our case this follows a standard set of cut-reduction rules for the calculus LJ. For the fixed points, cut-reduction is inherited directly from the encoding of fixed points in system F that induces our rules, cf. Remark 5. Following Baelde and Miller [5], we give self-contained cut-reductions here:

Definition 10 (Cut-reduction for fixed points): Cut-reduction on μLJ-derivations, written $\rightsquigarrow_{\mathrm{cr}}$, is the smallest relation on derivations including all the usual cut-reductions of LJ[6] and the cut-reduction in Figure 5 and its dual for greatest fixed points. When speaking of (subsets of) μLJ as

[6]As usual, we allow these reductions to be performed on sub-derivations. I.e. they are 'context-closed'.

$$\mathsf{cut}\ \dfrac{\mu_r\ \dfrac{\Sigma \Rightarrow \sigma(\mu X\sigma(X))}{\Sigma \Rightarrow \mu X\sigma(X)} \qquad \mu_l\ \dfrac{\Gamma, \sigma(\rho) \Rightarrow \rho \quad \Delta, \rho \Rightarrow \tau}{\Gamma, \Delta, \mu X\sigma(X) \Rightarrow \tau}}{\Sigma, \Gamma, \Delta \Rightarrow \tau}$$

$$\rightsquigarrow$$

$$\mathsf{c}\ \dfrac{\mathsf{cut}\ \dfrac{\mathsf{cut}\ \dfrac{\mathsf{cut}\ \dfrac{\Sigma \Rightarrow \sigma(\mu X\sigma(X)) \qquad \sigma\ \dfrac{\mu_l\ \dfrac{\Gamma, \sigma(\rho) \Rightarrow \rho}{\Gamma, \mu X\sigma(X) \Rightarrow \rho}}{\Gamma, \sigma(\mu X\sigma(X)) \Rightarrow \sigma(\rho)}}{\Sigma, \Gamma \Rightarrow \sigma(\rho)} \qquad \Gamma, \sigma(\rho) \Rightarrow \rho}{\Sigma, \Gamma, \Gamma \Rightarrow \rho} \qquad \Delta, \rho \Rightarrow \tau}{\Sigma, \Delta, \Gamma, \Gamma \Rightarrow \tau}}{\Sigma, \Delta, \Gamma \Rightarrow \tau}$$

Fig. 5. Reduction of a key cut between μ_r and μ_l in μLJ.

$$N_r^0\ \dfrac{}{\Rightarrow N} \qquad N_r^1\ \dfrac{\Gamma \Rightarrow N}{\Gamma \Rightarrow N} \qquad N_l\ \dfrac{\Gamma \Rightarrow \sigma \quad \Gamma, \sigma \Rightarrow \sigma \quad \Delta, \sigma \Rightarrow \tau}{\Gamma, \Delta, N \Rightarrow \tau}$$

Fig. 6. Native rules for N in μLJ.

a computational model, we always mean with respect to $=_{\mathrm{cr}}$ (the reflexive transitive symmetric closure of $\rightsquigarrow_{\mathrm{cr}}$).

Definition 11 (Representability in μLJ): We define the *type of natural numbers* as $N := \mu X(1 + X)$. We also define the *numeral* $\underline{n} : N$ by induction on $n \in \mathbb{N}$:

$$\underline{0} \ := \ \mu_r\ \dfrac{+_r\ \dfrac{1\ \dfrac{}{\Rightarrow 1}}{\Rightarrow 1 + N}}{\Rightarrow N} \qquad \underline{n+1} \ := \ \mu_r\ \dfrac{+_r\ \dfrac{\underline{n}\ \dfrac{}{\Rightarrow N}}{\Rightarrow 1 + N}}{\Rightarrow N}$$

We say that a (possibly partial) function $f : \mathbb{N} \times \overset{k}{\cdots} \times \mathbb{N} \to \mathbb{N}$ is *representable* in μLJ if there is a μLJ-derivation P_f : $N, \overset{k}{\ldots}, N \Rightarrow N$ s.t., for any $n_1, \ldots, n_k \in \mathbb{N}$, the derivation,

$$\mathsf{cut}\ \dfrac{\underline{n_k}\ \dfrac{}{\Rightarrow N} \qquad \mathsf{cut}\ \dfrac{\underline{n_2}\ \dfrac{}{\Rightarrow N} \qquad \mathsf{cut}\ \dfrac{\underline{n_k}\ \dfrac{}{\Rightarrow N} \quad \overset{P_f}{N, \overset{k}{\ldots}, N \Rightarrow N}}{N, \overset{k-1}{\ldots}, N \Rightarrow N}}{\vdots}}{\Rightarrow N}$$

converts under $=_{\mathrm{cr}}$ to the numeral $\underline{f(n_1, \ldots, n_k)}$, whenever it is defined (otherwise it converts to no numeral). In this case we say that P_f *represents* f in μLJ.

Example 12 (Native rules for natural number computation): 'Native' rules for type N in μLJ are given in Figure 6. Their derivations in μLJ, as well as the derivations of corresponding 'native' cut-reductions in $\rightsquigarrow_{\mathrm{cr}}$, are routine. Note that, from here we can recover the usual recursor of system T, as shown formally by Clairambault [9].

C. Further examples

Example 13: The least and greatest fixed point operators μ and ν allow us to encode inductive data (natural numbers, lists, etc) and coinductive data (streams, infinite trees, etc). We have

$$\mu_r\ \dfrac{+_r^1\ \dfrac{\mathsf{id}\ \dfrac{}{N \Rightarrow N}}{N \Rightarrow 1 + N}}{N \Rightarrow N} \qquad \mu_r\ \dfrac{+_r^0\ \dfrac{1_r\ \dfrac{}{\Rightarrow 1}}{\Rightarrow 1 + (N \times L)}}{\Rightarrow L} \qquad \mu_r\ \dfrac{+_r^1\ \dfrac{\times\ \dfrac{\overset{\underline{n}}{\Rightarrow N} \quad \overset{\underline{l}}{\Rightarrow L}}{\Rightarrow N \times L}}{\Rightarrow 1 + (N \times L)}}{\Rightarrow L}$$

Fig. 7. Some computations on natural numbers and lists in μLJ.

$$\nu_l\ \dfrac{+_l\ \dfrac{1_l\ \dfrac{\to_r\ \dfrac{\mathsf{id}\ \dfrac{}{S \Rightarrow S}}{\Rightarrow S \to S}}{1 \Rightarrow S \to S} \qquad \times_l\ \dfrac{\to_r\ \dfrac{\nu_r\ \dfrac{\times_r\ \dfrac{\to_l\ \dfrac{\mathsf{id}\ \dfrac{}{S \Rightarrow S} \quad \mathsf{id}\ \dfrac{}{S \Rightarrow S}}{S \to S, S \Rightarrow S} \quad \mathsf{id}\ \dfrac{}{N \Rightarrow N}}{N, S \to S, S \Rightarrow N \times S}}{N, S \to S, S \Rightarrow S}}{N, S \to S \Rightarrow S \to S}}{N \times (S \to S) \Rightarrow S \to S}}{1 + (N \times (S \to S)) \Rightarrow S \to S}}{L \Rightarrow S \to S}$$

Fig. 8. Concatenation of a list and a stream in μLJ.

already seen the encoding of natural numbers. The type of lists and streams (both over natural numbers) can be represented by, respectively, $L := \mu X.1 + (N \times X)$ and $S = \nu X.N \times X$.[7] Figure 7, left-to-right, shows the encoding of the successor on natural numbers, the encoding of ε and $n :: l$ (i.e., the empty list and appending a natural number to a list). Figure 8 shows the encoding of a concatenation of a list and a stream into a stream (by recursion over the list with the invariant $S \to S$).[8]

III. A circular version of μLJ

In this section we shall develop a variation of μLJ that does not have rules for (co)iteration, but rather devolves such work to the proof structure. First, let us set up the basic system of rules we will work with:

Definition 14 (μLJ 'without (co)iteration'): Write μ'LJ for the system of all rules in Figures 2, 3 and 9 (but not 4).

A. 'Non-wellfounded' proofs over μ'LJ

'Coderivations' are generated *coinductively* by the rules of a system, dually to derivations that are generated inductively. I.e. they are possibly infinite proof trees (of height $\leq \omega$) generated by the rules of a system. We say that a coderivation is *regular* (or *circular*) if it has only finitely many distinct sub-coderivations. A regular coderivation can be represented as a finite labelled graph (possibly with cycles) as expected.

B. Computing with coderivations

Just like for usual derivations, the underlying notion of computation for coderivations is cut-reduction, and the notion of representability remains the same. However we must also

[7]As is typical, we use the '.' after a binder to indicate that its scope is as far to the right as possible.

[8]Both examples were originally given for μMALL in [17].

$$\mu'_l\ \dfrac{\Gamma, \sigma(\mu X.\sigma) \Rightarrow \tau}{\Gamma, \mu X.\sigma \Rightarrow \tau} \qquad \nu'_r\ \dfrac{\Gamma \Rightarrow \sigma(\nu X.\sigma)}{\Gamma \Rightarrow \nu X.\sigma}$$

Fig. 9. Further unfolding rules for μ and ν.

adapt the theory of cut-reduction to the different fixed point rules of μ'LJ.

Definition 15 (Cut-reduction on coderivations): $\rightsquigarrow_{\mathrm{cr}'}$ is the smallest relation on μ'LJ-coderivations including all the usual cut-reductions of LJ and,[9]

$$\text{cut}\ \dfrac{\mu_r\ \dfrac{\Gamma \Rightarrow \sigma(\mu X\sigma(X))}{\Gamma \Rightarrow \mu X\sigma(X)} \quad \mu'_l\ \dfrac{\Delta, \sigma(\mu X\sigma(X)) \Rightarrow \tau}{\Delta, \mu X\sigma(X) \Rightarrow \tau}}{\Gamma, \Delta \Rightarrow \tau}$$
$$\rightsquigarrow \quad \text{cut}\ \dfrac{\Gamma \Rightarrow \sigma(\mu X\sigma(X)) \quad \Delta, \sigma(\mu X\sigma(X)) \Rightarrow \tau}{\Gamma, \Delta \Rightarrow \tau}$$

and dually for greatest fixed points.

When speaking of (subsets of) coderivations as computational models, we typically mean with respect to $=_{\mathrm{cr}'}$ (the reflexive symmetric transitive closure of $\rightsquigarrow_{\mathrm{cr}'}$).

Example 16 (Decomposing the (co)iterators): The '(co)iterator' rules of Figure 4 can be expressed by regular coderivations using only the unfolding rules for fixed points as follows:

$$\text{cut}\ \dfrac{\mu'_l\ \dfrac{\text{c}\ \dfrac{\text{cut}\ \dfrac{\sigma\ \dfrac{\mu'_l\ \dfrac{\vdots}{\Gamma, \mu X\sigma(X) \Rightarrow \rho}\ \bullet}{\Gamma, \sigma(\mu X\sigma(X)) \Rightarrow \sigma(\rho)} \quad \Gamma, \sigma(\rho) \Rightarrow \rho}{\Gamma, \Gamma, \sigma(\mu X\sigma(X)) \Rightarrow \rho}}{\Gamma, \sigma(\mu X\sigma(X)) \Rightarrow \rho}}{\Gamma, \mu X\sigma(X) \Rightarrow \rho}\ \bullet \qquad \Delta, \rho \Rightarrow \tau}{\Gamma, \Delta, \mu X\sigma(X) \Rightarrow \tau} \tag{2}$$

Here we mark with $\bullet$ roots of identical coderivations, a convention that we shall continue to use throughout this work. Dually for the coiterator.

Moreover, one can verify that this embedding gives rise to a bona fide simulation of $=_{\mathrm{cr}}$ by $=_{\mathrm{cr}'}$. We do not cover the details at this point, but make a stronger statement later in Proposition 21.

Example 17 (Functors and η-expansion of identity): Thanks to the decomposition of (co)iterators above, we can derive 'functors' in CμLJ, cf. Definition 8. This gives rise to an 'η-expansion' of identity steps, reducing them to atomic form.

C. A totality criterion

We shall adapt to our setting a well-known 'termination criterion' from non-wellfounded proof theory. First, let us recall some standard proof theoretic concepts about (co)derivations, similar to those in [4], [7], [11], [13], [23].

Definition 18 (Ancestry): Fix a μ'LJ-coderivation P. We say that a type occurrence σ is an *immediate ancestor* of a type occurrence τ in P if they are types in a premiss and

[9]Again, we allow these reductions to be applied on sub-coderivations.

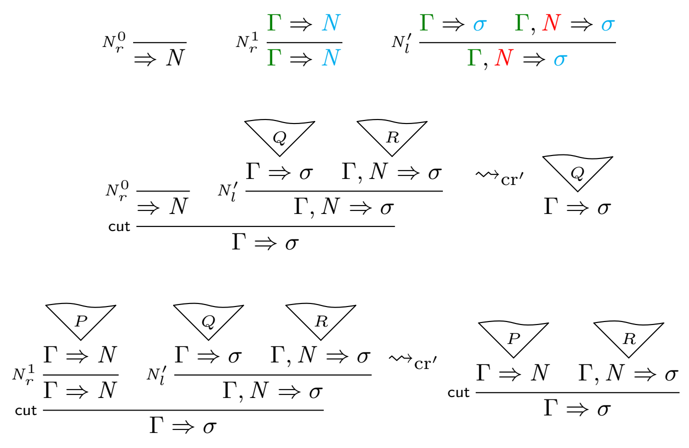

Fig. 10. Inference rules and cut-reduction steps for N in μ'LJ.

conclusion (respectively) of an inference step and, as typeset in Figure 2, Figure 3 and Figure 9, have the same colour. If σ and τ are in some Γ or Δ, then furthermore they must be in the same position in the list.

Being a binary relation, immediate ancestry forms a directed graph upon which our correctness criterion is built. Our criterion is essentially the same as that from [4], only for μLJ instead of μMALL.

Definition 19 (Threads and progress): A *thread* along (a branch of) P is a maximal path in P's graph of immediate ancestry. We say a thread is *progressing* if it is infinitely often principal and has a smallest infinitely often principal formula that is either a μ-formula on the LHS or a ν-formula on the RHS. A coderivation P is *progressing* if each of its infinite branches has a progressing thread.

We shall use several properties of (progressing) threads in Section V which are relatively standard, e.g. [17], [36].

Definition 20 (Circular system): CμLJ is the class of regular progressing μ'LJ-coderivations.

Referencing Example 16, and for later use, we shall appeal to the notion of *simulation* for comparing models of computation in this work. Recalling that we construe μLJ as a model of computation under $=_{\mathrm{cr}}$ and CμLJ as a model of computation under $=_{\mathrm{cr}'}$, we have:

Proposition 21 (Simulation): CμLJ simulates μLJ.

Proof sketch: Replace each instance of a (co)iterator by the corresponding regular coderivation in Example 16. Note that those coderivations are indeed progressing due to the progressing thread on $\mu X\sigma(X)$ along the unique infinite branch in the case of μ_l (dually for ν_r). The statement follows by closure of CμLJ under its rules. ■

Example 22 (Revisiting natural number computation): Just like for μLJ, we give native rules for N in μ'LJ, along with corresponding cut-reductions in Figure 10. As before, it is routine to show that these reductions are derivable using $\rightsquigarrow_{\mathrm{cr}'}$.

Now, specialising our simulation result to recursion on N, we have the following regular coderivation for the recursor of system T (at type σ):

$$\dfrac{\Gamma\Rightarrow\sigma \qquad {\scriptstyle \mathsf{cut}}\dfrac{{\scriptstyle N_l'}\dfrac{\vdots}{\Gamma, N\Rightarrow\sigma}\bullet \qquad \Gamma,N,\sigma\Rightarrow\sigma}{\Gamma,N\Rightarrow\sigma}}{\Gamma,N\Rightarrow\sigma}\,{\scriptstyle N_l'}$$

Indeed it is immediate that $\mathsf{C}\mu\mathsf{LJ}$ contains circular versions of system T from [11], [13], [23].

D. Reduction to the negative fragment

It is folklore that coinductive types can be eliminated using inductive types (possibly at the loss of strict positivity) using, say, a version of the Gödel-Gentzen negative translation, without affecting the class of representable functions (as long as N is included as a primitive data type) (see, e.g., [1]). Indeed this translation can be designed to eliminate other 'positive' connectives too, in particular $+$.[10]

The same trick does not quite work for coderivations since it introduces cuts globally that may break the progressing criterion in the limit of the translation. However a version of the Kolmogorov translation, more well behaved at the level of cut-free proof theory, is well-suited for this purpose.

In this section we establish such a reduction from $\mathsf{C}\mu\mathsf{LJ}$ to its 'negative' fragment. Not only is this of self-contained interest, being more subtle than the analogous argument for $\mu\mathsf{LJ}$ (and type systems with (co)inductive data types), this will also greatly simplify reasoning about the representable functions of $\mathsf{C}\mu\mathsf{LJ}$ in what follows, in particular requiring fewer cases in arguments therein.

Definition 23 (Negative fragments): We define $\mu\mathsf{LJ}^-$ as the subsystem of $\mu\mathsf{LJ}$ using only (native) rules and cut-reductions over $N,\times,\to,\mu$.[11] We define $\mu'\mathsf{LJ}^-$ and $\mathsf{C}\mu\mathsf{LJ}^-$ similarly, only as subsystems of $\mu'\mathsf{LJ}$-coderivations and their cut-reductions.

The main result of this subsection is:

Proposition 24: Any function on natural numbers representable in $\mathsf{C}\mu\mathsf{LJ}$ is also representable in $\mathsf{C}\mu\mathsf{LJ}^-$.

Proof idea: We give a bespoke combination of a Kolmogorov negative translation and a Friedman-Dragalin 'A-translation' (setting $A = N$). We define the translations $\cdot^N$ and $\cdot_N$ from arbitrary types to types over $\{N,\times,\to,\mu\}$ as follows, where $\neg\sigma := \sigma\to N$:

$$\begin{aligned}
\sigma^N &:= \neg\sigma_N\\
X_N &:= \neg X\\
1_N &:= N\\
(\sigma\times\tau)_N &:= \neg(\sigma^N\times\tau^N)\\
(\sigma\to\tau)_N &:= \neg(\sigma^N\to\tau^N)\\
(\sigma+\tau)_N &:= \neg\sigma^N\times\neg\tau^N\\
(\nu X\sigma)_N &:= \neg\neg\mu X\neg\sigma^N[\neg X/X]\\
(\mu X\sigma)_N &:= \neg\mu X\sigma^N
\end{aligned}$$

[10]Note that the attribution of 'positive' or 'negative' to a connective is unrelated to that of positive or negative context.

[11]In particular we insist that the native rules and cut-reductions for N are used to avoid extraneous occurrences of $+$ and remain internal to the fragment.

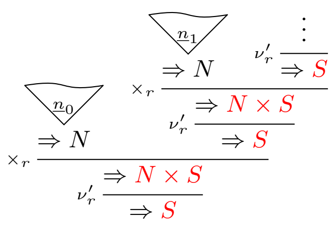

Fig. 11. Pointwise computation of a stream in $\mathsf{C}\mu\mathsf{LJ}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{S\Rightarrow S}\,{\scriptstyle\mathsf{id}}}{1,S\Rightarrow S}\,{\scriptstyle 1_l}\qquad \dfrac{\dfrac{\dfrac{\dfrac{}{N\Rightarrow N}\,{\scriptstyle\mathsf{id}}\qquad \dfrac{\vdots}{\color{red}{L},S\Rightarrow \color{red}{S}}\,{\scriptstyle\mu_l'}\bullet}{N,\color{red}{L},S\Rightarrow N\times S}\,{\scriptstyle\times_r}}{\color{red}{N\times L},S\Rightarrow N\times S}\,{\scriptstyle\times_l}}{\color{red}{N\times L},S\Rightarrow S}\,{\scriptstyle\nu_r'}}{\dfrac{\color{red}{1+(N\times L)},S\Rightarrow S}{\dfrac{\color{red}{L},S\Rightarrow S}{\color{red}{L}\Rightarrow S\to S}\,{\scriptstyle\to_r}}\,{\scriptstyle\mu_l'}\bullet}\,{\scriptstyle +_l}}{}$$

Fig. 12. Concatenation of a list and a stream in $\mathsf{C}\mu\mathsf{LJ}$.

The translation can be extended to coderivations by mapping every inference rule r to a gadget r^N preserving threads. ■

E. Further examples

Example 25: Figure 11 shows the encoding of a stream $n_0 :: n_1 :: n_2\ldots$ by a (not necessarily regular) coderivation.[12] Figure 12 shows the circular presentation of the concatenation of a list and a stream into a stream discussed in Example 13. Note that, compared to the inductive encoding of this function, the circular one has a more 'explicit' computational meaning. Both coderivations are progressing, as the red-thread contained in the only infinite branch progresses infinitely often.

It is worth discussing how computation over streams is simulated in $\mathsf{C}\mu\mathsf{LJ}^-$ via the double negation translation illustrated in Proposition 24. The type S of streams is translated into $\neg\neg\neg\mu X.\neg\neg\neg(N^N\times\neg\neg\neg X)$, for some appropriate translation N^N of the type for natural numbers. Hence, computation over streams is simulated by computation over a type of the form $(\mu X.\sigma\to N)\to N$. Note that this resembles (and embeds) the type $N\to N$ for representing streams in system T, so in some sense we can see $\cdot^N$-translation as extending/adapting the embedding of S into $N\to N$.

IV. Extensions to (un)typed term calculi

In light of the reduction to the negative fragment at the end of the previous section, we shall only consider types formed from $N,\times,\to,\mu$ henceforth.

A. From (co)derivations to (co)terms: rules as combinators

It will be convenient for us to extend our computational model from just (co)derivations to a larger class of untyped

[12]Indeed, the coderivation is regular just if the stream is a regular word.

(co)terms. The main technical reason behind this is to allow the definition of a higher-order computability model necessary for our ultimate totality argument for $\mathsf{C}\mu\mathsf{LJ}$. At the same time, we obtain a compressed notation for (co)derivations for notational convenience, and indeed carve out typed (conservative) extensions of the proof calculi thusfar considered.

In what follows, we use the metavariables r etc. to vary over inference steps of $\mu\mathsf{LJ}^-$ and/or $\mu'\mathsf{LJ}^-$ [13]

Definition 26 ((Co)terms): A *coterm*, written s, t etc., is generated coinductively by the grammar:[14]

$$s, t \ ::= \ \mathsf{r} \mid s\,t$$

A coterm is a *term* if it is a finite expression, i.e. generated inductively from the grammar above. If all steps in a (co)term are from a system R, we may refer to it as a R-(co)term.

Our notion of (co)term is untyped, in that an application st may be formed regardless of any underlying typing. (Co)terms will be equipped with a theory that (a) subsumes cut-reduction on (co)derivations; and (b) results in a computational model that is Turing complete. Before that, however, let us see how (co)derivations can be seen as (co)terms.

Definition 27 ((Co)derivations as (co)terms): We construe each $\mathsf{C}\mu\mathsf{LJ}^-$ coderivation as a coterm (and each $\mu\mathsf{LJ}^-$ derivation as a term) by identifying rule application with term application: if P ends with an inference step r with immediate sub-coderivations $P_1, \dots, P_n$ then P is $\mathsf{r}\,P_1 \dots P_n$.

Given a set A of (co)terms, the *closure* of A, written $\langle A \rangle$, is the smallest set of coterms containing A and closed under application, i.e. if $s, t \in \langle A \rangle$ then also $st \in \langle A \rangle$.

Of course if P is a derivation, then it is also a term. Of particular interest to us in this work will be the class $\langle \mathsf{C}\mu\mathsf{LJ}^- \rangle$, essentially finitary applications of progressing regular $\mu'\mathsf{LJ}^-$-coderivations.

Example 28 (Iterator coderivation as a regular coterm): Recalling the decomposition of the iterator as a circular coderivation in Example 16, let us specialise to the variation $\mathsf{iter}\,\dfrac{\sigma(\tau) \Rightarrow \tau}{\mu X \sigma(X) \Rightarrow \tau}$. By following Example 16, we can express $\mathsf{iter}\,P$ by a regular coderivation, say $\mathsf{iter}'(P)$, that, viewed as a coterm, satisfies the (syntactic) equation,

$$\mathsf{iter}'(P) \ = \ \mu'_l\,(\mathsf{cut}\,\sigma(\mathsf{iter}'(P))\,P) \tag{3}$$

Note that $\mathsf{iter}'(P)$ above is indeed a *regular* coterm: it has only finitely many distinct sub-coterms.

B. Computational models: theories of (co)terms

Let us henceforth make the following abbreviations:[15]

$$\langle\cdot,\cdot\rangle : {\times_r}\,\frac{\Rightarrow \sigma \quad \Rightarrow \tau}{\Rightarrow \sigma \times \tau} \qquad \mathsf{p}_i : \ {\times_l}\,\dfrac{\mathsf{w}\,\dfrac{\mathsf{id}\,\dfrac{}{\sigma_i \Rightarrow \sigma_i}}{\sigma_0, \sigma_1 \Rightarrow \sigma_i}}{\sigma_0 \times \sigma_1 \Rightarrow \sigma_i}$$

[13]I.e. instances of any inference rules in these systems.

[14]I.e. they are possibly infinite expressions (of depth $\leq \omega$).

[15]We may omit types from subscripts when unambiguous.

$$\begin{array}{rcl}
\mathsf{id}\;x & \rightsquigarrow & x \\
\mathsf{e}\;t\;\vec{x}\;x\;y\;\vec{y} & \rightsquigarrow & t\;\vec{x}\;y\;x\;\vec{y} \\
\mathsf{w}\;t\;\vec{x}\;x & \rightsquigarrow & t\;\vec{x} \\
\mathsf{c}\;t\;\vec{x}\;x & \rightsquigarrow & t\;\vec{x}\;x\;x \\
\mathsf{cut}\;s\;t\;\vec{x} & \rightsquigarrow & t\;\vec{x}\;(s\;\vec{x}) \\
\times_r\;s\;t\;\vec{x}\;\vec{y} & \rightsquigarrow & \langle s\,\vec{x}, t\,\vec{y}\rangle \\
\times_l\;t\,\vec{x}\,y & \rightsquigarrow & t\;\vec{x}\;(\mathsf{p}_0 y)\;(\mathsf{p}_1 y) \\
\mathsf{p}_i\langle x_0, x_1\rangle & \rightsquigarrow & x_i \\
\rightarrow_r\;t\;\vec{x}\;x & \rightsquigarrow & t\;\vec{x}\;x \\
\rightarrow_l\;s\;t\;\vec{x}\;\vec{y}\;z & \rightsquigarrow & t\;\vec{y}\;(z\,(s\;\vec{x}))
\end{array}$$

Fig. 13. Reduction for LJ^- (both $\rightsquigarrow_{\mathrm{r}}$ and $\rightsquigarrow_{\mathrm{r}'}$).

$$\mathsf{r}\;t\;\vec{x} \quad \rightsquigarrow_{\mathrm{r}'} \quad t\;\vec{x} \qquad \mathsf{r} \in \{\mu_r, \mu'_l\}$$

Fig. 14. Reduction for least fixed point rules in $\mu'\mathsf{LJ}$.

$$\mathsf{in}_\sigma : \mu_r\,\frac{\Rightarrow \sigma(\mu X \sigma(X))}{\Rightarrow \mu X \sigma(X)} \qquad \mathsf{iter}_\sigma : \mu_l\,\frac{\sigma(\tau) \Rightarrow \tau}{\mu X \sigma(X) \Rightarrow \tau}$$

$$0 : N^0_r \qquad \mathsf{s} : N^1_r\,\frac{\Rightarrow N}{\Rightarrow N} \qquad \mathsf{iter}_N : N_l\,\frac{\Rightarrow \sigma \quad \sigma \Rightarrow \sigma}{N \Rightarrow \sigma}$$

for $i \in \{0, 1\}$.

When referring to an arbitrary instance of a rule, the specification should be understood to be as originally typeset, unless otherwise indicated. In particular, we follow this convention to define our notion of reduction on coterms:

Definition 29 (Theories): We define two (context-closed) reduction relations on (co)terms:[16]

- $\rightsquigarrow_{\mathrm{r}}$ is generated by the clauses in Figures 13, 15 and 16.
- $\rightsquigarrow_{\mathrm{r}'}$ is generated by the rules in Figures 13, 14 and 17.

Again we write $=_{\mathrm{r}}$ and $=_{\mathrm{r}'}$ for the reflexive transitive symmetric closures of these relations, respectively.

When referring to (fragments of) $\langle \mu\mathsf{LJ}^- \rangle$ as a computational model, we typically mean with respect to $=_{\mathrm{r}}$, and when referring to (fragments of) $\langle \mathsf{C}\mu\mathsf{LJ}^- \rangle$ as a computational model, we typically mean with respect to $=_{\mathrm{r}'}$. However we also consider a (weakly) extensional version of $=_{\mathrm{r}'}$:[17]

- $=^\eta_{\mathrm{r}'}$ is the closure of $=_{\mathrm{r}'}$ under the rule $\eta\,\dfrac{t\,x =^\eta_{\mathrm{r}'} t'\,x}{t =^\eta_{\mathrm{r}'} t'}$.[18]

Example 30 (Iteration equations): The fundamental equation for iteration is indeed derivable by $\rightsquigarrow_{\mathrm{r}}$:

$$\begin{aligned}
\mathsf{iter}\,P\,(\mathsf{in}_\sigma\,x) &\rightsquigarrow_{\mathrm{r}} \mathsf{in}_\sigma\,x\,P \\
&\rightsquigarrow_{\mathrm{r}} P\,(\sigma(\mathsf{iter}\,P)\,x)
\end{aligned}$$

[16]The use of both variables and term metavariables in these rules is purely to aid parsing. All reduction rules are closed under substitution.

[17]This is not necessary to reason about representability, since extensionality can be eliminated for low type levels, but simplifies some theorem statements.

[18]x here must be a (fresh) variable, not a (co)term.

$$\begin{aligned}\mu_l\, s\, t\, \vec{x}\, \vec{y}\, z &\rightsquigarrow_{\mathrm{r}} t\, \vec{y}\, (\mathsf{iter}\,(s\,\vec{x})\, z)\\ \mu_r\, t\, \vec{x} &\rightsquigarrow_{\mathrm{r}} \mathsf{in}_\sigma(t\,\vec{x})\\ \mathsf{iter}\, t\, x &\rightsquigarrow_{\mathrm{r}} x\, t\\ \mathsf{in}_\sigma\, x\, t &\rightsquigarrow_{\mathrm{r}} t\,(\sigma(\mathsf{iter}\, t)\, x)\end{aligned}$$

Fig. 15. Reduction for least fixed points in $\mu\mathsf{LJ}$.

$$\begin{aligned}N_l\, s\, t\, u\, \vec{x}\, \vec{y}\, z &\rightsquigarrow_{\mathrm{r}} u\, \vec{y}\, (\mathsf{iter}_N\,(s\,\vec{x})\,(t\,\vec{x})\, z)\\ N_r^1\, t\, \vec{x} &\rightsquigarrow_{\mathrm{r}} \mathsf{s}\,(t\,\vec{x})\\ \mathsf{iter}_N\, s\, t\, 0 &\rightsquigarrow_{\mathrm{r}} s\\ \mathsf{iter}_N\, s\, t\, \mathsf{s}x &\rightsquigarrow_{\mathrm{r}} t\,(\mathsf{iter}_N\, s\, t\, x)\end{aligned}$$

Fig. 16. Reduction for N in $\mu\mathsf{LJ}^-$

For $\rightsquigarrow_{\mathrm{r}'}$, recalling Example 28 and using its notation, we can simulate the iteration equation for $\mathsf{iter}\, P$ with $\mathsf{iter}'(P)$:

$$\begin{aligned}\mathsf{iter}'(P)\,(\mathsf{in}_\sigma\, x) &\rightsquigarrow_{\mathrm{r}'} \mathsf{iter}'(P)\, x && \text{by } \mathsf{in}_\sigma \text{ reduction}\\ &\rightsquigarrow_{\mathrm{r}'} \mathsf{cut}\, \sigma(\mathsf{iter}'(P))\, P\, x && \text{by } \mu_l' \text{ reduction}\\ &\rightsquigarrow_{\mathrm{r}'} P\,(\sigma(\mathsf{iter}'(P))\, x) && \text{by cut reduction}\end{aligned}$$

More importantly for us, our notion of extensional reduction on coterms subsumes that of cut-reduction on coderivations. Since we have identified coderivations as coterms, we may state this rather succinctly, constituting the main result of this subsection:

Theorem 31 (Extensional conversion includes cut-conversion): $=_{\mathrm{cr}'} \subseteq =^{\eta}_{\mathrm{r}'}$.

C. An embedding into λ-terms

While the significant technical development of this work involves 'totality' arguments, e.g. in Section V showing that the representable partial functions of $\mathsf{C}\mu\mathsf{LJ}^-$ (under $=^{\eta}_{\mathrm{r}'}$) are total, we better address *determinism* too.

As it stands, $\rightsquigarrow_{\mathrm{r}'}$ can fire distinct reductions on the same or overlapping redexes, so there is a priori no guarantee that the output of representable functions is unique. However this can be shown by defining a straightfoward interpretation of coterms of $\langle\mathsf{C}\mu\mathsf{LJ}^-\rangle$ into the λ-calculus.

Theorem 32: The (untyped) λ-calculus, under $\beta\eta$-conversion, simulates $\langle\mathsf{C}\mu\mathsf{LJ}^-\rangle$ under $=^{\eta}_{\mathrm{r}'}$.
This simulation relies on the fact that we can express regular coterms as a finite system of equations, which are known to always have solutions in the (untyped) λ-calculus (e.g. [22]).

From here, by confluence of $\beta\eta$-reduction on λ-terms, we immediately have:

Corollary 33 (Uniqueness): Let $t \in \langle\mathsf{C}\mu\mathsf{LJ}^-\rangle$. If $\underline{m} =^{\eta}_{\mathrm{r}'} t =^{\eta}_{\mathrm{r}'} \underline{n}$ then $n = m$.

D. From typed terms back to proofs

Let us restrict our attention to $\mu\mathsf{LJ}^-$-terms in this subsection. In what follows, for a list of types $\vec{\sigma} = (\sigma_1, \dots, \sigma_n)$, we write $\vec{\sigma} \to \tau$ for $\sigma_1 \to \dots \to \sigma_n \to \tau$. In order to more easily carry out our realisability argument in Section VII, it will be convenient to work with a typed version of $\langle\mu\mathsf{LJ}^-\rangle$:

Definition 34 (Type assignment): Type assignment is the smallest (infix) relation ':' from terms to types satisfying:

$$\begin{aligned}N_l'\, s\, t\, \vec{x}\, 0 &\rightsquigarrow_{\mathrm{r}'} s\, \vec{x}\\ N_l'\, s\, t\, \vec{x}\, \mathsf{s}y &\rightsquigarrow_{\mathrm{r}'} t\, \vec{x}\, y\end{aligned}$$

Fig. 17. Reduction for N in $\mu'\mathsf{LJ}^-$.

- for each step $\mathsf{r}\,\dfrac{\vec{\sigma}_1 \Rightarrow \tau_1 \quad \cdots \quad \vec{\sigma}_n \Rightarrow \tau_n}{\vec{\sigma} \Rightarrow \tau}$ we have r : $(\vec{\sigma}_1 \to \tau_1) \to \cdots \to (\vec{\sigma}_n \to \tau_n) \to \vec{\sigma} \to \tau$.
- if $t : \sigma \to \tau$ and $s : \sigma$ then $ts : \tau$.
- if $t : \mu X \sigma(X)$ and $s : \sigma(\tau) \to \tau$ then $ts : \tau$.
- if $t : (\sigma \to X) \to X$, and σ positive in X, then $t : \mu X \sigma$.

We write $\langle\mu\mathsf{LJ}^-\rangle_{N,\times,\to,\mu}$ for the class of typed $\langle\mu\mathsf{LJ}^-\rangle$-terms.

Naturally, by inspection of the reduction rules of $\rightsquigarrow_{\mathrm{r}}$, we have 'subject reduction':

Proposition 35 (Type preservation): If $t : \tau$ and $t =_{\mathrm{r}} t'$ then $t' : \tau$.

By induction on typing derivations we can obtain the main result of this subsection:

Theorem 36 (Terms to derivations): $\langle\mu\mathsf{LJ}^-\rangle_{N,\times,\to,\mu}$ and $\mu\mathsf{LJ}^-$ represent the same natural number functions.

V. Totality of circular proofs

In this section we provide a semantics for (circular) proofs, using higher-order computability theoretic tools. Our aim is to show that $\mathsf{C}\mu\mathsf{LJ}$ represents only total functions on $\mathbb{N}$ (Corollary 40), by carefully extending circular proof theoretic techniques to our semantics.

Throughout this section we shall only consider types formed from $N, \times, \to, \mu$, unless otherwise indicated.

A. A type structure of regular coterms

We shall define a type structure whose domain will be contained within $\langle\mathsf{C}\mu\mathsf{LJ}^-\rangle$. Before that, it will be convenient to have access to a notion of a 'good' set of terms.

A *(totality) candidate* is some $A \subseteq \langle\mathsf{C}\mu\mathsf{LJ}^-\rangle$ that is closed under $=^{\eta}_{\mathrm{r}'}$. We henceforth expand our language of types by including each candidate A as a type constant.

An immediate albeit powerful observation is that the class of candidates forms a complete lattice under set inclusion. This justifies the following definition of our type structure:

Definition 37 (Type structure): For each type σ we define $|\sigma| \subseteq \langle\mathsf{C}\mu\mathsf{LJ}^-\rangle$ by:

$$\begin{aligned}|A| &:= A\\ |N| &:= \{t \mid \exists n \in \mathbb{N}.\, t =^{\eta}_{\mathrm{r}'} \underline{n}\}\\ |\sigma \times \tau| &:= \{t \mid \mathsf{p}_0 t \in |\sigma| \text{ and } \mathsf{p}_1 t \in |\tau|\}\\ |\sigma \to \tau| &:= \{t \mid \forall s \in |\sigma|.\, ts \in |\tau|\}\\ |\mu X \sigma(X)| &:= \bigcap\{A \text{ a candidate} \mid |\sigma(A)| \subseteq A\}\end{aligned}$$

We write $|\sigma(\cdot)|$ for the function on candidates $A \mapsto |\sigma(A)|$. As we shall see, the interpretation of μ-types above is indeed a least fixed point of the corresponding operation on candidates.

A natural, but important, property is:

Proposition 38 (Closure under conversion): If $t \in |\tau|$ and $t =^{\eta}_{\mathrm{r}'} t'$ then $t' \in |\tau|$.

Let us point out that this immediately entails, by contraposition and symmetry of $=^{\eta}_{\mathrm{r}'}$, closure of *non-elementhood* of the type structure under conversion: if $t \notin |\tau|$ and $t =^{\eta}_{\mathrm{r}'} t'$ then also $t' \notin |\tau|$. This simple observation is important for our main totality argument, which proceeds by contradiction.

The main result of this section is:

Theorem 39 (Interpretation): For any $\mathsf{C}\mu\mathsf{LJ}^-$-coderivation $P : \Sigma \Rightarrow \tau$ and $\vec{s} \in |\Sigma|$ we have $P\vec{s} \in |\tau|$.

Corollary 40: $\mathsf{C}\mu\mathsf{LJ}^- \subseteq \langle \mathsf{C}\mu\mathsf{LJ}^- \rangle$ represents only total functions on $\mathbb{N}$ with respect to $=^{\eta}_{\mathrm{r}'}$.

Proof idea: Consider a $\mathsf{C}\mu\mathsf{LJ}^-$-coderivation $P : \vec{N} \Rightarrow N$. By the Interpretation Theorem 39 and closure under conversion, Proposition 38, we have for all $\vec{m} \in \mathbb{N}$ there is $n \in \mathbb{N}$ with $P\underline{\vec{m}} =^{\eta}_{\mathrm{r}'} \underline{n}$. ■

B. Montonicity and transfinite types

To prove our main Interpretation Theorem, we shall need to appeal to a lot of background theory on fixed point theorems, ordinals and approximants, fixed point formulas, and cyclic proof theory. In fact we will go on to formalise this argument within fragments of second-order arithmetic.

At this point it is pertinent to observe that the positivity constraint we impose for fixed point types indeed corresponds to monotonicity of the induced operation on candidates with respect to our type structure:

Lemma 41 (Monotonicity): Let $A \subseteq B$ be candidates.

1) If $\sigma(X)$ is positive in X then $|\sigma(A)| \subseteq |\sigma(B)|$;
2) If $\sigma(X)$ is negative in X then $|\sigma(B)| \subseteq |\sigma(A)|$.

These properties are proved (simultaneously) by a straightforward induction on the structure of $\sigma(X)$. Note that, by the Knaster-Tarski fixed point theorem this yields:

Proposition 42 ("Fixed points" are fixed points): $|\mu X \sigma(X)|$ is the least fixed point of $|\sigma(\cdot)|$ on candidates.

We will need to appeal to an alternative characterisation of fixed points via an inflationary construction, yielding a notion of 'approximant' that:

(a) allows us to prove the Interpretation Theorem by reduction to well-foundedness of approximants (or, rather, the ordinals that index them); and
(b) allows a logically simpler formalisation within second-order arithmetic, cf. Section VI, crucial for obtaining a tight bound on representable functions,

Definition 43 (Approximants): Let F be a monotone operation on candidates, with respect to $\subseteq$. For ordinals α we define $F^\alpha(A)$ by transfinite induction on α:[19]

- $F^0(A) := \varnothing$
- $F^{\mathsf{s}\alpha}(A) := F(F^\alpha(A))$
- $F^\lambda(A) := \bigcup\limits_{\alpha<\lambda} F^\alpha(A)$, when λ is a limit ordinal.

Writing Ord for the class of all ordinals, the following is well-known:

Proposition 44 (Fixed points via approximants): Let μF be the least fixed point of a monotone operation F. We have $\mu F = \bigcup\limits_{\alpha \in \mathrm{Ord}} F^\alpha(\varnothing)$.

[19]In fact, for our purposes we will only need the special case of the definition when $A = \varnothing$.

From here it is convenient to admit formal type expressions representing approximants. We henceforth expand the language of types to be closed under:[20]

- for $\sigma(X)$ positive in X, α an ordinal, $\sigma^\alpha(\tau)$ is a type.

Definition 45 (Type structure, continued): We duly expand Definition 37 to account for transfinite types by setting $|\sigma^\alpha(\cdot)| := |\sigma(\cdot)|^\alpha$.

We have immediately from Proposition 44:

Corollary 46: $|\mu X \sigma(X)| = \bigcup\limits_{\alpha \in \mathrm{Ord}} |\sigma^\alpha(\varnothing)|$.

C. Ordinal assignments

Let us write $\tau < \sigma$ if σ has higher *priority* than τ, in the usual sense of fixed point logics, e.g., [17], [36], [39]: either σ has *Fischer-Ladner* class strictly larger than τ, or they are in the same class and σ is a subformula of τ.

Let τ be a type whose $<$-greatest fixed point subformula occurring in *positive* position is $\mu X \sigma(X)$. We write τ^α for $\tau[\sigma^\alpha(\varnothing)/\mu X \sigma(X)]$, i.e. τ with each occurrence of $\mu X \sigma(X)$ in positive position replaced by $\sigma^\alpha(\varnothing)$. τ_α is defined the same way by for the $<$-greatest fixed point subformula occurring in *negative* position.

Note that, if τ has n fixed point subformulas in positive position and $\vec{\alpha} = \alpha_1, \dots, \alpha_n$ then $\tau^{\vec{\alpha}} = (\cdots((\tau^{\alpha_1})^{\alpha_2})\cdots)$ is a positive-μ-free (transfinite) type. Similarly for negatively occurring fixed points. We shall call such sequences *(positive or negative) assignments* (respectively).

We shall order assignments by a lexicographical product order, i.e. by setting $\vec{\alpha} < \vec{\beta}$ when there is some i with $\alpha_i < \beta_i$ but $\alpha_j = \beta_j$ for all $j < i$.[21]

By the Monotonicity Lemma 41 we have:

Proposition 47 (Positive and negative approximants): If $t \in |\tau|$ there are (least) ordinal(s) $\vec{\alpha}$ s.t. $t \in |\tau^{\vec{\alpha}}|$.

Dually, if $t \notin |\tau|$ there are (least) ordinal(s) $\vec{\alpha}$ s.t. $t \notin |\tau_{\vec{\alpha}}|$.

D. Reflecting non-totality in rules of $\mu'\mathsf{LJ}^-$

Before giving our non-total branch construction, we first need a local definition that will facilitate our construction:

Definition 48 (Reflecting non-totality): Fix a $\mu'\mathsf{LJ}^-$-step,

$$\mathsf{r}\,\frac{\Sigma_0 \Rightarrow \tau_0 \quad \cdots \quad \Sigma_{n-1} \Rightarrow \tau_{n-1}}{\Sigma \Rightarrow \tau}$$

for some $n \leq 2$ and regular coderivations $P_i : \Sigma_i \Rightarrow \tau_i$.

For $\vec{s} \in |\Sigma|$ s.t. $\mathsf{r}\vec{P}\vec{s} \notin |\tau|$, we can choose a premiss $\Sigma' \Rightarrow \tau'$ and $P' : \Sigma' \Rightarrow \tau'$ and some inputs $\vec{s}' \in |\Sigma'|$ s.t. $P'\vec{s}' \notin |\tau'|$.

The $\rightarrow$ cases for the Definition above are particularly subtle, requiring consideration of least ordinal assignments similar to the handling of $\vee$-left in fixed point modal logics, cf. e.g. [28]. Full details for this definition can be found in [10].

[20]Again we shall only need the special case of $\tau = \varnothing$ for our purposes.

[21]Note that this renders the order type of $\vec{\alpha}$ simply $\alpha_n \times \cdots \times \alpha_1$, but it will be easier to explicitly work with the lexicographical order.

E. *Non-total branch construction: proof of Theorem 39*

From here the proof of the Interpretation Theorem 39 proceeds by contradiction, as is usual in cyclic proof theory. The Definition above is used to construct an infinite 'non-total' branch, along which there must be a progressing thread. We assign ordinals approximating the critical fixed point formula and positive formulas of higher priority, which must always be present as positive subformulas along the thread.[22] By the construction of Theorem 48, we note that this ordinal assignment sequence is non-increasing; moreover at any μ_l' step on the critical fixed point, the corresponding ordinal must be a successor and strictly decreases. Thus the ordinal sequence does not converge, yielding a contradiction.

VI. Some reverse mathematics of Knaster-Tarski

In order to obtain sub-recursive upper bounds on the functions represented by $\mathsf{C}\mu\mathsf{LJ}$, we will need to formalise the totality argument of the previous section itself within fragments of 'second-order' arithmetic. To this end we will need to formalise some of the reverse mathematics about fixed point theorems on which our type structure $|\cdot|$ relies.

First let us note that we have an analogue of 'functoriality' in predicate logic, by structural induction on positive formulas:

Lemma 49 (Monotonicity): Let $\varphi(X, x)$ be positive in X.

$$\vdash \forall x(Xx \to Yy) \to \forall x(\varphi(X, x) \to \varphi(Y, x))$$

In what follows this will often facilitate arguments by allowing a form of 'deep inference' reasoning.

A. *Subsystems of second-order arithmetic*

We shall work with subtheories of full second-order arithmetic (PA2) such as $\mathsf{ACA}_0, \Pi^1_1\text{-}\mathsf{CA}_0, \Pi^1_2\text{-}\mathsf{CA}_0$ etc., whose definitions may be found in standard textbooks, e.g. [35]. We shall freely use basic facts about them.

A simple consequence of Σ^1_2-choice in $\Pi^1_2\text{-}\mathsf{CA}_0$ is that Σ^1_2 (and Π^1_2) is provably closed under positive Σ^1_1 (resp. Π^1_1) combinations (even with Σ^1_1 and Π^1_1 parameters). We shall actually need a refinement of this fact to take account of polarity, so we better state it here. First let us set up some notation.

Definition 50 (Polarised analytical hierarchy): We write $\Sigma^{1,+}_n(\vec{X}, \neg\vec{Y})$ for the class of Σ^1_n formulas positive in $\vec{X}$ and negative in $\vec{Y}$. Similarly for $\Pi^{1,+}_n(\vec{X}, \neg\vec{Y})$. φ is $\Delta^{1,+}_n(\vec{X}, \neg\vec{Y})$, in a theory T, if it is T-provably equivalent to both a $\Sigma^{1,+}_n(\vec{X}, \neg\vec{Y})$-formula and a $\Pi^{1,+}_n(\vec{X}, \neg\vec{Y})$-formula.

Lemma 51 (Polarised substitution lemma, $\Pi^1_2\text{-}\mathsf{CA}_0$): If $\varphi(X) \in \Sigma^{1,+}_1(X, \vec{X}, \neg\vec{Y})$ and $\psi \in \Sigma^{1,+}_2(\vec{X}, \neg\vec{Y})$ then $\varphi(\psi) \in \Sigma^{1,+}_2(\vec{X}, \neg\vec{Y})$. Similarly for Π in place of Σ.

Proof idea: By induction on the structure of φ, using Σ^1_2-AC at each alternation of a FO and SO quantifier. ■

Note that the Lemma above, in particular, allows for arbitrary substitutions of Σ^1_1 and Π^1_1 formulas free of $\vec{X}, \vec{Y}$, which we shall rely on implicitly in the sequel.

[22]Positive here with respect to the LHS, i.e. negative if on the RHS.

We shall also assume basic facts about orders and comparison within second-order arithmetic. We write $\mathrm{WO}(\alpha)$ for a (Π^1_1) formula stating that α is a (countable) well-order, $<_\alpha$ for comparison within α, and $\preceq$ for comparison of well-orders (see, e.g., [35]).

B. *Knaster-Tarski theorem and approximants*

Throughout this section let $\varphi(X, x) \in \Delta^{1,+}_2(X, \vec{X}, \neg\vec{Y})$. We shall typically ignore/suppress the parameters $\vec{X}, \vec{Y}$, focussing primarily on X and x, and sometimes even write φ instead of $\varphi(X, x)$. We shall work within $\Pi^1_2\text{-}\mathsf{CA}_0$ unless otherwise stated.

Remark 52 (Reverse mathematics of fixed point theorems): Let us point out that, while previous work on the reverse mathematics of fixed point theorems exist in the literature, e.g. [29], even for the Knaster-Tarski theorem [33], these results apply to situations when the lattice or space at hand is countable (a subset of $\mathbb{N}$). Here we require a version of the theorem where sets themselves are elements of the lattice, under inclusion.

While we can define (bounded) approximants along any well-order simply by appeal to ATR, it will be helpful to retain the well-order (and other free set variables) as a parameter of an explicit formula to be bound later in comprehension instances, and so we give the constructions explicitly here.

Definition 53 ((Bounded) approximants): If $\mathrm{WO}(\alpha)$ and $a \in \alpha$ we write $(\Lambda X \lambda x\, \varphi)^\alpha(a, x)$ (or even $\varphi^\alpha(a, x)$) for:

$$\exists F \subseteq \alpha \times \mathbb{N}\ (\forall b \in \alpha \forall y(Fby \to \exists c <_\alpha b\, \varphi(Fc, y)) \wedge Fax)$$

We also write,

$$\begin{aligned}(\Lambda X \lambda x\, \varphi)^\alpha(x) &:= \exists a \in \alpha\, (\Lambda X \lambda x\, \varphi)^\alpha(a, x)\\ (\Lambda X \lambda x\, \varphi)^{\mathrm{WO}}(x) &:= \exists \alpha(\mathrm{WO}(\alpha) \wedge (\Lambda X \lambda x\, \varphi)^\alpha(x))\end{aligned}$$

sometimes written simply $\varphi^\alpha(x)$ and $\varphi^{\mathrm{WO}}(x)$ resp.

In this section we shall mostly use the succinct notations φ^α and φ^{WO}, being more suggestive of earlier notations from Section V. However note that proper book-keeping for variable abstraction is required to model fixed point dependencies when actually formalising the type structure $|\cdot|$.

Proposition 54: If $\mathrm{WO}(\alpha)$ then φ^α is $\Delta^{1,+}_2(\vec{X}, \neg\vec{Y})$. In particular, $\varphi^\alpha(a, x)$ is equivalent to:

$$\forall G \subseteq \alpha \times \mathbb{N}(\forall b \in \alpha \forall y(\exists c <_\alpha b\, \varphi(Gc, y) \to Gby) \to Gax) \tag{4}$$

Eventually we will also show that $\varphi^{\mathrm{WO}} \in \Delta^{1,+}_2(\vec{X}, \neg\vec{Y})$ too, but we shall need to prove the fixed point theorem first, in order to appeal to a duality property. First let us note a consequence of the above proposition:

Corollary 55 ((Bounded) recursion): Suppose $\alpha \preceq \beta$.

1) (Bounded recursion) $\varphi^\alpha(a) = \bigcup\limits_{b <_\alpha a} \varphi(\varphi^\alpha(b))$. I.e.

$$\varphi^\alpha(a, x) \leftrightarrow \exists b <_\alpha a\, \varphi(\varphi^\alpha(b), x)$$

2) (Recursion) $\varphi^\alpha = \bigcup\limits_{\beta \prec \alpha} \varphi(\varphi^\beta)$, i.e.

$$\varphi^\alpha(x) \leftrightarrow \exists \beta \prec \alpha\, \varphi(\varphi^\beta, x)$$

Proposition 56 ((Bounded) approximants are inflationary): Let $\alpha, \beta \in \mathrm{WO}$. We have the following:

1) $a \leq_\alpha b \to \forall x(\varphi^\alpha(a,x) \to \varphi^\alpha(b,x)))$
2) $\alpha \preceq \beta \to \forall x(\varphi^\alpha(x) \to \varphi^\beta(x))$

Definition 57 (Least (pre)fixed points): Define $(\mu X \lambda x \varphi)(y)$ (or even $(\mu\varphi)(y)$) by:

$$(\mu X \lambda x \varphi)(y) := \forall X(\forall x(\varphi(X,x) \to Xx) \to Xy)$$

Note that we may treat $\mu\varphi$ as a set in Π^1_2-CA_0 since $\varphi \in \Delta^1_2$, and so $(\mu\varphi)(y)$ is Π^1_2. By mimicking a text-book proof of the Knaster-Tarski theorem we obtain:

Proposition 58 ('Knaster-Tarski'): $\varphi(\mu\varphi) = \mu\varphi$, i.e.

$$\forall x(\varphi(\mu\varphi, x) \to (\mu\varphi)(x))$$

The main result of this subsection is:

Theorem 59 (LFP dual characterisation): $\mu\varphi = \varphi^{\mathrm{WO}}$, i.e.:

$$\forall x\left((\mu\varphi)(x) \leftrightarrow \varphi^{\mathrm{WO}}(x)\right)$$

One of the main consequences of the Characterisation Theorem is:

Corollary 60 (Π^1_2-CA_0): $\mu X \lambda x \varphi$ is $\Delta^{1,+}_2(\vec{X}, \neg\vec{Y})$.

Let us point out that the above result can be seen as a partial arithmetisation of some purely descriptive results about μ-definable sets in Lubarsky's work [24].

C. Arithmetising the totality argument

Thanks to the results of this section, we may duly formalise the type structure $|\cdot|$ from Section V within Π^1_2-CA_0 and prove basic properties. In particular, by induction on type we have:

Corollary 61: Let σ be positive in $\vec{X}$ and negative in $\vec{Y}$. $|\sigma|$ is Π^1_2-CA_0-provably $\Delta^{1,+}_2(\vec{X}, \neg\vec{Y})$.

We may formalise the entire totality argument of Section V within Π^1_2-CA_0, in a similar fashion to analogous arguments in [11], [13], [23], only peculiarised to the current setting. This requires some bespoke arithmetical arguments and properties of our type structure. As a consequence we obtain:

Theorem 62: Any $\mathsf{C}\mu\mathsf{LJ}^-$-representable function on natural numbers is provably recursive in Π^1_2-CA_0.

VII. Realisability with fixed points

So far we have shown an *upper bound* on the representable functions of $\mu\mathsf{LJ}$ and $\mathsf{C}\mu\mathsf{LJ}$, namely that they are all provably recursive in Π^1_2-CA_0. In this section we turn our attention to proving the analogous *lower bound*, namely by giving a *realisability* interpretation into $\mu\mathsf{LJ}$ (in fact typed-$\langle\mu\mathsf{LJ}^-\rangle$) from a theory over which Π^1_2-CA_0 is conservative.

In particular, we consider a version of first-order arithmetic, $\mu\mathsf{PA}$, with native fixed point operators, (essentially) introduced by Möllerfeld in [27]. Unlike that work, we shall formulate $\mu\mathsf{PA}$ in a purely first-order fashion in order to facilitate the ultimate realisability interpretation.

Let us write $\mathcal{L}_1(\vec{X})$ for the language of (first-order) arithmetic augmented by predicate symbols $\vec{X}$, which we shall refer to as 'variables'. $\mathcal{L}_\mu$-formulas are generated just like $\mathcal{L}_1$-formulas with the additional clause:

- if φ is a formula with free variables $X, \vec{X}, x, \vec{x}$, and in which X occurs positively, and t is a (number) term with free variables $\vec{y}$ then $t \in \mu X \lambda x \varphi$ (sometimes $(\mu X \lambda x \varphi)(t)$) is a formula with free variables $\vec{X}, \vec{x}, \vec{y}$.

Semantically we construe $\mu X \lambda x \varphi$ in the standard model as a bona fide least fixed point of the operator defined by φ.

A. Theories $\mu\mathsf{PA}$ and $\mu\mathsf{HA}$

Let us expand Peano Arithmetic (PA) to the language $\mathcal{L}_\mu$, i.e. by including induction instances for all $\mathcal{L}_\mu$-formulas. The theory we consider here is equivalent to (the first-order part of) Möllerfeld's $\mathsf{ACA}_0(\mathcal{L}_\mu)$.

Definition 63 (Theory): The theory $\mu\mathsf{PA}$ is the extension of PA by the following axioms for formulas $\varphi(X,x)$, $\psi(x)$:

(pre) $\forall y(\varphi(\mu X \lambda x \varphi, y) \to y \in \mu X \lambda x \varphi)$
(ind) $\forall x(\varphi(\psi, x) \to \psi(x)) \to (y \in \mu X \lambda x \varphi \to \psi(y))$

We may construe $\mu\mathsf{PA}$ as a proper fragment of full second-order arithmetic PA2 by the interpretation:

$$t \in \mu X \lambda x \varphi \quad := \quad \forall X(\forall x(\varphi(X,x) \to Xx) \to Xt) \qquad (5)$$

The axioms above are readily verified by mimicking a standard textbook algebraic proof of Knaster-Tarski in the logical setting, cf. Proposition 58.

Möllerfeld's main result was that Π^1_2-CA_0 is in fact Π^1_1-conservative over his theory $\mathsf{ACA}_0(\mathcal{L}_\mu)$, and so we have:

Theorem 64 (Implied by [27]): Π^1_2-CA_0 is arithmetically conservative over $\mu\mathsf{PA}$.

We set $\mu\mathsf{HA}$ to be the intuitionistic counterpart of $\mu\mathsf{PA}$. I.e. $\mu\mathsf{HA}$ is axiomatised by Heyting Arithmetic HA (extended to the language $\mathcal{L}_\mu$) and the schemes (pre) and (ind) in Definition 63 above. We shall assume that $\mu\mathsf{HA}$ uses only the logical symbols $\wedge, \to, \exists, \forall, \mu \cdot \lambda \cdot$.

Once again, we may construe $\mu\mathsf{HA}$ as a proper fragment of full second-order Heyting Arithmetic HA2 by (5) above. By essentially specialising known conservativity results for second-order arithmetic, we may thus show that $\mu\mathsf{PA}$ and $\mu\mathsf{HA}$ provably define the same functions on natural numbers.

Proposition 65 (Implied by [38]): $\mu\mathsf{PA}$ (and so also Π^1_2-CA_0) is Π^0_2-conservative over $\mu\mathsf{HA}$.

B. Realisability judgement

In what follows we shall work with the untyped calculus $\langle\mu\mathsf{LJ}^-\rangle$ and its typed version $\langle\mu\mathsf{LJ}^-\rangle_{N,\times,\to,\mu}$ from Section IV. For convenience, we identify numerals of these calculi with numerals of the language of arithmetic.

A *(realisability) candidate* is an (infix) relation $\cdot A \cdot$ relating a (untyped) term to a natural number such that $\cdot An$ is always closed under $=_{\mathrm{r}}$. Let us expand $\mathcal{L}_\mu$ by a (unary) predicate symbol for each realisability candidate. The idea is that each $\cdot An$ consists of 'just the realisers' of the sentence $A\underline{n}$.

Definition 66 (Realisability judgement): For each term $t \in \langle\mu\mathsf{LJ}^-\rangle$ and closed formula φ we define the (meta-level) judgement $t\,\mathbf{r}\,\varphi$ as follows:[23]

[23]Recall that closed terms of arithmetic always evaluate to numerals. We shall identify closed terms with their evaluation throughout this section to simplify the exposition. This also avoids any confusion arising from metavariable clash between terms of arithmetic and terms of $\langle\mu\mathsf{LJ}^-\rangle$.

- $t\,\mathbf{r}\,\underline{m}=\underline{n}$ if $\underline{m} =_{\mathrm{r}} t =_{\mathrm{r}} \underline{n}$.
- $t\,\mathbf{r}\,\varphi_0 \wedge \varphi_1$ if $\mathsf{p}_0 t\,\mathbf{r}\varphi_0$ and $\mathsf{p}_1 t\,\mathbf{r}\,\varphi_1$.
- $t\,\mathbf{r}\,A\underline{n}$ if tAn.
- $t\,\mathbf{r}\,\exists x\varphi(x)$ if there is $n\in\mathbb{N}$ with $\mathsf{p}_0 t =_{\mathrm{r}} \underline{n}$ and $\mathsf{p}_1 t\,\mathbf{r}\,\varphi(\underline{n})$
- $t\,\mathbf{r}\,\varphi\to\psi$ if, whenever $s\,\mathbf{r}\,\varphi$, we have $ts\,\mathbf{r}\,\psi$.
- $t\,\mathbf{r}\,\forall x(\varphi(x)\to\psi(x))$ if whenever $u\,\mathbf{r}\,(\underline{n}=\underline{n}\wedge\varphi(\underline{n}))$ we have $tu\,\mathbf{r}\,\psi(\underline{n})$.
- $t\,\mathbf{r}\,\forall x\varphi(x)$, where φ is not a $\to$-formula, if for all $n\in\mathbb{N}$ we have $t\underline{n}\,\mathbf{r}\,\varphi(\underline{n})$
- $t\,\mathbf{r}\,\underline{n}\in\mu X\lambda x\varphi(X,x)$ if $t\,\mathbf{r}\,\forall x(\varphi(A,x)\to Ax)\to A\underline{n}$ for all candidates A.

Definition 67 (Realising type): The *realising type* of a (possibly open) formula φ without candidate symbols, written $\mathtt{t}(\varphi)$, is given by:

- $\mathtt{t}(s=t) := N$
- $\mathtt{t}(Xt) := X$
- $\mathtt{t}(\varphi\wedge\psi) := \mathtt{t}(\varphi)\times\mathtt{t}(\psi)$
- $\mathtt{t}(\varphi\to\psi) := \mathtt{t}(\varphi)\to\mathtt{t}(\psi)$
- $\mathtt{t}(\exists x\varphi) := N\times\mathtt{t}(\varphi)$
- $\mathtt{t}(\forall x(\varphi\to\psi)) := (N\times\mathtt{t}(\varphi))\to\mathtt{t}(\psi)$
- $\mathtt{t}(\forall x\varphi) := N\to\mathtt{t}(\varphi)$ if φ not a $\to$-formula.
- $\mathtt{t}(t\in\mu X\lambda x\varphi) := \mu X(N\times\mathtt{t}(\varphi))$

Note that there are some peculiarities of the notions of realisability judgement and type above, in particular the handling of the $\forall$-case(s). The main reason for this is that there is a type-mismatch between the constants iter and in in $\mu\mathsf{LJ}^-$ and the axioms for induction and pre-fixed points in $\mu\mathsf{HA}$. This arises from the fact that arithmetical quantifiers implicitly range only over N, and not other domains; this is also the reason for the N occurrence in the realising type for fixed point formulas above. An alternative approach via 'abstract realisability' (see, e.g., [37]) could also be possible, for instance as in [6] for a similar setting.

The main result of this subsection is:

Theorem 68 (Realisability): If $\mu\mathsf{HA}\vdash\varphi$ then there is a (typed) term $t:\mathtt{t}(\varphi)$ in $\langle\mu\mathsf{LJ}^-\rangle_{N,\times,\to,\mu}$ such that $t\,\mathbf{r}\,\varphi$.

The rest of this section is devoted to a proof of this result, but first let us state our desired consequence:

Corollary 69: Any provably recursive function of $\mu\mathsf{PA}$ is representable in $\langle\mu\mathsf{LJ}^-\rangle_{N,\times,\to,\mu}$ (and so also $\mu\mathsf{LJ}^-$).

C. Some properties of the realisability model

We need some closure properties of our realisability model:

Lemma 70 (Closure under $=_{\mathrm{r}}$): If $t\,\mathbf{r}\,\varphi$ and $t=_{\mathrm{r}} t'$ then $t'\,\mathbf{r}\,\varphi$.

Note that, despite its simplicity, the above lemma has the following powerful consequence, as it renders the set $\{(t,n) : t\,\mathbf{r}\,\psi(n)\}$ itself a candidate:

Proposition 71: If $u\,\mathbf{r}\,\underline{n}\in\mu X\lambda x\varphi(X,x)$ then $u\,\mathbf{r}\,\forall x(\varphi(\psi,x)\to\psi(x))\to\psi(\underline{n})$.
We also have:

Lemma 72 (Functoriality lemma): If $t\,\mathbf{r}\,\psi\to\psi'$ and $\mathtt{t}(\varphi(X,\vec{X}))=\sigma(X,\vec{X})$ then for all candidates $\vec{A}$:

1) for $\sigma(X,\vec{X})$ +ive in X, $\sigma(t,\vec{X})\,\mathbf{r}\,\varphi(\psi,\vec{A})\to\varphi(\psi',\vec{A})$;
2) for $\sigma(X,\vec{X})$ -ive in X, $\sigma(t,\vec{X})t\,\mathbf{r}\,\varphi(\psi',\vec{A})\to\varphi(\psi,\vec{A})$.

D. Main realisability argument

From here we are able to complete the proof of the main Realisability Theorem 68. The axioms and rules of intuitionistic predicate logic are realised as usual and induction is realised by iter_N. The (ind) axioms are realised by $\mathsf{iter}_{N\times\sigma}$, appealing to Lemma 70 and Proposition 71. The (pre) axioms are modified slightly (adding $\underline{n}=\underline{n}$ to antecedent in conjunction) so that it is directly realised by $\mathsf{in}_{N\times\sigma}$, relying on Lemmas 70 and 72.

VIII. Extension to other fixed point logics

As promised in the introduction, our results cover more than just the systems $\mu\mathsf{LJ}$ and $\mathsf{C}\mu\mathsf{LJ}$, accounting also for linear logic systems from [2]–[5], [14]–[16], [18], [19].

We briefly recall the systems $\mu\mathsf{MALL}$ and $\mathsf{C}\mu\mathsf{MALL}$ from, say, [4],[24] $\mu\mathsf{MALL}$ is the extension of MALL by fixed point rules μ_r, μ_l (with no context in the left-premiss),[25] ν_l and ν_r (with no context in the right-premiss). We shall write $\mathsf{C}\mu\mathsf{MALL}$ for the circular version of this system, defined in a similar way to $\mathsf{C}\mu\mathsf{LJ}$ over the extension of MALL by rules μ_r, μ_l', ν_l and ν_r'. Both systems are construed as computational models in terms of their notions of cut-conversion (analogous to $=_{\mathrm{cr}}$ and $=_{\mathrm{cr}'}$ respectively). We have:

Theorem 73 (Computational equivalence): The following systems represent the same functions on natural numbers:

1) $\mu\mathsf{LJ}$
2) $\mu\mathsf{MALL}$
3) $\mathsf{C}\mu\mathsf{MALL}$
4) $\mathsf{C}\mu\mathsf{LJ}$

Proof idea: To establish the direction $1\implies 2$ we use Baelde's encoding of linear logic [2] in $\mu\mathsf{MALL}$, where modal formulas $?\sigma$ are defined by the least fixed point formula $\mu X(\bot\oplus(X\parr X)\oplus\sigma)$ and, dually, formulas $!\sigma$ are defined by the greatest fixed point formula $\nu X(1\&(X\otimes X)\&\sigma)$. Then, we define a translation $(\_)_!$ from $\mu\mathsf{LJ}$ to $\mu\mathsf{MALL}$. This translation is obtained by extending the standard call-by-name translation of intuitionistic logic into linear logic (see, e.g., [21]) which, in particular, satisfies the equations $(\sigma\to\tau)_! = !\sigma_!\multimap\tau_!$ and $(\mu X.\sigma)_! = \mu X.!\sigma_!$.

The direction $2\implies 3$ is straightforward and its proof is similar to the one for Proposition 21 (see, e.g., [17]).

Concerning the direction $3\implies 4$, we define a translation $(\_)^N$ of $\mathsf{C}\mu\mathsf{MALL}$ into $\mathsf{C}\mu\mathsf{LJ}$ similar to the one sketched in the proof of Proposition 24, which maps, in particular, the greatest fixed point $\nu X\sigma$ into a least fixed point $\neg\mu X\neg\sigma^N[\neg X/X]$.

Finally the direction $4\implies 1$ is a consequence of our grand tour series of results, cf. Figure 1. ■

IX. Conclusions

In this work we investigated the computational expressivity of fixed point logics. Our main contribution is a characterisation of the functions representable in the systems $\mu\mathsf{LJ}, \mathsf{C}\mu\mathsf{LJ}, \mu\mathsf{MALL}, \mathsf{C}\mu\mathsf{MALL}$ as the functions provably

[24] $\mathsf{C}\mu\mathsf{MALL}$ is the regular fragment of $\mu\mathsf{MALL}^\infty$ in that work.
[25] This is crucial for cut-reduction due to the underlying linearity.

recursive in the second-order theory Π^1_2-CA_0. Referring to Rathjen's ordinal notation system in [30], this means that all these systems represent just the functions computable by recursion on ordinals of $\mathfrak{T}[\omega/\theta]$. To this end we used a range of techniques from proof theory, reverse mathematics, higher-order computability and metamathematics. This contribution settles the question of computational expressivity of (circular) systems with fixed points, cf. Question 1.

In future work it would be interesting to investigate the computational expressivity of systems with only *strictly* positive fixed points, where μ, ν may only bind variables that are *never* under the left of $\to$. We suspect that such systems are computationally weaker than those we have considered here. On the arithmetical side, it would be interesting to investigate the *higher-order* reverse mathematics of the Knaster-Tarski theorem, cf. [33] and the present work.

References

[1] Jeremy Avigad and Solomon Feferman. Gödel's functional ("Dialectica") interpretation. *Handbook of proof theory*, 137:337–405, 1998.

[2] David Baelde. Least and greatest fixed points in linear logic. *ACM Trans. Comput. Log.*, 13(1):2:1–2:44, 2012.

[3] David Baelde, Amina Doumane, Denis Kuperberg, and Alexis Saurin. Bouncing threads for circular and non-wellfounded proofs: Towards compositionality with circular proofs. In Christel Baier and Dana Fisman, editors, *LICS '22, Haifa, Israel, August 2 - 5, 2022*, pages 63:1–63:13. ACM, 2022.

[4] David Baelde, Amina Doumane, and Alexis Saurin. Infinitary proof theory: the multiplicative additive case. In Jean-Marc Talbot and Laurent Regnier, editors, *CSL '16, August 29 - September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 42:1–42:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[5] David Baelde and Dale Miller. Least and greatest fixed points in linear logic. In Nachum Dershowitz and Andrei Voronkov, editors, *LPAR '07, Yerevan, Armenia, October 15-19, 2007, Proceedings*, volume 4790 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2007.

[6] Ulrich Berger and Hideki Tsuiki. Intuitionistic fixed point logic. *Annals of Pure and Applied Logic*, 172(3):102903, 2021.

[7] James Brotherston and Alex Simpson. Complete sequent calculi for induction and infinite descent. pages 51 – 62, 08 2007.

[8] Pierre Clairambault. Least and greatest fixpoints in game semantics. In Ralph Matthes and Tarmo Uustalu, editors, *FICS '09, Coimbra, Portugal, September 12-13, 2009*, pages 39–45. Institute of Cybernetics, 2009.

[9] Pierre Clairambault. Strong functors and interleaving fixpoints in game semantics. *RAIRO Theor. Informatics Appl.*, 47(1):25–68, 2013.

[10] Gianluca Curzi and Anupam Das. Computational expressivity of (circular) proofs with fixed points, 2023.

[11] Anupam Das. A circular version of Gödel's T and its abstraction complexity. *CoRR*, abs/2012.14421, 2020.

[12] Anupam Das. On the logical complexity of cyclic arithmetic. *Logical Methods in Computer Science*, Volume 16, Issue 1, January 2020.

[13] Anupam Das. On the logical strength of confluence and normalisation for cyclic proofs. In Naoki Kobayashi, editor, *FSCD '21, July 17-24, 2021, Buenos Aires, Argentina (Virtual Conference)*, volume 195 of *LIPIcs*, pages 29:1–29:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[14] Abhishek De, Farzad Jafar-Rahmani, and Alexis Saurin. Phase semantics for linear logic with least and greatest fixed points. In Anuj Dawar and Venkatesan Guruswami, editors, *FSTTCS '22, December 18-20, 2022, IIT Madras, Chennai, India*, volume 250 of *LIPIcs*, pages 35:1–35:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[15] Abhishek De, Luc Pellissier, and Alexis Saurin. Canonical proof-objects for coinductive programming: infinets with infinitely many cuts. In Niccolò Veltri, Nick Benton, and Silvia Ghilezan, editors, *PPDP '21, Tallinn, Estonia, September 6-8, 2021*, pages 7:1–7:15. ACM, 2021.

[16] Abhishek De and Alexis Saurin. Infinets: The parallel syntax for non-wellfounded proof-theory. In Serenella Cerrito and Andrei Popescu, editors, *TABLEAUX '19, London, UK, September 3-5, 2019, Proceedings*, volume 11714 of *Lecture Notes in Computer Science*, pages 297–316. Springer, 2019.

[17] 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.

[18] Thomas Ehrhard and Farzad Jafarrahmani. Categorical models of linear logic with fixed points of formulas. In *LICS '21*, New York, NY, USA, 2021. Association for Computing Machinery.

[19] Thomas Ehrhard, Farzad Jafarrahmani, and Alexis Saurin. On relation between totality semantic and syntactic validity. In *TLLA '21*, Rome (virtual), Italy, June 2021.

[20] Jean-Yves Girard. *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur*. PhD thesis, Éditeur inconnu, 1972.

[21] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

[22] J. Roger Hindley and Jonathan P. Seldin. *Lambda-Calculus and Combinators: An Introduction*. Cambridge University Press, 2 edition, 2008.

[23] 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.

[24] Robert S. Lubarsky. μ-definable sets of integers. *The Journal of Symbolic Logic*, 58(1):291–313, 1993.

[25] Nax Paul Mendler. Recursive types and type constraints in second-order lambda calculus. In *Logic in Computer Science*, 1987.

[26] Nax Paul Mendler. Inductive types and type constraints in the second-order lambda calculus. *Annals of Pure and Applied Logic*, 51(1):159–172, 1991.

[27] Michael Möllerfeld. *Generalized inductive definitions. The μ-calculus and Π^1_2-comprehension*. PhD thesis, 2002. University of Münster, https://nbn-resolving.de/urn:nbn:de:hbz:6-85659549572.

[28] Damian Niwinski and Igor Walukiewicz. Games for the mu-calculus. *Theor. Comput. Sci.*, 163(1&2):99–116, 1996.

[29] Weiguang Peng and Takeshi Yamazaki. Two kinds of fixed point theorems and reverse mathematics. *Mathematical Logic Quarterly*, 63(5):454–461, 2017.

[30] Michael Rathjen. Recent advances in ordinal analysis: Π^1_2-CA and related systems. *Bulletin of Symbolic Logic*, 1(4):468–485, 1995.

[31] Michael Rathjen and Wilfried Sieg. Proof Theory. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2022 edition, 2022.

[32] John C. Reynolds. Towards a theory of type structure. In Bernard J. Robinet, editor, *Programming Symposium, Proceedings Colloque sur la Programmation, Paris, France, April 9-11, 1974*, volume 19 of *Lecture Notes in Computer Science*, pages 408–423. Springer, 1974.

[33] Takashi Sato and Takeshi Yamazaki. Reverse mathematics and order theoretic fixed point theorems. *Arch. Math. Log.*, 56(3-4):385–396, 2017.

[34] Alex Simpson. Cyclic arithmetic is equivalent to peano arithmetic. In Javier Esparza and Andrzej S. Murawski, editors, *FOSSACS '17, Held as Part of ETAPS '17, Uppsala, Sweden, April 22-29, 2017, Proceedings*, volume 10203 of *Lecture Notes in Computer Science*, pages 283–300, 2017.

[35] Stephen G. Simpson. *Subsystems of second order arithmetic*. Perspectives in mathematical logic. Springer, 1999.

[36] Thomas Studer. On the proof theory of the modal mu-calculus. *Studia Logica: An International Journal for Symbolic Logic*, 89(3):343–363, 2008.

[37] A.S. Troelstra. Chapter VI - Realizability. In Samuel R. Buss, editor, *Handbook of Proof Theory*, volume 137 of *Studies in Logic and the Foundations of Mathematics*, pages 407–473. Elsevier, 1998.

[38] Sergei Tupailo. On the intuitionistic strength of monotone inductive definitions. *J. Symb. Log.*, 69(3):790–798, 2004.

[39] Yde Venema. Lectures on the modal μ-calculus. *Renmin University in Beijing (China)*, 2008.