Logical Methods in Computer Science
Volume 21, Issue 4, 2025, pp. 18:1–18:58
https://lmcs.episciences.org/

Submitted May 28, 2024
Published Nov. 04, 2025

ON THE COMPUTATIONAL EXPRESSIVITY OF (CIRCULAR) PROOFS WITH FIXED POINTS

GIANLUCA CURZI [a] AND ANUPAM DAS [b]

[a] University of Gothenburg and University of Birmingham
e-mail address: gianluca.curzi@gu.se

[b] University of Birmingham
e-mail address: 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 propositional 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 recursive 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 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 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.

1. 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

This work was supported by a UKRI Future Leaders Fellowship, 'Structure vs Invariants in Proofs' (project reference MR/S035540/1), by the Wallenberg Academy Fellowship Prolongation project 'Taming Jörmungandr: The Logical Foundations of Circularity' (project reference 251080003), and by the VR starting grant "Proofs with Cycles in Computation" (project reference 251088801).

DOI:10.46298/LMCS-21(4:18)2025

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 [NW96]. 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* [BDS16, EJ21, EJS21, BDKS22, DS19, DPS21, DJS22] after foundational work on related finitary systems in [BM07, Bae12]. 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 1.1. 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 [Cla09], 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., [RS22]). More recently, inspired by Lubarsky's work on 'μ-definable sets' [Lub93], Möllerfeld has notably classified the proof theoretic strength of extensions of PA by general inductive definitions in [Mö02].

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 precisely 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 effective ordinal analysis (see [Rat95]). The best known lower bound for $\mu\mathsf{LJ}$ before was Gödel's T (see, e.g., [Cla09]), which has the same proof theoretic strength as PA. The best known upper bound was Girard-Reynold's F, thanks to its impredicative encodings of fixed points, which has the same proof theoretic strength as second-order arithmetic $\mathsf{PA2}$.

1.1. **Outline and contribution.** The structure of our overall argument is visualised in Figure 1, outlining a cycle of inclusions of 'representable functions'. Here the upper row consists of theories of arithmetic, where the representable functions of an arithmetic theory T are just its provably total recursive functions; i.e. those functions $f : \mathbb{N} \to \cdots \to \mathbb{N}$ with graph computed by some Σ^0_1 formula $\varphi_f(\vec{x}, y)$ such that $T \vdash \forall\vec{x}\exists y\varphi_f(\vec{x}, y)$. The lower row consists of type systems whose representable functions are just those admitting a typing derivation with conclusion $N \to \cdots \to N$ computing the function under its operational semantics (as in, e.g., Definition 2.11).

(1) is a standard embedding of finitary proofs into circular proofs (Proposition 3.9). (2) reduces $\mathsf{C}\mu\mathsf{LJ}$ to its 'negative fragment', in particular free of greatest fixed points (ν), via a double negation translation (Proposition 3.12).

(3) is one of our main contributions: we build a higher-order computability model $|\cdot|$ that interprets $\mathsf{C}\mu\mathsf{LJ}^-$ (Theorem 5.4), and moreover formalise this construction itself within

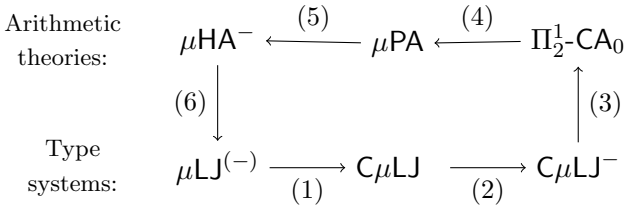

Figure 1: Summary of the main 'grand tour' of this work. All arrows indicate inclusions of representable functions.

Π^1_2-CA_0 to obtain our upper bound (Theorem 6.21). The domain of this model a priori is an (untyped) term extension of $\mathsf{C}\mu\mathsf{LJ}^-$. It is important for logical complexity 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 6.17).

(4) is an intricate and nontrivial result established by Möllerfeld in [Mö02], which we use as a 'black box'. (5) is again a double negation translation, morally a specialisation of the Π^0_2-conservativity of full second-order arithmetic $\mathsf{PA2}$ over its intuitionistic counterpart $\mathsf{HA2}$, composed with a relativisation of quantifiers to $\mathbb{N}$ (Propositions 7.5 and 7.8).

(6) is our second main contribution: we provide a realisability interpretation from $\mu\mathsf{HA}^-$ into $\mu\mathsf{LJ}$ (Theorem 7.17), morally by considerable specialisation of the analogous interpretation from $\mathsf{HA2}$ into Girard-Reynolds' system F. 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 3.12).

1.2. **Related work.** Fixed points have been studied extensively in type systems for programming languages. In particular foundational work by Mendler in the late '80s [Men87, Men91] already cast inductive type systems as fragments of second-order ones such as Girard-Reynolds' F [Gir72, Rey74]. Aside from works we have already mentioned, (a variant of) (5) has already been obtained by Tupailo in [Tup04]. Berger and Tsuiki have also obtained a similar result to (6) in a related setting [BT21], for *strictly* positive fixed points, where bound variables may never occur under the left of an arrow. 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 [Sim17, Das20b] for (cyclic) (fragments of) PA and [Das20a, Das21, KPP21] for (circular) (fragments of) Gödel's system T.

1.3. **Comparison to preliminary version.** This paper is an expansion of the preliminary conference version [CD23]. In this version we additionally include full proofs of all our results, as well as further examples and narrative.

We have reformulated our realisability argument in Section 7 into a form of *abstract* realisability, inspired by the approach of [BT21]. This factors the approach of the preliminary version by a more careful relativisation of quantifiers to deal with an inconvenient type mismatch when realising inductive predicates.

Finally, in the preliminary version we also showed equivalence of $\mu\mathsf{LJ}$ and $\mathsf{C}\mu\mathsf{LJ}$ with their counterparts in linear logic from (see, e.g., [Bae12, BDS16]) via appropriate proof interpretations. These results will be expanded upon in a separate self-contained paper.

1.4. **Notation.** 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 relexive symmetric transitive 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.

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

In this section we recall the system $\mu\mathsf{LJ}$ from [Cla09, Cla13]. More precisely, we present the 'strong' version of $\mu\mathsf{LJ}$ from [Cla13].

2.1. **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:

- $\mathrm{FV}(X) := \{X\}$
- $\mathrm{FV}(1) := \varnothing$
- $\mathrm{FV}(\sigma \star \tau) := \mathrm{FV}(\sigma) \cup \mathrm{FV}(\tau)$, for $\star \in \{+, \times, \to\}$
- $\mathrm{FV}(\kappa X\, \sigma) := \mathrm{FV}(\sigma) \setminus \{X\}$, for $\kappa \in \{\mu, \nu\}$

A pretype is *closed* if it has no free variables (otherwise it is *open*).

Throughout this work we shall assume some standard conventions on variable binding, in particular that each occurrence of a binder μ and ν binds a variable distinct from all other binder occurrences in consideration. This avoids having to deal with variable renaming explicitly. We follow usual bracketing conventions, in particular writing, say, $\rho \to \sigma \to \tau$ for $(\rho \to (\sigma \to \tau))$. Binders μX and νY bind as strongly as possible but we may write, say, $\mu X, Y.\, \sigma \to \tau$ for $\mu X \mu Y (\sigma \to \tau)$.

Definition 2.1 (Types and polarity)**.** *Positive* and *negative* variables in a pretype are defined as expected:

- X is positive in X.
- 1 is positive and negative in X.
- if σ, τ are positive (negative) in X then so is $\sigma \star \tau$, for $\star \in \{+, \times\}$.
- if σ is negative (positive) in X and τ is positive (resp., negative) in X, then $\sigma \to \tau$ is positive (resp., negative) in X.
- if σ is positive (negative) in X then so is $\kappa Y \sigma$ (resp.), for $\kappa \in \{\mu, \nu\}$, both when $Y = X$ and $Y \neq X$.

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 2.2 (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 [Cla09, Cla13] we do not impose this stronger requirement, requiring only positivity in the usual syntactic sense.

Definition 2.3 (System $\mu\mathsf{LJ}$)**.** A *cedent*, written Σ, Γ etc., is just a list of types. A *sequent* is an expression $\Sigma \Rightarrow \sigma$. The symbol $\Rightarrow$ is, formally, just a syntactic delimiter (but the arrow notation is suggestive). The system $\mu\mathsf{LJ}$ is given by the rules of Figures 2, 3 and 4 (colours may be ignored for now). 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 2.4 (General identity and substitutions)**.** Note that $\mu\mathsf{LJ}$ is equipped with a general identity rule, not only for atomic types. This has the apparently simple but useful consequence that typing derivations are closed under substitution of types for free variables, i.e. if $P(X) : \Gamma(X) \Rightarrow \sigma(X)$ in $\mu\mathsf{LJ}$ (with all occurrences of X indicated), then also $P(\tau) : \Gamma(\tau) \Rightarrow \sigma(\tau)$ in $\mu\mathsf{LJ}$ for any type τ. Later, this will allow us to derive inductively general functors for fixed points in $\mu\mathsf{LJ}$ rather than including them natively; this will in turn become important later for verifying our realisability model for $\mu\mathsf{LJ}$.

Remark 2.5 ($\mu\mathsf{LJ}$ as a fragment of second-order logic)**.** We may regard $\mu\mathsf{LJ}$ properly as a fragment of Girard-Reynolds System F [Gir72, Rey74], an extension of simple types to a second-order setting. In particular, (co)inductive types may be identified with second-order formulas by:

$$\mu X\sigma = \forall X((\sigma \to X) \to X)$$
$$\nu X\sigma = \exists X(X \times (X \to \sigma))$$

The rules for fixed points in $\mu\mathsf{LJ}$ are essentially inherited from this encoding, modulo some constraints on proof search strategy. 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), appealing to 'duality' for the latter. The cases for ν should be deemed analogous. As we shall soon see, in Subsection 3.4, we can indeed reduce our consideration to ν-free types, without loss of generality in terms of representable functions.

Remark 2.6 (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 2.7 (Variations of the fixed point rules)**.** It is common to consider context-free and 'weak' specialisations of the fixed point rules, e.g.:

$$\mathsf{iter}\ \frac{\sigma(\tau) \Rightarrow \tau}{\mu X\sigma(X) \Rightarrow \tau} \qquad \mathsf{coiter}\ \frac{\rho \Rightarrow \sigma(\rho)}{\rho \Rightarrow \nu X\sigma(X)} \tag{2.1}$$

In the presence of cut the '(co)iterator' rules above are equivalent to those of $\mu\mathsf{LJ}$ (see Appendix A for some further remarks). However 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.

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

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

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

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

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

Figure 2: Sequent calculus rules for LJ.

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

Figure 3: Some unfolding rules for μ and ν.

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

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

When giving a semantics that interprets cut directly, e.g. as we do for the term calculi in Section 4, it is often simpler to work with the (co)iterators above. In the remainder of this work we shall freely use the versions above in proofs too.

Definition 2.8 (Functors)**.** Let $\sigma(X)$ and $\rho(X)$ be (possibly open) types that are positive (and negative, respectively) in X. 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 as follows:

- If $\sigma(X) = X$ then $\sigma(P)$ is just P. Notice that it is never the case that $\rho = X$, as X can only occur negatively in ρ.
- If σ and ρ are 1 or some $Y \neq X$ then $\sigma(P)$ and $\tau(P)$ are defined respectively as follows:

$$\mathsf{w}\ \frac{\mathsf{id}\ \dfrac{}{\sigma \Rightarrow \sigma}}{\Gamma, \sigma \Rightarrow \sigma} \qquad \mathsf{w}\ \frac{\mathsf{id}\ \dfrac{}{\rho \Rightarrow \rho}}{\Gamma, \rho \Rightarrow \rho}$$

- If $\sigma = \sigma_1 \to \sigma_2$ and $\rho = \rho \to \rho$ then we define $\sigma(P)$ and $\rho(P)$ respectively as follows:

$$\dfrac{\dfrac{\dfrac{\sigma_1(P) \quad \sigma_2(P)}{\Gamma, \sigma_1(\tau') \Rightarrow \sigma_1(\tau) \quad \Gamma, \sigma_2(\tau) \Rightarrow \sigma_2(\tau')}\,{\to_l}}{\dfrac{\Gamma, \Gamma, \sigma(\tau), \sigma_1(\tau') \Rightarrow \sigma_2(\tau')}{\Gamma, \sigma(\tau), \sigma_1(\tau') \Rightarrow \sigma_2(\tau')}\,\mathsf{c}}}{\Gamma, \sigma(\tau) \Rightarrow \sigma(\tau')}\,{\to_r} \qquad \dfrac{\dfrac{\dfrac{\rho_1(P) \quad \rho_2(P)}{\Gamma, \rho_1(\tau) \Rightarrow \rho_1(\tau') \quad \Gamma, \rho_2(\tau') \Rightarrow \rho_2(\tau)}\,{\to_l}}{\dfrac{\Gamma, \Gamma, \rho(\tau'), \rho_1(\tau) \Rightarrow \rho_2(\tau)}{\Gamma, \rho(\tau'), \rho_1(\tau) \Rightarrow \rho_2(\tau)}\,\mathsf{c}}}{\Gamma, \rho(\tau') \Rightarrow \rho(\tau)}\,{\to_r}$$

- If $\sigma = \sigma_1 \times \sigma_2$ and $\rho = \rho_1 \times \rho_2$ then we define $\sigma(P)$ and $\rho(P)$ respectively as follows:

$$\dfrac{\dfrac{\dfrac{\sigma_1(P)}{\Gamma, \sigma_1(\tau) \Rightarrow \sigma_1(\tau')}\,{\times_l} \quad \dfrac{\sigma_2(P)}{\Gamma, \sigma_2(\tau) \Rightarrow \sigma_2(\tau')}\,{\times_l}}{\Gamma, \sigma(\tau) \Rightarrow \sigma_1(\tau') \quad \Gamma, \sigma(\tau) \Rightarrow \sigma_2(\tau')}\,{\times_r}}{\dfrac{\Gamma, \Gamma, \sigma(\tau) \Rightarrow \sigma(\tau)}{\Gamma, \sigma(\tau) \Rightarrow \sigma(\tau')}\,\mathsf{c}} \qquad \dfrac{\dfrac{\dfrac{\rho_1(P)}{\Gamma, \rho_1(\tau') \Rightarrow \rho_1(\tau)}\,{\times_l} \quad \dfrac{\rho_2(P)}{\Gamma, \rho_2(\tau') \Rightarrow \rho_2(\tau)}\,{\times_l}}{\Gamma, \rho(\tau') \Rightarrow \rho_1(\tau) \quad \Gamma, \rho(\tau') \Rightarrow \rho_2(\tau)}\,{\times_r}}{\dfrac{\Gamma, \Gamma, \rho(\tau') \Rightarrow \rho(\tau)}{\Gamma, \rho(\tau') \Rightarrow \rho(\tau)}\,\mathsf{c}}$$

- If $\sigma = \sigma_1 + \sigma_2$ and $\rho = \rho_1 + \rho_2$ then we define $\sigma(P)$ and $\rho(P)$ respectively as follows:

$$\dfrac{\dfrac{\dfrac{\sigma_1(P)}{\Gamma, \sigma_1(\tau) \Rightarrow \sigma_1(\tau')}\,{+_r} \quad \dfrac{\sigma_2(P)}{\Gamma, \sigma_2(\tau) \Rightarrow \sigma_2(\tau')}\,{+_r}}{\Gamma, \sigma_1(\tau) \Rightarrow \sigma(\tau') \quad \Gamma, \sigma_2(\tau) \Rightarrow \sigma(\tau')}\,{+_l}}{\Gamma, \sigma(\tau) \Rightarrow \sigma(\tau')} \qquad \dfrac{\dfrac{\dfrac{\rho_1(P)}{\Gamma, \rho_1(\tau') \Rightarrow \rho_1(\tau)}\,{+_r} \quad \dfrac{\rho_2(P)}{\Gamma, \rho_2(\tau') \Rightarrow \rho_2(\tau)}\,{+_r}}{\Gamma, \rho_1(\tau') \Rightarrow \rho(\tau) \quad \Gamma, \rho_2(\tau') \Rightarrow \rho(\tau)}\,{+_l}}{\Gamma, \rho(\tau') \Rightarrow \rho(\tau)}$$

- if $\sigma(X) = \mu Y \sigma'(X, Y)$ and $\rho(X) = \mu Y \rho'(X, Y)$ then we define $\sigma(P)$ and $\rho(P)$ respectively as follows:

$$\dfrac{\dfrac{\dfrac{\sigma'(P, \sigma(\tau'))}{\Gamma, \sigma'(\tau, \sigma(\tau')) \Rightarrow \sigma'(\tau', \sigma(\tau'))}\,{\mu_r}}{\Gamma, \sigma'(\tau, \sigma(\tau')) \Rightarrow \sigma(\tau')}\,{\mu_l}}{\Gamma, \sigma(\tau) \Rightarrow \sigma(\tau')} \qquad \dfrac{\dfrac{\dfrac{\rho'(P, \rho(\tau))}{\Gamma, \rho'(\tau', \rho(\tau)) \Rightarrow \rho'(\tau, \rho(\tau))}\,{\mu_r}}{\Gamma, \rho'(\tau', \rho(\tau)) \Rightarrow \rho(\tau)}\,{\mu_l}}{\Gamma, \rho(\tau') \Rightarrow \rho(\tau)}$$

  where $\sigma'(P, \sigma(\tau'))$ (resp., $\rho'(P, \rho(\tau))$) are obtained from the IH for $\sigma'(P, Y)$ (resp., $\rho'(P, Y)$) under substitution of $\sigma(\tau')$ for Y (resp., $\rho(\tau)$), cf. Remark 2.4.
- if $\sigma(X) = \nu Y \sigma'(X, Y)$ and $\rho(X) = \nu Y \rho'(X, Y)$ then we define $\sigma(P)$ and $\rho(P)$ respectively as follows:

$$\dfrac{\dfrac{\dfrac{\sigma'(P, \sigma(\tau))}{\Gamma, \sigma'(\tau, \sigma(\tau)) \Rightarrow \sigma'(\tau', \sigma(\tau))}\,{\nu_l}}{\Gamma, \sigma(\tau) \Rightarrow \sigma'(\tau', \sigma(\tau))}\,{\nu_r}}{\Gamma, \sigma(\tau) \Rightarrow \sigma(\tau')} \qquad \dfrac{\dfrac{\dfrac{\rho'(P, \rho(\tau'))}{\Gamma, \rho'(\tau', \rho(\tau')) \Rightarrow \rho'(\tau, \rho(\tau'))}\,{\nu_l}}{\Gamma, \rho(\tau') \Rightarrow \rho'(\tau, \rho(\tau'))}\,{\nu_r}}{\Gamma, \rho(\tau') \Rightarrow \rho(\tau)}$$

where $\sigma'(P, \sigma(\tau))$ (resp., $\rho'(P, \rho(\tau'))$) are obtained from the IH for $\sigma'(P, Y)$ (resp., $\rho'(P, Y)$) under substitution of $\sigma(\tau)$ for Y (resp., $\rho(\tau')$), cf. Remark 2.4.

Example 2.9 (Post-fixed point)**.** *It is implicit in the rules of* $\mu\mathsf{LJ}$ *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 5). 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* $\mu'_l\dfrac{\Gamma, \sigma(\mu X\sigma(X)) \Rightarrow \tau}{\Gamma, \mu X\sigma(X) \Rightarrow \tau}$ *we may use a derivation that mimics standard textbook-style proofs of Knaster-Tarski:*

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

Dually, we can derive $\nu'_r\dfrac{\Gamma \Rightarrow \sigma(\nu X\sigma(X))}{\Gamma \Rightarrow \nu X\sigma(X)}$ *as follows:*

$$\nu_r\dfrac{\Gamma \Rightarrow \sigma(\nu X\sigma(X)) \quad \sigma\dfrac{\nu_l\dfrac{\mathsf{id}\dfrac{}{\sigma(\nu X\sigma(X)) \Rightarrow \sigma(\nu X\sigma(X))}}{\nu X\sigma(X) \Rightarrow \sigma(\nu X\sigma(X))}}{\sigma(\nu X\sigma(X)) \Rightarrow \sigma(\sigma(\nu X\sigma(X)))}}{\Gamma \Rightarrow \nu X\sigma(X)}$$

2.2. **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 2.5. Following Baelde and Miller [BM07], we give self-contained cut-reductions here:

Definition 2.10 (Cut-reduction for fixed points)**.** *Cut-reduction* on $\mu\mathsf{LJ}$-derivations, written $\rightsquigarrow_{\mathrm{cr}}$, is the smallest relation on derivations including all the usual cut-reductions of LJ and the reductions in Figure 5. As usual, we allow these reductions to be performed on sub-derivations. I.e. they are 'context-closed'.

When speaking of (subsets of) $\mu\mathsf{LJ}$ as a computational model, we always mean with respect to the relation $\rightsquigarrow^*_{\mathrm{cr}}$ unless otherwise stated. More precisely:

Definition 2.11 (Representability in $\mu\mathsf{LJ}$)**.** We define the *type of natural numbers* as $N := \mu X(1 + X)$. We also define the *numeral* $\underline{n} : N$ by induction on $n \in \mathbb{N}$:

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

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

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

$$\dfrac{\dfrac{\Delta \Rightarrow \rho \quad \Gamma, \rho \Rightarrow \sigma(\rho)}{\Delta, \Gamma \Rightarrow \nu X\sigma(X)}\,\nu_r \quad \dfrac{\Sigma, \sigma(\nu X\sigma(X)) \Rightarrow \tau}{\Sigma, \nu X\sigma(X) \Rightarrow \tau}\,\nu_l}{\Delta, \Gamma, \Sigma \Rightarrow \tau}\,\mathsf{cut} \qquad \rightsquigarrow$$

$$\dfrac{\Delta \Rightarrow \rho \qquad \dfrac{\Gamma, \rho \Rightarrow \sigma(\rho) \qquad \dfrac{\dfrac{\dfrac{}{\rho \Rightarrow \rho}\,\mathsf{id} \quad \Gamma, \rho \Rightarrow \sigma(\rho)}{\Gamma, \rho \Rightarrow \nu X\sigma(X)}\,\nu_r}{\Gamma, \sigma(\rho) \Rightarrow \sigma(\nu X\sigma(X))}\,\sigma \qquad \Sigma, \sigma(\nu X\sigma(X)) \Rightarrow \tau}{\Gamma, \Sigma, \sigma(\rho) \Rightarrow \tau}\,\mathsf{cut}}{\Gamma, \Gamma, \Sigma, \rho \Rightarrow \tau}\,\mathsf{cut}}{\dfrac{\Delta, \Gamma, \Gamma, \Sigma, \Delta \Rightarrow \tau}{\Delta, \Gamma, \Sigma \Rightarrow \tau}\,\mathsf{c}}\,\mathsf{cut}$$

Figure 5: Cut-reduction rules for μ and ν in $\mu\mathsf{LJ}$.

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

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

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

Example 2.12 (Native rules for natural number computation)**.** *'Native' rules for type N in $\mu\mathsf{LJ}$ are given in Figure 6, all routinely derivable in $\mu\mathsf{LJ}$, as in Figure 7. The corresponding 'native' cut-reductions, derivable using $\rightsquigarrow_{\mathrm{cr}}$ cf. Figure 5, are also routine. We shall examine*

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

Figure 6: Native rules for N in $\mu\mathsf{LJ}$.

$$N_r^0 := \mu_r\,\frac{+_r^0\,\frac{1_r\,\frac{}{\Rightarrow 1}}{\Rightarrow 1 + N}}{\Rightarrow N} \qquad N_r^1 := \mu_r\,\frac{+_r^1\,\frac{\Gamma \Rightarrow N}{\Gamma \Rightarrow 1 + N}}{\Gamma \Rightarrow N} \qquad N_l := \mu_l\,\frac{+_l\,\frac{1_l\,\frac{\Gamma \Rightarrow \sigma}{\Gamma, 1 \Rightarrow \sigma} \quad \Gamma, \sigma \Rightarrow \sigma}{\Gamma, 1 + \sigma \Rightarrow \sigma} \quad \Delta, \sigma \Rightarrow \tau}{\Gamma, \Delta, N \Rightarrow \tau}$$

Figure 7: Constructing and destructing natural numbers in $\mu\mathsf{LJ}$.

$$\mu_r\,\frac{+_r^0\,\frac{1_r\,\frac{}{\Rightarrow 1}}{\Rightarrow 1 + (N \times L)}}{\Rightarrow L} \qquad \mu_r\,\frac{+_r^1\,\frac{\times\,\frac{\overset{n}{\triangledown} \quad \overset{l}{\triangledown}}{\Rightarrow N \quad \Rightarrow L}}{\Rightarrow N \times L}}{\Rightarrow 1 + (N \times L)}}{\Rightarrow L}$$

Figure 8: Constructing lists in $\mu\mathsf{LJ}$.

this further in Section 4. Note that, from here we can recover the usual recursor of system T*, as shown formally by Clairambault [Cla13].*

Example 2.13. *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 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 $S = \nu X(N \times X)$. Figure 8, left-to-right, shows the encoding of ε and $n :: l$ (i.e., the empty list and the operation appending a natural number to a list). Figure 9 shows the encoding of a concatenation of a list and a stream into a stream (by recursion over the list with the invariant $S \to S$).*[1]

3. A circular version of $\mu\mathsf{LJ}$

In this section we shall develop a variation of $\mu\mathsf{LJ}$ 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 3.1 ($\mu\mathsf{LJ}$ 'without (co)iteration')**.** Write $\mu'\mathsf{LJ}$ for the system of all rules in Figures 2, 3 and 10 (but not 4).

[1]Both examples were originally given for $\mu\mathsf{MALL}$ in [Dou17].

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

Figure 9: Concatenation of a list and a stream in $\mu\mathsf{LJ}$.

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

Figure 10: Further unfolding rules for μ and ν.

3.1. **'Non-wellfounded' proofs over $\mu'\mathsf{LJ}$.** '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 generated by the rules of a system.

Definition 3.2 (Coderivations)**.** A ($\mu'\mathsf{LJ}$-)*coderivation* P is a possibly infinite rooted tree (of height $\leq \omega$) generated by the rules of $\mu'\mathsf{LJ}$. Formally, we identify P with a prefix-closed subset of $\{0,1\}^*$ (i.e. a binary tree) where each node is labelled by an inference step from $\mu'\mathsf{LJ}$ such that, whenever $\alpha \in \{0,1\}^*$ is labelled by a step $\dfrac{S_1 \quad \cdots \quad S_n}{S}$, for $n \leq 2$, α has n children in P labelled by steps with conclusions $S_1, \ldots, S_n$ respectively.

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) in the natural way.

3.2. **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 adapt the theory of cut-reduction to the different fixed point rules of $\mu'\mathsf{LJ}$.

Definition 3.3 (Cut-reduction on coderivations)**.** $\rightsquigarrow_{\mathrm{cr}'}$ is the smallest relation on $\mu'\mathsf{LJ}$-coderivations including all the usual cut-reductions of LJ and the cut-reductions in Figure 11,[2]. When speaking of (subsets of) coderivations as computational models, we typically mean with respect to $\rightsquigarrow^*_{\mathrm{cr}'}$.

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

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

$$\dfrac{\nu'_r\dfrac{\Gamma \Rightarrow \sigma(\nu X\sigma(X))}{\Gamma \Rightarrow \nu X\sigma(X)} \quad \nu_l\dfrac{\Delta, \sigma(\nu X\sigma(X))}{\Delta, \nu X\sigma(X) \Rightarrow \tau}}{\Gamma, \Delta \Rightarrow \tau}\ \mathsf{cut} \quad \rightsquigarrow \quad \dfrac{\Gamma \Rightarrow \sigma(\nu X\sigma(X)) \quad \Delta, \sigma(\nu X\sigma(X)) \Rightarrow \tau}{\Gamma, \Delta \Rightarrow \tau}\ \mathsf{cut}$$

Figure 11: Cut-reduction for least and greatest fixed points in $\mathsf{C}\mu\mathsf{LJ}$.

Example 3.4 (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:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Gamma, \mu X\sigma(X) \Rightarrow \rho}\,\mu'_l\bullet}{\Gamma, \sigma(\mu X\sigma(X)) \Rightarrow \sigma(\rho)}\,\sigma \quad \Gamma, \sigma(\rho) \Rightarrow \rho}{\Gamma, \Gamma, \sigma(\mu X\sigma(X)) \Rightarrow \rho}\,\mathsf{cut}}{\Gamma, \sigma(\mu X\sigma(X)) \Rightarrow \rho}\,\mathsf{c}}{\Gamma, \mu X\sigma(X) \Rightarrow \rho}\,\mu'_l\bullet \qquad \Delta, \rho \Rightarrow \tau}{\Gamma, \Delta, \mu X\sigma(X) \Rightarrow \tau}\,\mathsf{cut} \tag{3.1}$$

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

$$\dfrac{\Delta \Rightarrow \tau \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Gamma, \tau \Rightarrow \nu X\sigma(X)}\,\nu'_r\bullet}{\Gamma, \sigma(\tau) \Rightarrow \sigma(\nu X\sigma(X))}\,\sigma \quad \Gamma, \tau \Rightarrow \sigma(\tau)}{\Gamma, \Gamma, \tau \Rightarrow \sigma(\nu X\sigma(X))}\,\mathsf{cut}}{\Gamma, \tau \Rightarrow \sigma(\nu X\sigma(X))}\,\mathsf{c}}{\Gamma, \tau \Rightarrow \nu X\sigma(X)}\,\nu'_r\bullet}{\Gamma, \Delta \Rightarrow \nu X\sigma(X)}\,\mathsf{cut}$$

Moreover, one can verify that this embedding gives rise to a bona fide simulation of $\rightsquigarrow^*_{\mathrm{cr}}$ by $\rightsquigarrow^*_{\mathrm{cr}'}$. We do not cover the details at this point, but make a stronger statement later in Proposition 3.9.

Example 3.5 (Functors and η-expansion of identity)**.** *Thanks to the decomposition of (co)iterators above, we can derive 'functors' in $\mathsf{C}\mu\mathsf{LJ}$, cf. Definition 2.8. This gives rise to an 'η-expansion' of identity steps, reducing them to atomic form. The critical fixed point cases are:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\mu X\sigma(X) \Rightarrow \mu X\sigma(X)}\,\mu_l\bullet}{\sigma(\mu X\sigma(X)) \Rightarrow \sigma(\mu X\sigma(X))}\,\sigma}{\sigma(\mu X(\sigma(X)) \Rightarrow \mu X\sigma(X)}\,\mu_r}{\mu X\sigma(X) \Rightarrow \mu X\sigma(X)}\,\mu'_l\bullet \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\nu X\sigma(X) \Rightarrow \nu X\sigma(X)}\,\nu_l\bullet}{\sigma(\nu X\sigma(X)) \Rightarrow \sigma(\nu X\sigma(X))}\,\sigma}{\sigma(\nu X(\sigma(X)) \Rightarrow \nu X\sigma(X)}\,\nu'_r}{\nu X\sigma(X) \Rightarrow \nu X\sigma(X)}\,\nu_l\bullet$$

Notice that the functors σ indicated above will depend on smaller identities, cf. Definition 2.8, calling the inductive hypothesis. Note that the coderivations above are 'logic-independent', and indeed this reduction is common in other circular systems for fixed point logics, such as the modal μ-calculus and $\mu\mathsf{MALL}$ (see, e.g., [BDS16]).

3.3. **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 [BDS16, KPP21, Das20a, Das21].

Definition 3.6 (Ancestry)**.** Fix a $\mu'\mathsf{LJ}$-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 conclusion (respectively) of an inference step and, as typeset in Figure 2, Figure 3 and Figure 10, 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 [BDS16], only for $\mu\mathsf{LJ}$ instead of $\mu\mathsf{MALL}$.

Definition 3.7 (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 5 which are relatively standard, e.g. [Koz83, Stu08, KMV22].

Definition 3.8 (Circular system)**.** $\mathsf{C}\mu\mathsf{LJ}$ is the class of regular progressing $\mu'\mathsf{LJ}$-coderivations.

Referencing Example 3.4, and for later use, we shall appeal to the notion of *simulation* for comparing models of computation in this work. Recalling that we construe $\mu\mathsf{LJ}$ as a model of computation under $\rightsquigarrow^*_{\mathrm{cr}}$ and $\mathsf{C}\mu\mathsf{LJ}$ as a model of computation under $\rightsquigarrow^*_{\mathrm{cr}'}$, we have:

Proposition 3.9 (Simulation)**.** $\mathsf{C}\mu\mathsf{LJ}$ *simulates* $\mu\mathsf{LJ}$.

Proof sketch. Replace each instance of a (co)iterator by the corresponding regular coderivation in Example 3.4. 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 $\mathsf{C}\mu\mathsf{LJ}$ under its rules. □

Example 3.10 (Revisiting natural number computation)**.** *Just like for $\mu\mathsf{LJ}$, we give native rules for N in $\mu'\mathsf{LJ}$, along with corresponding cut-reductions in Figure 12. We just show how to derive the conditional:*

$$\dfrac{\dfrac{\dfrac{\Gamma \Rightarrow \sigma}{\Gamma, 1 \Rightarrow \sigma}\,{}_{1_l} \quad \Gamma, N \Rightarrow \sigma}{\Gamma, 1 + N \Rightarrow \sigma}\,{}_{+_l}}{\Gamma, N \Rightarrow \sigma}\,{}_{\mu'_l}$$

As before, it is routine to show that these reductions are derivable using $\rightsquigarrow_{\mathrm{cr}'}$.

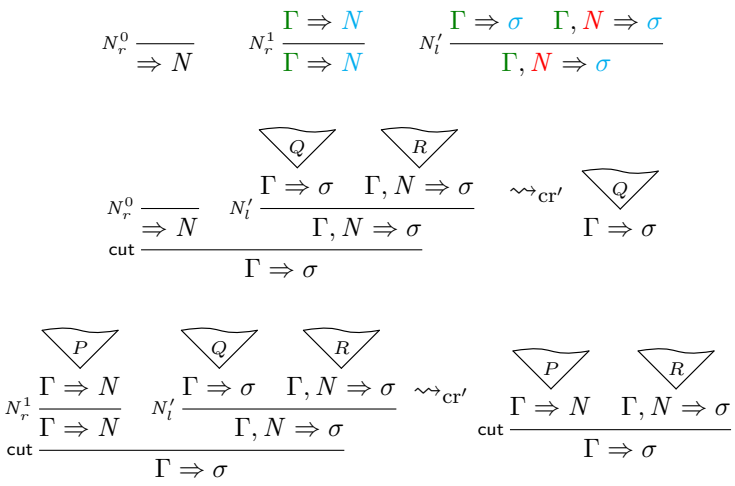

Figure 12: Native inference rules and cut-reduction steps for N in $\mu'\mathsf{LJ}$.

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

$$\dfrac{\Gamma \Rightarrow \sigma \qquad \dfrac{\vdots}{\Gamma, N \Rightarrow \sigma}\,N'_l \quad \Gamma, N, \sigma \Rightarrow \sigma}{\Gamma, N \Rightarrow \sigma}\,\mathsf{cut} \qquad \Gamma, N \Rightarrow \sigma}{\Gamma, N \Rightarrow \sigma}\,N'_l$$

Indeed it is immediate that $\mathsf{C}\mu\mathsf{LJ}$ contains circular versions of system T from [Das20a, Das21, KPP21].

3.4. **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., [AF98]). Indeed this translation can be designed to eliminate other 'positive' connectives too, in particular $+$.[3]

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 3.11 (Negative fragments)**.** We define $\mu\mathsf{LJ}^-$ as the subsystem of $\mu\mathsf{LJ}$ using only rules and cut-reductions over $N, \times, \to, \mu$. In particular we insist on the native rules

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

and cut-reductions for N from Figure 12 to avoid extraneous occurrences of $+$ from N and remain internal to the fragment. 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 3.12. *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 $\neg\sigma := \sigma \to N$:

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

The translation can be extended to coderivations by mapping every inference rule r to a gadget r^N preserving threads. Further details are given in Appendix B. □

Example 3.13. *The left coderivation of Figure 13 shows the encoding of a stream* $n_0 :: n_1 :: n_2 \ldots$ *by a (not necessarily regular) coderivation. Note that this coderivation is regular just if the stream is ultimately periodic. The right coderivation shows the circular presentation of the concatenation of a list and a stream into a stream discussed in Example 2.13. Note that, compared to the inductive encoding of this function, the circular one has an arguably more 'explicit' computational meaning. Both coderivations are progressing, by the red progressing threads in their only infinite branches.*

It is worth discussing how computation over streams is simulated in $\mathsf{C}\mu\mathsf{LJ}^-$ *via the double negation translation illustrated in Proposition 3.12. 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* $(\sigma \to N) \to N$*. 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* $N \to N$*.*

4. 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.

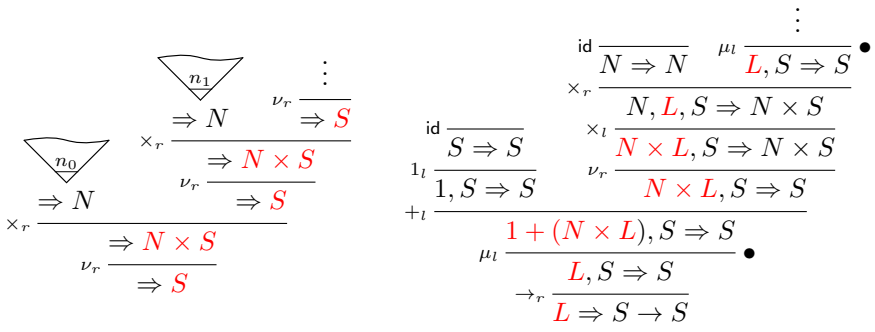

Figure 13: Left: pointwise computation of a stream in $\mu'\mathsf{LJ}$. Right: concatenation of a list and a stream in $\mathsf{C}\mu\mathsf{LJ}$.

4.1. **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 (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 $\mu'\mathsf{LJ}^-$, i.e. instances of any inference rules in these systems.

Definition 4.1 ((Co)terms [Das21])**.** A *coterm*, written s, t etc., is generated coinductively by the grammar:

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

I.e. coterms are possibly infinite expressions (of depth $\leq \omega$) generated by the grammar above. 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 4.2 ((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 $P_1, \ldots, P_n$ then P is $\mathsf{r}\, P_1 \ \ldots \ P_n$.

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 4.3 (Iterator coderivation as a regular coterm)**.** *Recalling the decomposition of the iterator as a circular coderivation in Example 3.4, let us specialise to the variation*

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

Figure 14: Reduction for LJ^- (both $\rightsquigarrow_{\mathrm{r}}$ and $\rightsquigarrow_{\mathrm{r}'}$).

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

Figure 15: Reduction for least fixed point rules in $\mu'\mathsf{LJ}$.

$\mathsf{iter}\,\dfrac{\sigma(\tau) \Rightarrow \tau}{\mu X \sigma(X) \Rightarrow \tau}$. *By following Example 3.4, we can express* $\mathsf{iter}\,P$ *by a regular coderivation, say* $\mathsf{iter}'(P)$*, that, viewed as a coterm, satisfies the (syntactic) equation,*

$$\mathsf{iter}'(P) \;=\; \mu'_l\,(\mathsf{cut}\,\sigma(\mathsf{iter}'(P))\,P) \tag{4.1}$$

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

4.2. **Computational models: theories of (co)terms.** Let us henceforth make the following abbreviations:

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

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

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

for $i \in \{0, 1\}$. We may omit types from subscripts when unimportant. When writing (co)terms using the derivations above, we employ the convention that p_i and s bind stronger than other applications, i.e. we write simply $t\,\mathsf{p}_i u$ for $t\,(\mathsf{p}_i u)$ and $t\,\mathsf{s} u$ for $t\,(\mathsf{s}\,u)$.

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 4.4 (Theories)**.** We define two (context-closed) reduction relations on (co)terms:

- $\rightsquigarrow_{\mathrm{r}}$ is generated by the clauses in Figures 14, 16 and 17.
- $\rightsquigarrow_{\mathrm{r}'}$ is generated by the rules in Figures 14, 15 and 18.

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

Figure 16: Reduction for least fixed points in $\mu\mathsf{LJ}$.

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

Figure 17: Reduction for N in $\mu\mathsf{LJ}^-$

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

Figure 18: Reduction for N in $\mu'\mathsf{LJ}^-$.

In all cases the lengths of vectors $\vec{x}$ and $\vec{y}$ match those of the relevant contexts Γ and Δ from the original typesetting of the rules, i.e. from Figures 2 to 4, 6 and 12. Note that the use of both variables and term metavariables in these reduction rules is purely to aid parsing. All reduction rules are closed under substitution.

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

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

Above x must be a fresh variable, not a general (co)term.

Admitting some extensionality is not necessary to reason about representability, since extensionality can be eliminated for low type levels, but simplifies some of the theorem statements.

Example 4.5 (Iteration equations)**.** *The fundamental equation for iteration is indeed derivable by* $\rightsquigarrow_{\mathrm{r}}$*:*

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

For $\rightsquigarrow_{\mathrm{r}'}$*, recalling Example 4.3 and using its notation, we can simulate the iteration equation for* $\mathsf{iter}\, P$ *with* $\mathsf{iter}'(P)$*:*

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

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 4.6 (Extensional reduction includes cut-reduction)**.** $\rightsquigarrow_{\mathrm{cr}'} \subseteq =^{\eta}_{\mathrm{r}'}$.

Proof. We show that, if $P \rightsquigarrow_{\mathrm{cr}'} P'$ then, for some $n \geq 0$ sufficiently large, and for any $x_1, \ldots, x_n$, $P\, x_1 \ldots x_n =_{\mathrm{r}'} P'\, x_1 \ldots x_n$. We then conclude by repeatedly applying the rule (η).

Suppose $P \rightsquigarrow_{\mathrm{cr}'} P'$. It suffices to consider the case where the last rule of P is the cut rule rewritten by the cut-reduction step. Indeed, the latter implies the general statement by appealing to the context closure of $\rightsquigarrow_{\mathrm{r}'}$. We only consider some relevant cases.

If P has the form,

$$\mathsf{cut}\ \dfrac{\times_r\ \dfrac{\overset{P_1}{\triangledown}\ \Gamma \Rightarrow \sigma \quad \overset{P_2}{\triangledown}\ \Delta \Rightarrow \tau}{\Gamma, \Delta \Rightarrow \sigma \times \tau} \qquad \times_l\ \dfrac{\overset{P_3}{\triangledown}\ \Sigma, \sigma, \tau \Rightarrow \gamma}{\Sigma, \sigma \times \tau \Rightarrow \gamma}}{\Gamma, \Delta, \Sigma \Rightarrow \gamma}$$

we have:

$$\begin{aligned}
\mathsf{cut}\,(\times_r P_1\, P_2)\,(\times_l P_3)\,\vec{x}\,\vec{y}\,\vec{z} &\rightsquigarrow_{\mathrm{r}'} \times_l P_3\,\vec{z}\,(\times_r P_1\, P_2\,\vec{x}\,\vec{y}) \\
&\rightsquigarrow_{\mathrm{r}'} \times_l P_3\,\vec{z}\,\langle P_1\,\vec{x}, P_2\,\vec{y}\rangle \\
&\rightsquigarrow^{*}_{\mathrm{r}'} P_3\,\vec{z}\,(P_1\,\vec{x})\,(P_2\,\vec{y})
\end{aligned}$$

$$\begin{aligned}
\mathsf{cut}\, P_2\,(\mathsf{cut}\, P_1\, P_3)\,\vec{x}\,\vec{y}\,\vec{z} &\rightsquigarrow_{\mathrm{r}'} (\mathsf{cut}\, P_1\, P_3)\,\vec{x}\,\vec{z}\,(P_2\,\vec{y}) \\
&\rightsquigarrow_{\mathrm{r}'} P_3\,\vec{z}\,(P_1\,\vec{x})\,(P_2\,\vec{y})
\end{aligned}$$

If P has the form,

$$\mathsf{cut}\ \dfrac{\rightarrow_r\ \dfrac{\overset{P_1}{\triangledown}\ \Gamma, \sigma \Rightarrow \tau}{\Gamma \Rightarrow \sigma \to \tau} \qquad \rightarrow_l\ \dfrac{\overset{P_2}{\triangledown}\ \Delta \Rightarrow \sigma \quad \overset{P_3}{\triangledown}\ \Sigma, \tau \Rightarrow \gamma}{\Delta, \sigma \to \tau, \Sigma \Rightarrow \gamma}}{\Gamma, \Delta, \Sigma \Rightarrow \gamma}$$

we have:

$$\begin{aligned}
\mathsf{cut}\,(\rightarrow_r P_1)\,(\rightarrow_l P_2\, P_3)\,\vec{x}\,\vec{y}\,\vec{z} &\rightsquigarrow_{\mathrm{r}'} \rightarrow_l P_2\, P_3\,\vec{y}\,(\rightarrow_r P_1\,\vec{x})\,\vec{z} \\
&\rightsquigarrow_{\mathrm{r}'} P_3\,\vec{z}\,(\rightarrow_r P_1\,\vec{x}\,(P_2\,\vec{y})) \\
&\rightsquigarrow_{\mathrm{r}'} P_3\,\vec{z}\,(P_1\,\vec{x}\,(P_2\,\vec{y}))
\end{aligned}$$

$$\begin{aligned}
\mathsf{cut}\,(\mathsf{cut}\, P_2\, P_1)\, P_3\,\vec{x}\,\vec{y}\,\vec{z} &\rightsquigarrow_{\mathrm{r}'} P_3\,\vec{z}\,(\mathsf{cut}\, P_2\, P_1\,\vec{x}\,\vec{y}) \\
&\rightsquigarrow_{\mathrm{r}'} P_3\,\vec{z}\,(P_1\,\vec{x}\,(P_2\,\vec{y}))
\end{aligned}$$

If P has the form,

$$\mathsf{cut}\ \dfrac{\mu_r\ \dfrac{\overset{P_1}{\triangledown}\ \Gamma \Rightarrow \sigma(\mu X \sigma(X))}{\Gamma \Rightarrow \mu X \sigma(X)} \qquad \mu'_l\ \dfrac{\overset{P_2}{\triangledown}\ \Delta, \sigma(\mu X \sigma(X)) \Rightarrow \gamma}{\Delta, \mu X \sigma(X) \Rightarrow \gamma}}{\Gamma, \Delta \Rightarrow \gamma}$$

$$
\begin{array}{rcl}
\mathsf{tt} & := & \lambda x, y.x \\
\mathsf{ff} & := & \lambda x, y.y \\
\mathsf{if}\ s\ \mathsf{then}\ t\ \mathsf{else}\ u & := & s\,t\,u \\
\langle\!\langle s, t \rangle\!\rangle & := & \lambda z.z\,s\,t \\
\pi_0 & := & \lambda z.z\,\mathsf{tt} \\
\pi_1 & := & \lambda z.z\,\mathsf{ff} \\
\lceil \underline{0} \rceil & := & \langle\!\langle \mathsf{ff}, \lambda x.x \rangle\!\rangle \\
\lceil \underline{n+1} \rceil & := & \langle\!\langle \mathsf{tt}, \lceil n \rceil \rangle\!\rangle \\
\mathsf{s} & := & \lambda x.\langle\!\langle \mathsf{tt}, x \rangle\!\rangle \\
\mathsf{p} & := & \lambda x.\pi_1\,x \\
\mathsf{cond} & := & \lambda x, y, z.\mathsf{if}\ (\pi_0\,z)\ \mathsf{then}\ x\ \mathsf{else}\ y\,(\mathsf{p}\,z)
\end{array}
\qquad
\begin{array}{rcl}
\mathsf{if}\ \underline{\mathsf{tt}}\ \mathsf{then}\ s\ \mathsf{else}\ t & =_{\beta,\eta} & s \\
\mathsf{if}\ \underline{\mathsf{ff}}\ \mathsf{then}\ s\ \mathsf{else}\ t & =_{\beta,\eta} & t \\
\pi_i\,\langle\!\langle s_0, s_1 \rangle\!\rangle & =_{\beta,\eta} & s_i \\
\mathsf{p}\,\lceil \underline{0} \rceil & =_{\beta,\eta} & \lceil 0 \rceil \\
\mathsf{p}\,(\mathsf{s}\,\lceil \underline{n} \rceil) & =_{\beta,\eta} & \lceil \underline{n} \rceil \\
\mathsf{cond}\,\lceil \underline{0} \rceil\,s\,t & =_{\beta,\eta} & s \\
\mathsf{cond}\,(\mathsf{s}\,\lceil \underline{n} \rceil)\,s\,t & =_{\beta,\eta} & t\,\lceil n \rceil
\end{array}
$$

Figure 19: Macros for λ-terms and corresponding reductions.

$$
\begin{array}{rcl}
[\![\mathsf{id}]\!] & := & \lambda t.t \\
[\![\mathsf{e}]\!] & := & \lambda t.\lambda \vec{u}.\lambda x.\lambda y.\lambda \vec{v}.t\,\vec{u}\,y\,x\,\vec{v} \\
[\![\mathsf{cut}]\!] & := & \lambda t.\lambda s.\lambda \vec{u}.\lambda \vec{v}.s\,\vec{v}\,(t\,\vec{u}) \\
[\![\mathsf{c}]\!] & := & \lambda t.\lambda \vec{u}.\lambda x.t\,\vec{u}\,x\,x \\
[\![\mathsf{w}]\!] & := & \lambda t.\lambda \vec{u}.\lambda x.t\,\vec{u} \\
[\![\rightarrow_r]\!] & := & \lambda t.\lambda \vec{u}.\lambda x.t\,\vec{u}\,x \\
[\![\rightarrow_l]\!] & := & \lambda t.\lambda s.\lambda \vec{u}.\lambda \vec{v}.\lambda x.s\,\vec{v}\,(x\,(t\,\vec{u}))
\end{array}
\qquad
\begin{array}{rcl}
[\![\times_r]\!] & := & \lambda t.\lambda s.\lambda \vec{u}.\lambda \vec{v}.\langle\!\langle t\,\vec{u}, s\,\vec{v} \rangle\!\rangle \\
[\![\times_l]\!] & := & \lambda t.\lambda \vec{u}.\lambda x.t\,\vec{u}\,(\pi_0\,x)\,(\pi_1\,x) \\
[\![\mu_r]\!] & := & \lambda t.\lambda \vec{u}.t\,\vec{u} \\
[\![\mu_l]\!] & := & \lambda t.\lambda \vec{u}.\lambda z.t\,\vec{u}\,z \\
[\![N_r^0]\!] & := & \lceil 0 \rceil \\
[\![N_r^1]\!] & := & \lambda t.\lambda \vec{x}.\mathsf{s}\,(t\,\vec{x}) \\
[\![N_l']\!] & := & \lambda t.\lambda s.\lambda \vec{u}.\lambda x.\mathsf{cond}\,x\,(t\,\vec{u})\,(s\,\vec{u})
\end{array}
$$

Figure 20: Translation of $\mathsf{C}\mu\mathsf{LJ}^-$ into λ-terms.

we have:

$$
\begin{array}{rcl}
\mathsf{cut}\,(\mu_r\,P_1)\,\mu_l'\,P_2\,\vec{x}\,\vec{y} & \rightsquigarrow_{\mathrm{r}'} & (\mu_l'\,P_2)\,\vec{y}\,(\mu_r\,P_1\,\vec{x}) \\
& \rightsquigarrow_{\mathrm{r}'}^{*} & P_2\,\vec{y}\,(P_1\,\vec{x}) \\
\mathsf{cut}\,P_1\,P_2\,\vec{x}\,\vec{y} & \rightsquigarrow_{\mathrm{r}'} & P_2\,\vec{y}\,(P_1\,\vec{x})
\end{array}
$$

□

4.3. **An embedding into λ-terms.** While the significant technical development of this work involves 'totality' arguments, e.g. in Section 5 showing that the representable partial functions of $\mathsf{C}\mu\mathsf{LJ}^-$ (under $=_{\mathrm{r}'}^{\eta}$) are total, we better address *determinism* too.

As it stands, $=_{\mathrm{r}'}$ and $=_{\mathrm{r}'}^{\eta}$ may fire distinct reductions on the same (co)terms, 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 (untyped) λ-calculus. We shall prove that the (untyped) λ-calculus, under $\beta\eta$-reduction, simulates $\langle \mathsf{C}\mu\mathsf{LJ}^- \rangle$ under $\rightsquigarrow_{\mathrm{r}'}^{\eta}$. 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. As the techniques are rather standard, we shall be quite succinct in the exposition.

λ-*terms*, written s, t etc., are generated as usual by:

$$s, t \quad ::= \quad x \quad | \quad (t\,s) \quad | \quad \lambda x\,t$$

We write Λ for the set of all terms. The notion of 'free variable' is defined as expected, and we write $\mathrm{FV}(t)$ for the set of free variables of the term t.

We work with a standard equational theory on λ-terms. $=_{\beta,\eta}$ is the smallest congruence on Λ satisfying:

$$\lambda x t\, s =_\beta t[s/x] \qquad\qquad \lambda x(tx) =_\eta t \quad (x \notin \mathrm{FV}(t))$$

Figure 19 displays some macros for λ-terms (and the corresponding reduction rules) we will adopt in this subsection. We define an interpretation of coterms in $\langle \mathsf{C}\mu\mathsf{LJ}^- \rangle$ into Λ. We start with an interpretation of the basic inference rules:

Definition 4.7 (Interpreting rules)**.** To each inference step r of $\mathsf{C}\mu\mathsf{LJ}^-$ we associate a λ-term $[\![\mathsf{r}]\!]$ as shown in Figure 20.

It is easy to see that this interpretation preserves equations from Figure 14, 15, and 18:

Lemma 4.8. *The following equations hold in Λ:*

$$\begin{array}{rcl}
[\![\mathsf{id}]\!]\ x & =_{\beta,\eta} & x \\
[\![\mathsf{e}]\!]\ t\ \vec{x}\ x\ y\ \vec{y} & =_{\beta,\eta} & t\ \vec{x}\ y\ x\ \vec{y} \\
[\![\mathsf{w}]\!]\ t\ \vec{x}\ x & =_{\beta,\eta} & t\ \vec{x} \\
[\![\mathsf{c}]\!]\ t\ \vec{x}\ x & =_{\beta,\eta} & t\ \vec{x}\ x\ x \\
[\![\mathsf{cut}]\!]\ s\ t\ \vec{x}\ \vec{y} & =_{\beta,\eta} & t\ \vec{y}\ (s\ \vec{x}) \\
[\![\times_r]\!]\ s\ t\ \vec{x}\ \vec{y} & =_{\beta,\eta} & \langle\!\langle s\,\vec{x}, t\,\vec{y} \rangle\!\rangle \\
[\![\times_l]\!]\ t\,\vec{x}\,y & =_{\beta,\eta} & t\ \vec{x}\ ([\![\mathsf{p}_0]\!]y)\ ([\![\mathsf{p}_1]\!]y)
\end{array}
\qquad
\begin{array}{rcl}
[\![\mathsf{p}_i]\!]\langle\!\langle x_0, x_1 \rangle\!\rangle & =_{\beta,\eta} & x_i \\
[\![\to_r]\!]\ t\ \vec{x}\ x & =_{\beta,\eta} & t\ \vec{x}\ x \\
[\![\to_l]\!]\ s\ t\ \vec{x}\ \vec{y}\ z & =_{\beta,\eta} & t\ \vec{y}\ (z\,(s\ \vec{x})) \\
[\![\mu_r]\!]\ t\ \vec{x} & =_{\beta,\eta} & t\ \vec{x} \\
[\![\mu_l]\!]\ t\ \vec{x} & =_{\beta,\eta} & t\ \vec{x} \\
[\![N_l']\!]\, s\, t\, \vec{x}\, \lceil 0 \rceil & =_{\beta,\eta} & s\,\vec{x} \\
[\![N_l']\!]\, s\, t\, \vec{x}\, \mathsf{s} y & =_{\beta,\eta} & t\,\vec{x}\,y
\end{array}$$

Also, if $[\![t]\!]\,x =_{\beta,\eta} [\![t']\!]\,x$, for $x \notin \mathrm{FV}(t)$, then $[\![t]\!] =_{\beta,\eta} [\![t']\!]$ by η-reduction in Λ.

We now show how to extend $[\![\_]\!]$ to regular coderivations, by noting that any such coterm can be described by a finite system of simultaneous equations. From here we rely on a well-known result that such finite systems of equations always admit solutions in the untyped lambda calculus, with respect to $=_{\beta,\eta}$ (see, e.g., [HS08]).

Definition 4.9 (Interpreting regular coderivations)**.** Consider a $\mathsf{C}\mu\mathsf{LJ}^-$ coderivation P with subcoderivations $\vec{P} = P_1, \ldots, P_n$ where $P = P_1$. Suppose each P_i is concluded by an inference step r_i with immediate subcoderivations $\vec{P}_i$ (a list of regular coderivations among $\vec{P}$). Write $\mathcal{E}_P$ for the system of equations $\{x_i = [\![\mathsf{r}_i]\!]\vec{x}_i\}_{i=1}^n$, where we set $\vec{x}_i := x_{i_1}, \ldots, x_{i_k}$ when $\vec{P}_i = P_{i_1}, \ldots, P_{i_k}$. We define $[\![P]\!]$ to be some/any solution to x_1 of $\mathcal{E}_P$ in Λ, with respect to $=_{\beta,\eta}$.

From here we extend the definition of $[\![\cdot]\!]$ to all coterms in $\langle \mathsf{C}\mu\mathsf{LJ}^- \rangle$ inductively as expected, setting $[\![ts]\!] := [\![t]\!][\![s]\!]$.

Now, immediately from the definition of $[\![\cdot]\!]$ and Lemma 4.8 we arrive at our intended interpretation:

Proposition 4.10. *If $t \in \langle \mathsf{C}\mu\mathsf{LJ}^- \rangle$ and $t \rightsquigarrow^\eta_{\mathrm{r}'} s$ then $[\![t]\!] =_{\beta,\eta} [\![s]\!]$.*

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

Corollary 4.11 (Uniqueness)**.** *Let $t \in \langle \mathsf{C}\mu\mathsf{LJ}^- \rangle$. If $\underline{m} =^\eta_{\mathrm{r}'} t =^\eta_{\mathrm{r}'} \underline{n}$ then $n = m$.*

Proof. Clearly, by Proposition 4.10 we have $[\![t]\!] =_{\beta,\eta} [\![\underline{n}]\!]$ and $[\![t]\!] =_{\beta,\eta} [\![\underline{m}]\!]$. Since $[\![\underline{n}]\!] =_{\beta,\eta} \lceil \underline{n} \rceil$, we have $[\![t]\!] =_{\beta,\eta} \lceil \underline{n} \rceil$ and $[\![t]\!] =_{\beta,\eta} \lceil \underline{m} \rceil$. Since $\lceil \underline{n} \rceil$ and $\lceil \underline{m} \rceil$ are normal forms, by confluence of Λ it must be that $\lceil \underline{n} \rceil = \lceil \underline{m} \rceil$, and hence $n = m$. □

4.4. **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 7, it will be convenient to work with a typed version of $\langle \mu\mathsf{LJ}^- \rangle$:

Definition 4.12 (Type assignment)**.** *Type assignment* is the smallest (infix) relation ':' from terms to types satisfying:

- 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 $\mathsf{r} : (\vec{\sigma}_1 \to \tau_1) \to \cdots \to (\vec{\sigma}_n \to \tau_n) \to \vec{\sigma} \to \tau$.
- if $t : \sigma \to \tau$ and $s : \sigma$ then $ts : \tau$.
- if $t : \mu X \sigma(X)$ and $s : \sigma(\tau) \to \tau$ then $ts : \tau$.

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

The main result of this subsection is:

Theorem 4.13 (Terms to derivations)**.** *The natural number functions represented by* $\langle \mu\mathsf{LJ}^- \rangle_{N,\times,\to,\mu}$ *are already representable by* $\mu\mathsf{LJ}^-$*.*

Proof sketch. First, given a derivation P of $\Rightarrow \sigma \to \tau$ in $\mu\mathsf{LJ}^-$, we define the derivation $\mathsf{uncurry}(P)$ of $\sigma \Rightarrow \tau$ as follows:

$$\mathsf{cut}\,\dfrac{\begin{array}{c}P\\ \Rightarrow \sigma \to \tau\end{array} \qquad \to_l\,\dfrac{\mathsf{id}\,\dfrac{}{\sigma \Rightarrow \sigma} \qquad \mathsf{id}\,\dfrac{}{\tau \Rightarrow \tau}}{\sigma, \sigma \to \tau \Rightarrow \tau}}{\sigma \Rightarrow \tau}$$

We now define an interpretation of type assignments $t : \tau$ into derivations $P_{\mathcal{D}}$ of $\Rightarrow \tau$ by induction on $t : \tau$ as follows:

- For each step $\mathsf{r} : (\vec{\sigma}_1 \to \tau_1) \to \cdots \to (\vec{\sigma}_n \to \tau_n) \to \vec{\sigma} \to \tau$, P_{r} is the derivation of r in $\mu\mathsf{LJ}^-$:

$$\to_r\,\dfrac{\mathsf{r}\,\dfrac{\dfrac{\dfrac{}{\vec{\sigma}_1 \to \tau_1, \vec{\sigma}_1 \Rightarrow \tau_1} \quad \cdots \quad \dfrac{\dfrac{}{\vec{\sigma}_n \to \tau_n, \vec{\sigma}_n \Rightarrow \tau_n}}{}}{}}{\vec{\sigma}_1 \to \tau_1, \dots, \vec{\sigma}_n \to \tau_n, \vec{\sigma} \Rightarrow \tau}}{\Rightarrow (\vec{\sigma}_1 \to \tau_1) \to \cdots \to (\vec{\sigma}_n \to \tau_n) \to \vec{\sigma} \to \tau}$$

- If $t : \sigma \to \tau$ and $s : \sigma$ then P_{ts} is defined as:

$$\mathsf{cut}\,\dfrac{\begin{array}{c}P_s\\ \Rightarrow \sigma\end{array} \qquad \begin{array}{c}\mathsf{uncurry}(P_t)\\ \vee\\ \sigma \Rightarrow \tau\end{array}}{\Rightarrow \tau}$$

• If $t : \mu X\sigma(X)$ and $s : \sigma(\tau) \to \tau$ then P_{ts} is defined as:

$$\dfrac{P_t \quad \dfrac{\overbrace{\quad}^{\mathsf{uncurry}(s)} \quad \dfrac{}{\tau \Rightarrow \tau}\,\mathsf{id}}{\mu X\sigma(X) \Rightarrow \tau}\,\mu_l}{\Rightarrow \tau}\,\mathsf{cut}$$

Now, to show that typed-$\langle \mu\mathsf{LJ}^- \rangle$ and $\mu\mathsf{LJ}^-$ represent the same functions on natural numbers, it suffices to prove that if $t : \tau$ and $t \rightsquigarrow_{\mathrm{r}} t'$ then $P_t \rightsquigarrow^*_{\mathrm{cr}} P_{t'}$. For this let us observe:

• Whenever t is typed, so are all its subterms by definition of type assignment.
• Any derivation $P_{t(s)}$ in the form $P_t(P_s)$, for appropriate $P_t(\cdot)$ (with leaf $\cdot$).

Thus the simulation of any reduction step $t(s) \rightsquigarrow_{\mathrm{r}} t(s')$, with redex s, is reduced to showing $s \rightsquigarrow_{\mathrm{r}} s' \implies P_s \rightsquigarrow^*_{\mathrm{cr}} P_{s'}$. This boils down to checking that the reductions in Figure 14, Figure 16 and Figure 17 are simulated by a series of cut-reduction rules on $\mu\mathsf{LJ}^-$, which is routine. □

5. Totality of circular proofs

In this section we provide a semantics for (circular) proofs, using computability theoretic tools. Our aim is to show that $\mathsf{C}\mu\mathsf{LJ}$ represents only total functions on $\mathbb{N}$ (Corollary 5.5), 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.

5.1. 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 5.1 (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.

Remark 5.2 (Alternative SO interpretation)**.** Recalling the second-order interpretation of μ-types, Remark 2.5, an alternative definition of $|\mu X\sigma(X)|$ could be $\bigcap_A(\sigma(A) \to A) \to A$. This gives rise to a different type structure, indeed similar to the realisability model we give later in Section 7. However such a choice does not allow us to readily interpret $\langle \mathsf{C}\mu\mathsf{LJ}^- \rangle$: the totality argument in this section, Theorem 5.4, crucially exploits the fact that μ-types are interpreted as bona-fide least fixed points.

A routine but important property is:

Proposition 5.3 (Closure under conversion)**.** *If $t \in |\tau|$ and $t =^{\eta}_{\mathrm{r}'} t'$ then $t' \in |\tau|$.*

Proof. By induction on the structure of τ. The base cases when τ is a candidate A or the type N follow immediately from the definitions. For the remaining cases:

- If $\tau = \tau_0 \times \tau_1$, then $\mathsf{p}_i t' =^{\eta}_{\mathrm{r}'} \mathsf{p}_i t \in |\tau_i|$ by IH, for $i = 0, 1$, so indeed $t' \in |\tau|$.
- If $\tau = \tau_0 \to \tau_1$ and $s \in \tau_0$, then $t's =^{\eta}_{\mathrm{r}'} ts \in |\tau_1|$ by IH, so indeed $t' \in |\tau|$.
- If $\tau = \mu X\tau'(X)$ and A is a candidate with $|\sigma(A)| \subseteq A$, then $t' \in A$ by IH, so indeed $t' \in |\tau|$. □

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 $t' \notin |\tau|$.

The main result of this section is:

Theorem 5.4 (Interpretation)**.** *For any $\mathsf{C}\mu\mathsf{LJ}^-$-coderivation $P : \Sigma \Rightarrow \tau$ and $\vec{s} \in |\Sigma|$ we have $P\vec{s} \in |\tau|$.*

The rest of this section is devoted to proving this result, but before that let us state our desired consequence:

Corollary 5.5. *$\mathsf{C}\mu\mathsf{LJ}^-$ 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 Theorem 5.4 and closure under conversion, Proposition 5.3 (namely applying cut-reductions), we have for all $\vec{m} \in \mathbb{N}$ there is $n \in \mathbb{N}$ with $P\underline{\vec{m}} =^{\eta}_{\mathrm{r}'} \underline{n}$. □

5.2. **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.

Since the class of candidates forms a complete lattice under set inclusion, we can specialise the well-known Knaster-Tarski fixed point theorem:

Proposition 5.6 (Knaster-Tarski for candidates)**.** *Let F be a monotone operation on candidates, with respect to $\subseteq$. F has a least fixed point $\mu F = \bigcap\{A \text{ a candidate} \mid F(A) \subseteq A\}$.*

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 5.7 (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 5.8 ("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 6, crucial for obtaining a tight bound on representable functions,

Definition 5.9 (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:

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

For our purposes we will only need the special case of the definition above when $A = \varnothing$. Writing Ord for the class of all ordinals, the following is well known:

Proposition 5.10 (Fixed points via approximants)**.** *Let μF be the least fixed point of a monotone operation F. We have $\mu F = \bigcup_{\alpha\in\mathrm{Ord}} F^\alpha(\varnothing)$.*

From here it is convenient to admit formal type expressions representing approximants.

Convention 5.11 (Transfinite types)**.** We henceforth expand the language of types to be closed under:

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

Again we shall only need the special case of $\tau = \varnothing$ for our purposes. We call such types *transfinite* when we need to distinguish them from ordinal-free types.

Definition 5.12 (Type structure, continued)**.** We expand Definition 5.1 to account for transfinite types by setting $|\sigma^\alpha(\tau)| := |\sigma(\cdot)|^\alpha(|\tau|)$. In other words:

- $|\sigma^0(\tau)| := |\tau|$
- $|\sigma^{\mathsf{s}\alpha}(\tau)| := |\sigma(|\sigma^\alpha(\tau)|)|$
- $|\sigma^\lambda(\tau)| := \bigcup_{\alpha<\lambda} |\sigma^\alpha(\tau)|$, when λ is a limit ordinal.

We have immediately from Proposition 5.10:

Corollary 5.13. $|\mu X\sigma(X)| = \bigcup_{\alpha\in\mathrm{Ord}} |\sigma^\alpha(\varnothing)|$.

5.3. **Ordinal assignments.** We shall write $\sigma \subseteq \tau$ if σ is a subformula of τ.

The *Fischer-Ladner (well) preorder*, written $\preceq_{\mathrm{FL}}$, is the smallest extension of $\subseteq$, restricted to closed formulas, satisfying $\sigma(\mu X\sigma(X)) \preceq_{\mathrm{FL}} \mu X\sigma(X)$. We write $\sigma \approx_{\mathrm{FL}} \tau$ if $\sigma \preceq_{\mathrm{FL}} \tau \preceq_{\mathrm{FL}} \sigma$ and $\sigma \prec_{\mathrm{FL}} \tau$ if $\sigma \preceq_{\mathrm{FL}} \tau \not\preceq_{\mathrm{FL}} \sigma$. Note that $\approx_{\mathrm{FL}}$-equivalence classes are naturally (well) partially ordered by $\preceq_{\mathrm{FL}}$.

The *Fischer-Ladner closure* of a type τ, written $\mathrm{FL}(\tau)$, is the set $\{\sigma \mid \sigma \preceq_{\mathrm{FL}} \tau\}$. Note that $\mathrm{FL}(\tau)$ is the smallest set of closed types closed under subformulas and, whenever $\mu X\sigma(X) \in \mathrm{FL}(\tau)$, then also $\sigma(\mu X\sigma(X)) \in \mathrm{FL}(\tau)$; this is necessarily a finite set.

Definition 5.14 (Priority)**.** We say that a type σ has higher *priority* than a type τ, written $\sigma > \tau$, if $\tau \prec_{\mathrm{FL}} \sigma$ or $\sigma \approx_{\mathrm{FL}} \tau$ and $\sigma \subseteq \tau$.

Note that $<$ is a (strict) well partial order on types. The priority order is commonly used in modal fixed point logics, e.g., [Stu08, Dou17, Ven08].

Convention 5.15. In what follows, we shall assume an arbitrary extension of $<$ to a *total* well order.

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 $j < i$. 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 on ordinal sequences.

By the Monotonicity Lemma 5.7 we have:

Proposition 5.16 (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. $t \notin |\tau_{\vec{\alpha}}|$.*

Since the ordinals given by the above Proposition are points of first entry for an element into a fixed point, note that, for $\vec{\alpha}$ as in the Proposition above, each α_i must be a successor ordinal.

5.4. **Reflecting non-totality in rules of $\mu'\mathsf{LJ}^-$.** Before giving our non-total branch construction, let us first make a local definition that will facilitate our construction:

Definition 5.17 (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 define a premiss $\Sigma' \Rightarrow \tau'$ and $P' : \Sigma' \Rightarrow \tau'$ and some inputs $\vec{s}' \in |\Sigma'|$ such that $P'\vec{s}' \notin |\tau'|$ as follows:

- r cannot be id since $\mathsf{id}s =^{\eta}_{\mathsf{r}'} s$.

- If r is $\mathsf{e}\dfrac{\Gamma,\sigma,\rho,\Delta\Rightarrow\tau}{\Gamma,\rho,\sigma,\Delta\Rightarrow\tau}$ and $\vec{s}=(\vec{r},r,t,\vec{t})$ with $\vec{r}\in|\Gamma|, r\in|\rho|, s\in|\sigma|, \vec{t}\in|\Delta|$, we set $(\Sigma'\Rightarrow\tau'):=(\Gamma,\sigma,\rho,\Delta\Rightarrow\tau)$, $P':=P_0$ and $\vec{s}':=(\vec{r},t,r,\vec{t})$.
- If r is $\mathsf{w}\dfrac{\Sigma_0\Rightarrow\tau}{\Sigma_0,\sigma\Rightarrow\tau}$ and $\vec{s}=(\vec{s}_0,s)$ with $\vec{s}_0\in|\Sigma_0|$ and $s\in|\sigma|$ then we set $P':=P_0$, $(\Sigma'\Rightarrow\tau'):=(\Sigma_0\Rightarrow\tau)$ and $\vec{s}':=\vec{s}_0$.
- If r is $\mathsf{c}\dfrac{\Sigma_0,\sigma,\sigma\Rightarrow\tau}{\Sigma_0,\sigma\Rightarrow\tau}$ and $\vec{s}=(\vec{s}_0,s)$ with $\vec{s}_0\in|\Sigma_0|, s\in|\sigma|$, then we set $(\Sigma'\Rightarrow\tau'):=(\Sigma_0\Rightarrow\tau)$, $P':=P_0$ and $\vec{s}':=(\vec{s}_0,s,s)$.
- If r is $\mathsf{cut}\dfrac{\Sigma_0\Rightarrow\tau_0\quad\Sigma_1,\tau_0\Rightarrow\tau}{\Sigma_0,\Sigma_1\Rightarrow\tau}$ and $\vec{s}_0\in|\Sigma_0|, \vec{s}_1\in|\Sigma_1|$ we have:

$$\begin{array}{lll} & \mathsf{cut}P_0P_1\vec{s}_0\vec{s}_1\ \notin|\tau| & \\ \Longrightarrow & P_1\vec{s}_1(P_0\vec{s}_0)\ \notin|\tau| & \text{by } \mathsf{cut} \text{ reduction}\end{array}$$

  Now, if $P_0\,\vec{s}_0\ \notin|\tau_0|$ then we set $(\Sigma'\Rightarrow\tau'):=(\Sigma_0\Rightarrow\tau_0)$, $P':=P_0$ and $\vec{s}':=\vec{s}_0$. Otherwise $P_0\,\vec{s}_0\ \in|\tau_0|$ so we set $(\Sigma'\Rightarrow\tau'):=(\Sigma_1\Rightarrow\tau_1)$, $P':=P_1$ and $\vec{s}':=(\vec{s}_1,P_0\,\vec{s}_0)$.

- r cannot be $N_r^0\dfrac{}{\Rightarrow N}$ as $\underline{0}\in|N|$.
- If r is $N_r^1\dfrac{\Sigma\Rightarrow N}{\Sigma\Rightarrow N}$ and $\vec{s}\in|\Sigma|$,

$$\begin{array}{lll} & N_r^1P_0\vec{s}\notin|N| & \\ \Longrightarrow & \mathsf{s}(P_0\vec{s})\notin|N| & \text{by } N_r^1 \text{ reduction}\\ \Longrightarrow & P_0\vec{s}\notin|N| & \text{by context closure of } =^{\eta}_{\mathrm{r}'}\end{array}$$

  so we set $(\Sigma'\Rightarrow\tau'):=(\Sigma\Rightarrow N)$, $P':=P_0$ and $\vec{s}':=\vec{s}$.
- If r is $N_l'\dfrac{\Sigma_0\Rightarrow\tau\quad\Sigma_0,N\Rightarrow\tau}{\Sigma_0,N\Rightarrow\tau}$ and $\vec{s}_0\in|\Sigma_0|$, $s\in|N|$, then $s=^{\eta}_{\mathrm{r}'}\underline{n}$ for some $n\in\mathbb{N}$ by definition of $|N|$. If $n=0$ then,

$$\begin{array}{lll} & N_l'P_0P_1\vec{s}s\notin|\tau| & \\ \Longrightarrow & N_l'P_0P_1\vec{s}\underline{0}\notin|\tau| & \text{by closure under conversion}\\ \Longrightarrow & P_0\vec{s}\notin|\tau| & \text{by } N_l' \text{ reduction}\end{array}$$

  so we set $(\Sigma'\Rightarrow\tau'):=(\Sigma_0\Rightarrow\tau)$, $P':=P_0$, $\vec{s}':=\vec{s}$.
  Otherwise if $n=n'+1$ then,

$$\begin{array}{lll} & N_l'P_0P_1\vec{s}s\notin|\tau| & \\ \Longrightarrow & N_l'P_0P_1\vec{s}(\mathsf{s}\underline{n}')\notin|\tau| & \text{by closure under conversion}\\ \Longrightarrow & P_1\vec{s}\underline{n}'\notin|\tau| & \text{by } N_l' \text{ reduction}\end{array}$$

  so we set $(\Sigma'\Rightarrow\tau'):=(\Sigma_0,N\Rightarrow\tau)$, $P':=P_1$, $\vec{s}':=(\vec{s},\underline{n}')$.
- If r is $\times r\dfrac{\Sigma_0\Rightarrow\tau_0\quad\Sigma_1\Rightarrow\tau_1}{\Sigma_0,\Sigma_1\Rightarrow\tau_0\times\tau_1}$ and $\vec{s}_0\in|\Sigma_0|, \vec{s}_1\in|\Sigma_1|$,

$$\begin{array}{lll} & \times_rP_0P_1\vec{s}_0\vec{s}_1\notin|\tau_0\times\tau_1| & \\ \Longrightarrow & \langle P_0\vec{s}_0,P_1\vec{s}_1\rangle\notin|\tau_0\times\tau_1| & \text{by } \times_r \text{ reduction}\\ \Longrightarrow & \mathsf{p}_i\langle P_0\vec{s}_0,P_1\vec{s}_1\rangle\notin|\tau_i| & \text{by } |\times| \\ \Longrightarrow & P_i\vec{s}_i\notin|\tau_i| & \text{by } \mathsf{p}_i\langle\cdot,\cdot\rangle \text{ reduction}\end{array}$$

for some $i \in \{0,1\}$, so we set $(\Sigma' \Rightarrow \tau') := (\Sigma_i \Rightarrow \tau_i)$, $P' := P_i$, $\vec{s}' := \vec{s}_i$.

- If r is $\times_l \dfrac{\Gamma, \sigma_0, \sigma_1 \Rightarrow \tau}{\Gamma, \sigma_0 \times \sigma_1 \Rightarrow \tau}$ and $\vec{r} \in |\Gamma|$ and $s \in |\sigma_0 \times \sigma_1|$:

$$\begin{array}{rl} & \times_l P_0 \vec{r} s \ \notin \ |\tau| \\ \Longrightarrow & P_0 \vec{r} \, \mathsf{p}_0 s \, \mathsf{p}_1 s \ \notin \ |\tau| \quad \text{by } \times_l \text{ reduction} \end{array}$$

Now, by definition of $|\sigma_0 \times \sigma_1|$ we have indeed $\mathsf{p}_i s \in |\sigma_i|$ for $i \in \{0,1\}$, so we set $(\Sigma' \Rightarrow \tau') = (\Gamma, \sigma_0, \sigma_1 \Rightarrow \tau)$, $P' = P_0$ and $\vec{s}' = (\vec{r}, \mathsf{p}_i s)$.

- If r is $\to_r \dfrac{\Sigma, \rho \Rightarrow \sigma}{\Sigma \Rightarrow \rho \to \sigma}$ and $\vec{s} \in |\Sigma|$,

$$\begin{array}{rll} & \to_r P_0 \vec{s} \ \notin \ |\rho \to \sigma| & \\ \Longrightarrow & \to_r P_0 \vec{s} \ \notin \ |(\rho \to \sigma)_{\vec{\alpha}}| & \text{for some least } \vec{\alpha} \\ \Longrightarrow & \to_r P_0 \vec{s} \ \notin \ |\rho^{\vec{\alpha}_0} \to \sigma_{\vec{\alpha}_1}| & \text{by definition} \\ \Longrightarrow & \to_r P_0 \vec{s} s \ \notin \ |\sigma_{\vec{\alpha}_1}| & \text{for some } s \in |\rho^{\vec{\alpha}_0}| \\ \Longrightarrow & P_0 \vec{s} s \ \notin \ |\sigma_{\vec{\alpha}_1}| & \text{by } \to_r \text{ reduction} \\ \Longrightarrow & P_0 \vec{s} s \notin |\sigma| & \text{by monotonicity} \end{array}$$

where $\vec{\alpha}_0$ and $\vec{\alpha}_1$ are appropriate subsequences of $\vec{\alpha}$ in case not all fixed point subformulas of $\rho \to \sigma$ occur in ρ or in σ. So we set $(\Sigma' \Rightarrow \tau') := (\Sigma, \rho \Rightarrow \sigma)$, and $P' := P_0$ and $\vec{s}' := (\vec{s}, s)$.

- If r is $\to_l \dfrac{\Gamma \Rightarrow \rho \quad \Delta, \sigma \Rightarrow \tau}{\Gamma, \Delta, \rho \to \sigma \Rightarrow \tau}$ let $\vec{r} \in |\Gamma|, \vec{t} \in |\Delta|$ and $s \in |\rho \to \sigma|$. Like before, let $\vec{\alpha}$ be the least assignment such that $s \in |(\rho \to \sigma)^{\vec{\alpha}}| = |\rho_{\vec{\alpha}_0} \to \sigma^{\vec{\alpha}_1}|$. We have two cases:
  - if $P_0 \vec{r} \notin |\rho_{\vec{\alpha}_0}|$ then we set $(\Sigma' \Rightarrow \tau') := (\Gamma \Rightarrow \rho)$, and $P' := P_0$ and $\vec{s}' := \vec{r}$.
  - otherwise $P_0 \vec{r} \in |\rho_{\vec{\alpha}_0}|$ and we have:

$$\begin{array}{rl} & \to_l P_0 P_1 \vec{r} \vec{t} s \notin |\tau| \\ \Longrightarrow & P_1 \vec{t} (s(P_0 \vec{r})) \notin |\tau| \quad \text{by } \to_l \text{ reduction} \end{array}$$

  Since $P_0 \vec{r} \in |\rho_{\vec{\alpha}_0}|$ and $s \in |\rho_{\vec{\alpha}_0} \to \sigma^{\vec{\alpha}_1}|$ we have $s(P_0 \vec{r}) \in |\sigma^{\vec{\alpha}_1}|$ so we set $(\Sigma' \Rightarrow \tau') := (\Delta, \sigma \Rightarrow \tau)$, and $P' := P_1$ and $\vec{s}' := (\vec{t}, s(P_0 \vec{r}))$.

- If r is $\mu_r \dfrac{\Sigma \Rightarrow \sigma(\mu X \sigma(X))}{\Sigma \Rightarrow \mu X \sigma(X)}$ and $\vec{s} \in |\Sigma|$,

$$\begin{array}{rll} & \mu_r P_0 \vec{s} \ \notin \ |\mu X \sigma(X)| & \\ \Longrightarrow & P_0 \vec{s} \notin |\mu X \sigma(X)| & \text{by } \mu_r \text{ reduction} \\ \Longrightarrow & P_0 \vec{s} \ \notin \ |\sigma(\mu X \sigma(X))| & \text{by Proposition 5.8} \end{array}$$

so we set $(\Sigma' \Rightarrow \tau') := (\Sigma \Rightarrow \sigma(\mu X \sigma(X)))$, $P' := P_0$ and $\vec{s}' := \vec{s}$.

- if r is $\mu_l' \dfrac{\Gamma, \sigma(\mu X \sigma(X)) \Rightarrow \tau}{\Gamma, \mu X \sigma(X) \Rightarrow \tau}$ and $\vec{r} \in |\Gamma|$ and $s \in |\mu X \sigma(X)|$ we have:

$$\begin{array}{rl} & \mu_l P_0 \vec{r} s \ \notin \ |\tau| \\ \Longrightarrow & P_0 \vec{r} s \ \notin \ |\tau| \quad \text{by } \mu_l \text{ reduction} \end{array}$$

Since $s \in |\mu X \sigma(X)|$ then also $s \in |\sigma(\mu X \sigma(X))|$ by Proposition 5.8, so we set $(\Sigma' \Rightarrow \tau') := (\Gamma, \sigma(\mu X \sigma(X)) \Rightarrow \tau)$, and $P' := P_0$ and $\vec{s}' := (\vec{r}, s)$.

Note that the $\to$ 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. [NW96].

5.5. **Non-total branch construction.** From here the proof of the Interpretation Theorem 5.4 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 on the LHS, or negative on the RHS, along the thread. Tracking the definition above, 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.

Proof of Theorem 5.4. Let $P : \Sigma \Rightarrow \tau$ and $\vec{s} \in |\Sigma|$ be as in the statement, and suppose for contradiction that $P\vec{s} \notin |\tau|$. Setting $P_0 := P$ and $\vec{s}_0 := \vec{s}$, we use Definition 5.17 to construct an infinite branch $(P_i : \Sigma_i \Rightarrow \tau_i)_{i<\omega}$ and associated inputs $(\vec{s}_i)_{i<\omega}$ by always setting $P_{i+1} := P'_i$, $(\Sigma_{i+1} \Rightarrow \tau_{i+1}) := (\Sigma'_i \Rightarrow \tau'_i)$ and $\vec{s}_{i+1} := \vec{s}'_i$.

Now let $(\rho_i)_{i\geq k}$ be a progressing thread along $(P_i)_{i<\omega}$, since P is progressing, and let $r_i \in \vec{s}_i$ be the corresponding input, for $i \geq k$, when ρ_i is on the LHS. Since $(\rho_i)_{i\geq k}$ is progressing, let $\mu X_1\sigma_1(X_1) > \cdots > \mu X_n\sigma_n(X_n)$ enumerate the fixed points occurring in every ρ_i positively in the LHS, and negatively in the RHS, such that $\mu X_n\sigma_n$ is the smallest infinitely often principal formula along $(\rho_i)_{i\geq k}$. Note that such a finite set of fixed points must exist since necessarily $\rho_{i+1} \preceq_{\mathrm{FL}} \rho_i$, and so the thread must eventually stabilise within some Fischer-Ladner class. WLoG, no $\mu X_j\sigma_j(X_j)$, for $j < n$, is principal along $(\rho_i)_i$.

By Proposition 5.16, let $\vec{\alpha}_i = \alpha_{i1}, \ldots, \alpha_{in}$, for $i \geq k$, be the least assignments such that:

- if ρ_i is on the LHS then $r_i \in |\rho_i^{\vec{\alpha}_i}|$;
- if ρ_i is on the RHS then $P_i\vec{s}_i \notin (\rho_i)_{\vec{\alpha}_i}$.

We claim that $(\vec{\alpha}_i)_{i\geq k}$ is a monotone non-increasing sequence of ordinal assignments that does not converge:

- By construction of P_i and $\vec{s}_i$, appealing to Definition 5.17, note that for each step for which $\mu X_n\sigma_n(X_n)$ is not principal (along $(\rho_i)_i$ on the LHS) we have that $\vec{\alpha}_{i+1} \leq \vec{\alpha}_i$. Note in particular that the $\to$-cases of Definition 5.17 are designed to guarantee $\vec{\alpha}_{i+1} \leq \vec{\alpha}_i$ at $\to$-steps.
- Now, consider a μ'_l-step along the progressing thread on the critical fixed point formula,

$$\mu'_l \frac{\Gamma, \sigma_n(\mu X_n\sigma_n(X_n)) \Rightarrow \pi}{\Gamma, \mu X_n\sigma_n(X_n) \Rightarrow \pi}$$

with $\rho_i = \mu X_n\sigma_n(X_n)$. Writing $\vec{\alpha}_{in} = \alpha_{i,1}, \ldots, \alpha_{i,n-1}$ (a prefix of $\vec{\alpha}_i$) we have:

$$\begin{aligned}
|(\mu X_n\sigma_n(X_n))^{\vec{\alpha}_{in}\alpha_{in}}| &= |(\mu X_n\sigma_n^{\vec{\alpha}_{in}}(X_n))^{\alpha_{in}}| \\
&= |\sigma_n^{\vec{\alpha}_{in}\alpha_{in}}(\varnothing)| \\
&= |\sigma_n^{\vec{\alpha}_{in}\mathsf{s}\alpha'_{in}}(\varnothing)| \\
&= |\sigma_n^{\vec{\alpha}_{in}}(|\sigma_n^{\vec{\alpha}_{in}\alpha'_{in}}(\varnothing)|)| \\
&= |\sigma_n^{\vec{\alpha}_{in}}(|(\mu X_n\sigma_n^{\vec{\alpha}_{in}}(X_n))^{\alpha'_{in}}|)| \\
&= |\sigma_n^{\vec{\alpha}_{in}}(|(\mu X_n\sigma_n(X_n))^{\vec{\alpha}_{in}\alpha'_{in}}|)|
\end{aligned}$$

for some α'_{in} since α_{ij} must be a successor ordinal. Thus indeed $\vec{\alpha}_{i+1} < \vec{\alpha}_i$.
This contradicts the well-foundedness of ordinals. □

6. 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 relies.

6.1. Language and theories of 'second-order' arithmetic.

Let us recall $\mathcal{L}_2$, the language of 'second-order' arithmetic, e.g. as given in [Sim99]. It extends the language of arithmetic $\mathcal{L}_1$ by:

- an additional sort of *sets*, whose variables are written X, Y etc. Individuals of $\mathcal{L}_1$ are considered of *number* sort.
- an *elementhood* (or *application*) relation $\in$ relating the number sort to the set sort. I.e. there are formulas $t \in X$ (also Xt) when t is a number term and X is a set variable. (We will not consider any non-variable set terms here.)

When speaking about the 'free variables' of a formula, we always include set variables as well as individual variables. We shall assume a De Morgan basis of connectives, namely $\vee, \wedge, \exists, \forall$ with negation only on atomic formulas. Hence, we say that a formula φ is *positive* (or *negative*) in X if no (every, resp.) subformula Xt occurs under a negation. Let us note that we have an analogue of 'functoriality' in predicate logic:

Lemma 6.1 (Monotonicity)**.** *Let $\varphi(X, x)$ be positive in X.*

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

Proof sketch. By (meta-level) induction on the structure of φ. □

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

We shall work with subtheories of full second-order arithmetic ($\mathsf{PA2}$) such as ACA_0, Π^1_1-CA_0, Π^1_2-CA_0 etc., whose definitions may be found in standard textbooks, e.g. [Sim99]. We shall freely use basic facts about them. For instance:

Proposition 6.2 (Some basic reverse mathematics, e.g. [Sim99])**.** *We have the following:*

(1) $\mathsf{ACA}_0 \subseteq \Pi^1_1$-$\mathsf{CA}_0 = \Sigma^1_1$-$\mathsf{CA}_0 \subseteq \Pi^1_2$-$\mathsf{CA}_0 = \Sigma^1_2$-$\mathsf{CA}_0$
(2) Π^1_1-$\mathsf{CA}_0 \vdash \mathsf{ATR}$ *(*arithmetical transfinite recursion*).*
(3) Δ^1_2-$\mathsf{AC}_0 \vdash \Sigma^1_2$-$\mathsf{AC}$ *(*axiom of choice*)*

A simple consequence of Σ^1_2-choice in Π^1_2-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 6.3 (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 6.4 (Polarised substitution lemma, Π^1_2-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 sketch. 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.

6.2. **Countable orders.** We can develop a basic theory of (countable) ordinals in even weak second-order theories, as has been done in [Sim99] and also comprehensively surveyed in [Hir05]. Let us point out that, while distinctions between natural notions of 'order comparison' are pertinent for weak theories, the theories we mainly consider contain ATR_0, for which order comparison is robust.

A *(countable) binary relation* is a pair $(X, \leq)$ where X and $\leq$ are sets, the latter construed as ranging over pairs. (Sometimes we write $\mathrm{Rel}(X, \leq)$ to specify this.) We say that $(X, \leq)$ is a *partial order*, written $\mathrm{PO}(X, \leq)$, if:

- $\forall x \in X\, x \leq x$
- $\forall x, y \in X (x \leq y \leq x \to x = x)$
- $\forall x, y, z (x \leq y \leq z \to x \leq z)$

$(X, \leq)$ is a *total order*, written $\mathrm{TO}(X, \leq)$, if it is a partial order that is *total*:

- $\forall x, y (x \leq y \vee y \leq x)$

Given a relation $\leq$, we may write $<$ for its *strict* version, given by

$$x < y := x \leq y \wedge \neg y \leq x$$

We employ similar notational conventions for other similar order-theoretic binary symbols.

We say that a binary relation $(X, \leq)$ is *well-founded*, written $\mathrm{WF}(X, \leq)$, if:

- $\forall f : \mathbb{N} \to X \exists x \neg f(x+1) < f(x)$

$(X, \leq)$ is a *well-order*, written $\mathrm{WO}(X, \leq)$, if it is a well-founded total order, i.e.:

$$\mathrm{WO}(X, \leq) := \mathrm{TO}(X, \leq) \wedge \mathrm{WF}(X, \leq)$$

Henceforth we shall write α, β etc. to range over countable binary relations. If $\alpha = (X, \leq)$ we may write $x \leq_\alpha y := x, y \in X \wedge x \leq y$, and similarly $x <_\alpha y$ for $x, y \in X \wedge x < y$. We may also write simply $x \in \alpha$ instead of $x \in X$, as abuse of notation.

It is not hard to see that ACA_0 admits an induction principle over any provable well-order (see, e.g., [Sim99]):

Fact 6.5 (Transfinite induction, ACA_0)**.** If $\mathrm{WO}(\alpha)$ then:

$$\forall X (\forall x \in \alpha (\forall y <_\alpha x\, X(y) \to X(x)) \to \forall x \in \alpha\, X(x))$$

In fact, in extensions of ACA_0, we even have (transfinite) induction on arithmetical formulas, a fact that we shall use implicitly in what follows. For instance, in Π^1_2-CA_0, we may admit (transfinite) induction on arithmetical combinations of Π^1_2 formulas.

6.3. **Comparing orders.** Following Simpson in [Sim99], given $\alpha, \beta \in \mathrm{WO}$ we write $\alpha \prec \beta$ if there is an order-isomorphism from α onto a proper initial segment of β. We also write $\alpha \approx \beta$ if α and β are order-isomorphic, and $\alpha \preceq \beta$ if $\alpha \prec \beta \vee \alpha \approx \beta$. Thanks to the uniqueness of comparison maps, we crucially have

Proposition 6.6 (ATR_0)**.** *$\prec, \preceq, \approx$ are Δ^1_1.*

We shall now state a number of well-known facts about comparison, all of which may be found in, e.g., [Sim99] or [Hir05].[4]

Proposition 6.7 (Facts about ordinal comparison, Π^1_1-CA_0)**.** *Let α, β, γ be well-orders. We have the following:*

(1) *(Comparison is a preorder)*
   (a) $\alpha \preceq \alpha$
   (b) $\alpha \preceq \beta \preceq \gamma \to \alpha \preceq \gamma$
(2) *(Comparison is pseudo-antisymmetric) If $\alpha \preceq \beta \preceq \alpha$ then $\alpha \approx \beta$.*
(3) *(Comparison is total)* $\alpha \preceq \beta \vee \beta \preceq \alpha$
(4) *(Comparison is well-founded)* $\forall F : \mathbb{N} \to \mathrm{WO}\, \exists x\ \neg F(x+1) \prec F(x)$

We shall assume basic ordinal existence principles, in particular constructions for successor ($\mathsf{s}\alpha$), addition ($\alpha + \beta$) and maximum ($\max(\alpha, \beta)$), initial segments (α_b for $b \in \alpha$), all definable and satisfying characteristic properties provably in ATR_0. We shall also make crucial use of a 'bounding' principle:

Proposition 6.8 (Σ^1_1-Bounding, ATR_0)**.** *Let $\varphi(\alpha) \in \Sigma^1_1$ with $\forall \alpha(\varphi(\alpha) \to \mathrm{WO}(\alpha))$. Then $\exists \beta \in \mathrm{WO} \forall \alpha(\varphi(\alpha) \to \alpha \preceq \beta)$.*

When appealing to Bounding, we use notation such as $\lceil \varphi \rceil$ for an ordinal bounding all $\alpha \in \mathrm{WO}$ such that $\varphi(\alpha)$.

6.4. **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 Π^1_2-CA_0 unless otherwise stated.

Remark 6.9 (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. [PY17], even for the Knaster-Tarski theorem [SY17], 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. Our operators are specified *non-uniformly*, namely via positive formulas. This amounts to a sort of non-uniform '3rd order reverse mathematics' of Knaster-Tarski, peculiar to the powerset lattice on $\mathbb{N}$.

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.

[4]Note that we have intentionally refrained from specifying the 'optimal' theories for each statement, for simplicity of exposition

Definition 6.10 ((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,

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

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

Proposition 6.11. *If* $\mathrm{WO}(\alpha)$ *then* φ^α *is* $\Delta_2^{1,+}(\vec{X}, \neg\vec{Y})$*. In particular,* $\varphi^\alpha(a, x)$ *is equivalent to:*

$$\forall G \subseteq \alpha \times \mathbb{N}(\forall b \in \alpha \forall y(\exists c <_\alpha b \varphi(Gc, y) \to Gby) \to Gax) \tag{6.1}$$

Proof. As written $\varphi^\alpha(a, x)$ is a positive Σ_1^1 combination of φ, so it is certainly $\Sigma_2^{1,+}(\vec{X}, \neg\vec{Y})$ by Lemma 6.4. So it suffices to prove the 'in particular' clause, since (6.1) is already a positive Π_1^1 combination of φ.

For this we note that we can 'merge' the two definitions into an instance of ATR, obtaining (after φ-comprehension) some H satisfying for all $b \in \alpha$ and y:

$$Hby \leftrightarrow \exists c <_\alpha b\, \varphi(Hc, y)) \tag{6.2}$$

Now we show that in fact $Hax \leftrightarrow \varphi^\alpha(a, x)$. For the left-to-right implication, we simply witness the $\exists F$ quantifier in the definition of φ^α by H. For the right-to-left implication, let $F \subseteq \alpha \times \mathbb{N}$ such that:

$$\forall b \in \alpha \forall y(Fby \to \exists c <_\alpha b \varphi(Fc, y)) \tag{6.3}$$

We show $\forall x(Fax \to Hax)$ by α-induction on a:

$$\begin{array}{rll} Fax & \implies \exists b <_\alpha a\varphi(Fb, x) & \text{by (6.3)} \\ & \implies \exists b <_\alpha a\varphi(Hb, x) & \text{by inductive hypothesis} \\ & \implies Hax & \text{by (6.2)} \end{array}$$

We may prove $Hax \leftrightarrow$ (6.1) similarly, concluding the proof. □

Eventually we will also show that $\varphi^{\mathrm{WO}} \in \Delta_2^{1,+}(\vec{X}, \neg\vec{Y})$ 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 6.12 ((Bounded) recursion)**.** *Let* $\alpha \in \mathrm{WO}$*. We have the following:*

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

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

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

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

Proof. 1 follows immediately from the equivalence between φ^α and H satisfying (6.2) in the preceding proof. For 2 we first need an intermediate result. For $b \in \alpha \in \mathrm{WO}$, let us write α_b for the initial segment of α up to (and including) b. We show,

$$\forall x(\varphi^\alpha(b, x) \leftrightarrow \varphi^{\alpha_b}(x)) \tag{6.4}$$

by transfinite induction on $b \in \alpha$. We have:

$$\begin{array}{rcll} \varphi^\alpha(b,x) & \iff & \exists c <_\alpha b\, \varphi(\varphi^\alpha(c),x) & \text{by 1} \\ & \iff & \exists c <_\alpha b\, \varphi(\varphi^{\alpha_c},x) & \text{by IH} \\ & \iff & \exists c <_\alpha b\, \varphi(\varphi^{\alpha_b}(c),x) & \text{by IH} \\ & \iff & \exists b' \in \alpha_b \exists c <_{\alpha_b} b' \varphi(\varphi^{\alpha_b}(c),x) & \text{set } b' = b \\ & \iff & \exists b' \in \alpha_b \varphi^{\alpha_b}(b',x) & \text{by 1} \\ & \iff & \varphi^{\alpha_b}(x) & \text{by dfn. of } \varphi^{\alpha_b} \end{array}$$

Now, turning back to 2, we have:

$$\begin{array}{rcll} \varphi^\alpha(x) & \iff & \exists c \in \alpha\, \varphi^\alpha(c,x) & \text{by definition} \\ & \iff & \exists c \in \alpha \exists d <_\alpha c\, \varphi(\varphi^\alpha(d),x) & \text{by 1} \\ & \iff & \exists c \in \alpha \exists d <_\alpha c\, \varphi(\varphi^{\alpha_d},x) & \text{by (6.4)} \\ & \iff & \exists c \in \alpha \exists \beta \prec \alpha_c\, \varphi(\varphi^\beta,x) & \text{by basic ordinal properties} \\ & \iff & \exists \beta \prec \alpha\, \varphi(\varphi^\beta,x) & \end{array}$$

□

Proposition 6.13 ((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) $\alpha \preceq \beta \to \forall x(\varphi^\alpha(x) \to \varphi^\beta(x))$

Proof. 1 follows directly from Bounded Recursion:

$$\begin{array}{rcll} \varphi^\alpha(a,x) & \implies & \exists c <_\alpha a\, \varphi(\varphi^\alpha(c),x) & \text{by Corollary 6.12.1} \\ & \implies & \exists c <_\alpha b\, \varphi(\varphi^\alpha(c),x) & \text{since } a \leq_\alpha b \\ & \implies & \varphi^\alpha(b,x) & \text{by Corollary 6.12.1} \end{array}$$

2 follows directly from Recursion:

$$\begin{array}{rcll} \varphi^\alpha(x) & \implies & \exists \gamma \prec \alpha\, \varphi(\varphi^\alpha,x) & \text{by Corollary 6.12.2} \\ & \implies & \exists \gamma \prec \beta\, \varphi(\varphi^\gamma,x) & \text{since } \alpha \preceq \beta \\ & \implies & \varphi^\beta(x) & \text{by Corollary 6.12.2} \end{array}$$

Definition 6.14 (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 6.15 ('Knaster-Tarski')**.** $\varphi(\mu\varphi) = \mu\varphi$*, i.e.*

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

Proof. For $\varphi(\mu\varphi) \subseteq \mu\varphi$, note that for any X with $\varphi(X) \subseteq X$ we have:

$$\begin{array}{rl} \mu\varphi\, x \to Xx & \text{by definition of } \mu\varphi \\ \varphi(\mu\varphi,x) \to \varphi(X,x) & \text{by Lemma 6.1} \\ \varphi(\mu\varphi,x) \to Xx & \text{since } \varphi(X) \subseteq X \end{array}$$

Thus $\forall X(\varphi(X) \subseteq X \to \forall x(\varphi(\mu\varphi,x) \to Xx))$, and so indeed $\varphi(\mu\varphi) \subseteq \mu\varphi$.

For $\mu\varphi \subseteq \varphi(\mu\varphi)$, we have:

$$\begin{array}{rcll} \varphi(\mu\varphi) & \subseteq & \mu\varphi & \text{by above} \\ \varphi(\varphi(\mu\varphi)) & \subseteq & \varphi(\mu\varphi) & \text{by Monotonicity Lemma 6.1} \\ \varphi(\mu\varphi) & \subseteq & \mu\varphi & \text{by comprehension on } \varphi(\mu\varphi) \end{array}$$

□

Lemma 6.16 (Closure ordinals)**.** $\exists \alpha \in \mathrm{WO}\, \varphi^{\mathrm{WO}} \subseteq \varphi^\alpha$*. I.e.*

$$\forall x(\varphi^{\mathrm{WO}}(x) \to \varphi^\alpha(x))$$

Proof sketch. We have:[5]

$$\begin{array}{lll} & \forall x(\varphi^{\mathrm{WO}}(x) \to \exists \alpha \in \mathrm{WO} \varphi^\alpha(x)) & \text{by dfn. of } \varphi^{\mathrm{WO}} \\ \therefore & \forall x \exists \alpha \in \mathrm{WO}(\varphi^{\mathrm{WO}}(x) \to \varphi^\alpha(x)) & \text{by pure logic} \\ \therefore & \exists F : \mathbb{N} \to \mathrm{WO}\, \forall x(\varphi^{\mathrm{WO}}(x) \to \varphi^{Fx}(x)) & \text{by } \Sigma^1_2\text{-choice} \\ \therefore & \forall x(\varphi^{\mathrm{WO}} \to \varphi^{\lceil F \rceil}(x)) & \text{by Prop. 6.13.2} \end{array}$$

where $\lceil F \rceil \in \mathrm{WO}$ is obtained by Bounding,[6] Proposition 6.8, such that $\forall x \lceil F \rceil \succeq Fx$. □

The main result of this subsection is:

Theorem 6.17 (LFP dual characterisation)**.** $\mu\varphi = \varphi^{\mathrm{WO}}$*, i.e.:*

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

Proof. For the left-right inclusion, $\mu\varphi \subseteq \varphi^{\mathrm{WO}}$, we show that φ^{WO} is a pre-fixed point of $\Lambda X \lambda x\, \varphi$, i.e.:

$$\forall x(\varphi(\varphi^{\mathrm{WO}}, x) \to \varphi^{\mathrm{WO}}(x))$$

By Lemma 6.16 let α such that $\varphi^{\mathrm{WO}} = \varphi^\alpha$. We have:

$$\begin{array}{rcll} \varphi(\varphi^{\mathrm{WO}}, x) & \Longrightarrow & \varphi(\varphi^\alpha, x) & \text{by assumption} \\ & \Longrightarrow & \varphi^{\mathsf{s}\alpha}(x) & \text{by Cor. 6.12.2} \\ & \Longrightarrow & \varphi^{\mathrm{WO}}(x) & \text{by definition of } \varphi^{\mathrm{WO}} \end{array}$$

Since we have access to φ^{WO} as a set, under Σ^1_2-CA, this indeed yields $\mu\varphi \subseteq \varphi^{\mathrm{WO}}$.

For the right-left inclusion, $\varphi^{\mathrm{WO}} \subseteq \mu\varphi$, let $\alpha \in \mathrm{WO}$ and we show:

$$\forall a \in \alpha(\varphi^\alpha(a, x) \to \mu\varphi\, x)$$

by transfinite induction on $a \in \alpha$. For logical complexity, recall that we indeed have access to $\mu\varphi$ as a set, since $\varphi \in \Delta^1_2$ so $\mu\varphi \in \Pi^1_2$. We have:

$$\begin{array}{rcll} \varphi^\alpha(a, x) & \Longrightarrow & \exists b <_\alpha a\, \varphi(\varphi^\alpha(b), x) & \text{by Corollary 6.12.1} \\ & \Longrightarrow & \exists b <_\alpha a\, \varphi(\mu\varphi, x) & \text{by inductive hypothesis} \\ & \Longrightarrow & \varphi(\mu\varphi, x) & \text{by vacuous quantification} \end{array}$$ □

One of the main consequences of the above result is:

Corollary 6.18 (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 amounts to a partial arithmetisation of purely descriptive characterisation of μ-definable sets in Lubarsky's work [Lub93]. Note also that, while we (in particular) obtain a Δ^1_2 bound on fixed point formulas, we crucially required $\mathsf{CA}\Pi^1_2$ to prove this, consistent with results of [Mö02] that we shall exploit in the next section to fill in our 'grand tour'.

[5] For the penultimate step, recall that we have access to φ^{WO} as a set, under Σ^1_2-comprehension.

[6] To be precise, we apply bounding on the formula $\exists x \forall a(Xa \leftrightarrow Fxa)$.

6.5. **Arithmetising the totality argument.** Thanks to the results of this section, we may duly formalise the type structure $|\cdot|$ from Section 5 within Π^1_2-CA_0 and prove basic properties. Throughout this section we shall identify terms with their codes. In the case of $\langle \mu\mathsf{LJ}^- \rangle$ and subsystems, we note that terms may still be specified finitely thanks to regularity of coderivations in $\mathsf{C}\mu\mathsf{LJ}^-$, and that syntactic equality is verifiable already in RCA_0 between different representations [Das20a, Das21]. Let us also note that the various notions of reduction, in particular $=_{\mathrm{r}'}$ and $=^{\eta}_{\mathrm{r}'}$ are Σ^0_1 relations on terms. (Note here that it is convenient that $=^{\eta}_{\mathrm{r}'}$ is formulated using only the extensionality rule, for Σ^0_1, rather than being fully extensional.)

Let us recall the $|\cdot|$ structure from Section 5. Note that, as written in Definition 5.1, these sets, a priori, climb up the analytical hierarchy due to alternation of μ and $\to$. This is where the results of the previous section come into play and serve to adequately control the logical complexity of $|\cdot|$.

Definition 6.19 (Type structure, formalised)**.** We define the following second-order formulas:

- $|X|(t) := Xt$
- $|N|(t) := \exists n\,(t =^{\eta}_{\mathrm{r}'} \underline{n})$
- $|\sigma \to \tau|(t) := \forall s(|\sigma|(s) \to |\tau|(ts))$
- $|\sigma \times \tau|(t) := |\sigma|(\mathsf{p}_0 t) \wedge |\tau|(\mathsf{p}_1 t)$
- $|\mu X \sigma|(t) := (\mu X \lambda x |\sigma|(x))(t)$

Weak theories such as RCA_0 are able to verify closure of $\langle \mathsf{C}\mu\mathsf{LJ}^- \rangle$ under conversion by a syntax analysis (see [Das21]). That each $|\sigma|$ itself is closed under conversion, cf. 5.3, is also available already in RCA_0 by (meta-level) induction on the structure of σ. More importantly, and critically, we have as a consequence of the previous section:

Corollary 6.20. *Let σ be positive in $\vec{X}$ and negative in $\vec{Y}$. $|\sigma|$ is Π^1_2-CA_0-provably $\Delta^{1,+}_2(\vec{X}, \neg\vec{Y})$.*

Proof sketch. We proceed by induction on the structure of σ, for which the critical case is when σ has the form $\mu X \tau$. By definition τ is positive in X, and so by IH $|\tau| \in \Delta^{1,+}_2(X, \vec{X}, \neg\vec{Y})$. We conclude by Corollary 6.18. □

Now, recall that a function $f : \mathbb{N}^k \to \mathbb{N}$ is *provably recursive* in a theory $T \subseteq \mathsf{PA2}$ if there is a Σ^0_1-formula $\varphi_f(\vec{x}, y)$ with:

- $\mathfrak{N} \models \varphi_f(\underline{\vec{m}}, \underline{n})$ if and only if $f(\vec{m}) = n$; and,
- $T \vdash \forall \vec{x} \exists y \varphi_f(\vec{x}, y)$.

We may formalise the entire totality argument within Π^1_2-CA_0, in a similar fashion to analogous arguments in [Das21, KPP21, Das20a], only peculiarised to the current setting. This requires some bespoke arithmetical arguments and properties of our type structure. As a consequence we shall obtain:

Theorem 6.21. *Any $\mathsf{C}\mu\mathsf{LJ}^-$-representable function on natural numbers is provably recursive in Π^1_2-CA_0.*

We give some more details for establishing the above theorem. In particular, we apply the fixed point theorems within second-order arithmetic from Section 6.4 to arithmetise the totality argument in Section 5 within Π^1_2-CA_0. We shall focus on explaining at a high level the important aspects of the formalisation, broadly following the structure of Section 5.

Notice that it will not actually be possible to prove the Interpretation Theorem 5.4 uniformly, due to Gödelian issues. Instead we will demonstrate a non-uniform version of it:

Theorem 6.22 (Nonuniform Interpretation, formalised)**.** *Let* $P : \Sigma \Rightarrow \tau$*. Then* Π^1_2-$\mathsf{CA}_0 \vdash \forall \vec{s} \in |\Sigma|.\ P\vec{s} \in |\tau|$.

Note that, from Theorem 6.22 above, Theorem 6.21 will follow directly via a version of Corollary 5.5 internal to Π^1_2-CA_0. In particular, when Σ contains only Ns and τ is N, note that the statement of Theorem 6.22, $\forall \vec{s} \in |\Sigma|.\ P\vec{s} \in |\tau|$, is indeed Π^0_2.

The rest of this section is devoted to justifying Theorem 6.22 above. Let us fix $P : \Sigma \Rightarrow \tau$ for the remainder of this section.

Formalising monotonicity and transfinite types. All the technology built up in Subsection 5.2 has been appropriately formalised in the previous Section 6.4.

Formalising closures and priorities. All the notions about closures and priorities in the totality argument involve only finitary combinatorics and are readily formalised within RCA_0.

Formalising ordinal assignments. We define ordinal assignments within Π^1_2-CA_0 relative to some $\vec{\alpha}$ varying over WO. What is important is to establish the 'positive and negative approximants' Proposition 5.16. However this follows directly from the Closure Ordinal Lemma 6.16 and the Monotonicity Lemma 6.1.

Formalising reflection of non-totality. Since P has only finitely many distinct lines, and so only finitely many distinct formulas, we will need to consider only finitely many distinct $|\sigma|$ henceforth which we combine to describe to establish the reflection of non-totality Definition 5.17 all at once within SO arithmetic. Working in Π^1_2-CA_0, let us point out that the description of $P', \Sigma', \tau', \vec{s}'$ from r and P's (finitely many) sub-coderivations $\vec{P}$ is recursive in the following oracles:

- $|\sigma|$, for each σ occurring in P, which is Δ^1_2 by Corollary 6.20; and,
- finding the least $\alpha \in \mathrm{WO}$ such that $|\sigma^\alpha|(t)$; this is Δ^1_2 by fixing the closure ordinal, cf. 6.16, say $\gamma \in \mathrm{WO}$ of $\mu|\sigma|$, and searching for the least appropriate $a \in \gamma$ instead; again this is Δ^1_2;
- finding an inhabitant s of some nonempty $|\sigma|$, for which we can simply take the 'least' (seen as a natural number coding it), and so is Δ^1_2.

Consequently we have that Definition 5.17, the description of $P', \Sigma', \tau', \vec{s}'$ from $\mathsf{r}, \vec{P}$, is indeed a Π^1_2-CA_0-provably Δ^1_2 formula in $P', \Sigma', \tau', \vec{s}', \mathsf{r}, \vec{P}$.

Formalising the non-total branch construction. The 'non-total' branch constructed in the proof of the Interpretation Theorem 5.4 is recursive in the 'reflecting non-totality' definition, and so we have access to it as a set within Π^1_2-CA_0.

One subtlety at this point is that Π^1_2-CA_0 needs to 'know' that P is indeed progressing. However earlier work on the cyclic proof theory of arithmetic [Sim17, Das20b], building on the reverse mathematics of ω-automaton theory [KMPS19], means that this poses us no problem at all:

Proposition 6.23 (Formalised cyclic proof checking [Das20b])**.** RCA_0 *proves that* P *is progressing, i.e. that every infinite branch has a progressing thread.*

From here we readily have that there is a progressing thread $(\rho_i)_{i\geq k}$ and inputs r_i along along the non-total branch. The assignment of ordinals $(\vec{\alpha}_i)_{i\geq k}$ along the branch follows by the 'positive and negative approximants' result (formalisation of Proposition 5.16). The verification that $\vec{\alpha}_{i+1} \leq \vec{\alpha}_i$ for non-critical steps follows by inspection of the 'reflecting non-totality' definition (formalisation of Definition 5.17). Finally, for the critical fixed point unfolding, we need that α_{in} is a successor. For this we need that the approximant at a limit ordinal is a union of smaller approximants, for which we use Recursion, cf. Corollary 6.12.

This concludes the argument for Theorem 6.22, and so also of Theorem 6.21.

7. 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 total 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 [Mö02]. Unlike that work, we shall formulate $\mu\mathsf{PA}$ in a purely first-order fashion in order to facilitate the ultimate realisability interpretation.

7.1. Language of arithmetic with fixed points.

We consider an extension of PA whose language is closed under (parameterised) least (and greatest) fixed points. Throughout this section we shall only use logical symbols among $\wedge, \to, \exists, \forall$, without loss of generality.

Convention 7.1. We shall henceforth assume, without loss of generality, that the language of arithmetic $\mathcal{L}_1$ contains a function symbol for each primitive recursive function definition. All arithmetic theories we consider, like PA, HA and their extensions, will contain the defining equational axioms for each of these function symbols. Note that these formulations are just *definitional extensions* of the usual versions of PA and HA. However, they allow us to construe Δ_0 formulas as just equations, simplifying some of the metamathematics herein.

Let us extend the language of arithmetic $\mathcal{L}_1$ by countably many predicate symbols, written X, Y etc., that we shall refer to as 'set variables'. In this way we shall identify $\mathcal{L}_1$-formulas with the arithmetical formulas of the language of second-order arithmetic $\mathcal{L}_2$, but we shall remain in the first-order setting for self-containment. As such, when referring to the 'free variables' of a formula, we include both set variables and number 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 $t \in \mu X\lambda x\varphi$ (or even $\mu\varphi\, t$ when X, x are clear from context) is a formula with free variables $\vec{X}, \vec{x}, \vec{y}$.

We also write $t \in \nu X\lambda x\varphi(X)$ for $t \notin \mu X\lambda x\neg\varphi(\neg X)$, a suggestive notation witnessing the De Morgan duality between μ and ν (in classical logic).

Remark 7.2 ($\mathcal{L}_\mu$ as a proper extension of $\mathcal{L}_2$)**.** Again referring to the identification of $\mathcal{L}_1$ formulas with arithmetical $\mathcal{L}_2$ formulas, we may construe $\mathcal{L}_\mu$ as a formal extension of $\mathcal{L}_2$ by certain relation symbols. Namely $\mathcal{L}_\mu$ may be identified with the closure of $\mathcal{L}_2$ under:

- if φ is an *arithmetical* formula with free variables among $\vec{X}, X, \vec{x}, x$, and in which X occurs positively, then there is a relation symbol $\mu X\lambda x\varphi$ taking $|\vec{X}|$ set inputs and $|\vec{x}, x|$ number inputs.

In this case, for arithmetical $\varphi(\vec{X}, X, \vec{x}, x)$, we simply write, say, $t \in \mu X\lambda x\varphi(\vec{A}, X, \vec{t}, x)$ instead of $\mu X\lambda x\varphi(X, \vec{X}, x, \vec{x})\vec{A}\vec{t}t$. Let us note that this is indeed the route taken by Möllerfeld. Instead we choose to treat the construct $\mu \cdot \lambda \cdot$ syntactically as 'binder'.

Semantically we construe $\mu X\lambda x\varphi$ in the intended model as a bona fide least fixed point of the (parametrised) operator defined by φ. Namely, in the sense of the remark above, we can set $(\mu X\lambda x\varphi)^{\mathfrak{N}}(\vec{A}, \vec{a})$ to be the least fixed point of the operator $(\Lambda X\lambda x\varphi(\vec{A}, X, \vec{a}, x))^{\mathfrak{N}}$, for all sets $\vec{A}$ and numbers $\vec{a}$. Note that by simple algebraic reasoning this forces $\nu X\lambda x\varphi$ to be interpreted as the analogous *greatest* fixed point in $\mathfrak{N}$.

As in earlier parts of this work, we shall frequently suppress variables in formulas to denote abstractions, e.g. for formulas $\varphi(X)$ and $\psi(x)$, with X, x clear from context, we may write $\varphi(\psi)$ for the formula obtained by replacing each subformula Xt of $\varphi(X)$ by $\psi(t)$.

7.2. **Theories μPA and μHA.** 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 7.3 (Theory)**.** The theory μPA is the extension of PA by the following axioms for formulas $\varphi(X, x)$ and $\psi(x)$:

- Pre_φ: $\forall x(\varphi(\mu\varphi, y) \to \mu\varphi\, x)$
- $\mathsf{Ind}_{\varphi,\psi}$: $\forall x(\varphi(\psi, x) \to \psi(x)) \to \forall x(\mu\varphi\, x \to \psi(x))$

We sometimes omit the subscripts of the axiom names above. We may construe μPA as a proper fragment of full second-order arithmetic PA2 by the interpretation:

$$t \in \mu X\lambda x\varphi \quad := \quad \forall X(\forall x(\varphi(X, x) \to Xx) \to Xt) \tag{7.1}$$

The axioms above are readily verified by mimicking a standard textbook algebraic proof of Knaster-Tarski in the logical setting, cf. Proposition 6.15.

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 7.4 (Implied by [Mö02])**.** Π^1_2-CA_0 *is arithmetically conservative over* μPA*.*

We set μHA to be the intuitionistic counterpart of μPA. I.e. μHA is axiomatised by Heyting Arithmetic HA (extended to the language $\mathcal{L}_\mu$) and the schemes Pre and Ind in Definition 7.3 above.

Once again, we may construe μHA as a proper fragment of full second-order Heyting Arithmetic HA2 by (7.1) above. By essentially specialising known conservativity results for second-order arithmetic, we may thus show that μPA and μHA provably define the same recursive functions on natural numbers.

Proposition 7.5 (Implied by [Tup04])**.** μPA *(so also* Π^1_2-CA_0*) is* Π^0_2*-conservative over* μHA*.*

We give a self-contained argument for this result in Appendix C, essentially by composing the Friedman-Dragalin A-translation and Gödel-Gentzen negative translation.

7.3. **Relativisation to $\mathbb{N}$.** It will be convenient for our realisability argument to work with a notion of *abstract realisability*, where realisability commutes with quantifiers in favour of explicit relativisation of quantifiers to suitable domains [Tro98]. The reason for this is that, a priori, all quantifiers of arithmetic are relativised to $\mathbb{N}$, but construing so for the fixed point axioms leads to type mismatch during realisability. For instance, one would naturally like to realise induction axiom for natural numbers by iter_N, for which it is natural to consider the $\forall$ of the inductive step unrelativised, whereas the $\forall$ of the conclusion should certainly be relativised to natural numbers. The same phenomenon presents for the Ind axioms more generally. Thus we shall include such relativisations explicitly to handle this distinction.

We introduce a new unary predicate symbol N and write $\mathcal{L}_\mu^{\mathsf{N}} := \mathcal{L}_\mu \cup \{\mathsf{N}\}$.[7] N will morally stand for the fixed point $\mu X \lambda x (x = 0 \vee \exists y (Xy \wedge x = \mathsf{s}y))$, computing the natural numbers. However we shall rather take logically equivalent formulations of the prefix and induction axioms for N that are *negative*, in fact ultimately realised by our analogous specialisations of the iter and in rules for N earlier:

- $\mathsf{Pre}_{\mathsf{N}}^0$: $\mathsf{N}\underline{0}$
- $\mathsf{Pre}_{\mathsf{N}}^{\mathsf{s}}$: $\forall x (\mathsf{N}x \to \mathsf{N}\mathsf{s}x)$
- $\mathsf{Ind}_{\mathsf{N},\varphi}$: $\varphi(0) \to \forall x (\varphi(x) \to \varphi(\mathsf{s}x)) \to \forall x (\mathsf{N}x \to \varphi(x))$

We sometimes write $\forall x^{\mathsf{N}} \varphi := \forall x (\mathsf{N}x \to \varphi)$ indicating that the universal quantifier is *relativised* to N. We similarly write $\exists x^{\mathsf{N}} \varphi := \exists x (\mathsf{N}x \wedge \varphi)$. In arithmetic all quantifiers are implicitly relativised in this way, by virtue of the induction axiom schema. However note that the 'correct' formulation of induction above, induced by the fixed point definition of N, has a non-relativised universal quantifier for the step case. This will turn out to be an important refinement in order to avoid type mismatches when conducting realisability. For this reason, our realisability argument must work from a theory in which this distinction is native:

Definition 7.6. Write $\mu\mathsf{HA}^-$ for the theory defined like $\mu\mathsf{HA}(\mathcal{L}_\mu^{\mathsf{N}})$ but, instead of the numerical induction axioms, we include the axioms $\mathsf{Pre}_{\mathsf{N}}^0$, $\mathsf{Pre}_{\mathsf{N}}^{\mathsf{s}}$ and $\mathsf{Ind}_{\mathsf{N}}$ defined earlier. I.e. $\mu\mathsf{HA}^-$ is the intuitionistic theory over $\mathcal{L}_\mu^{\mathsf{N}}$ including all the axioms of Robinson Arithmetic, all the defining equations of primitive recursive function symbols, all the Pre_φ and $\mathsf{Ind}_{\varphi,\psi}$ axioms, and all the axioms $\mathsf{Pre}_{\mathsf{N}}^0$, $\mathsf{Pre}_{\mathsf{N}}^{\mathsf{s}}$ and $\mathsf{Ind}_{\mathsf{N},\varphi}$.

The notation here is suggestive of our analogous notation for negative fragments of $\mu\mathsf{LJ}$ and $\mathsf{C}\mu\mathsf{LJ}$ earlier. Indeed we shall soon see that our notion of 'realising type' for $\mu\mathsf{HA}^-$ will have image over $N, \times, \to, \mu$. First, let us verify that $\mu\mathsf{HA}^-$ indeed interprets $\mu\mathsf{HA}$.

Definition 7.7. For formulas φ of $\mathcal{L}_\mu$ we define φ^{N} a formula of $\mathcal{L}_\mu^{\mathsf{N}}$ by:

- $(s = t)^{\mathsf{N}} := s = t$
- $(\varphi \star \psi)^{\mathsf{N}} := \varphi^{\mathsf{N}} \star \psi^{\mathsf{N}}$ for $\star \in \{\wedge, \to\}$
- $(\exists x \varphi)^{\mathsf{N}} := \exists x (\mathsf{N}x \wedge \varphi)$
- $(\forall x \varphi)^{\mathsf{N}} := \forall x (\mathsf{N}x \to \varphi)$
- $(t \in \mu X \lambda x \varphi)^{\mathsf{N}} := t \in \mu X \lambda x (\mathsf{N}x \wedge \varphi^{\mathsf{N}})$

We extend this translation to (definable) predicates, e.g. writing $\varphi^{\mathsf{N}}(\vec{X}, \vec{x}) := \varphi(\vec{X}, \vec{x})^{\mathsf{N}}$ and, in particular, $(\mu\varphi)^{\mathsf{N}} x := (x \in \mu\varphi)^{\mathsf{N}}$. Specialising a well-known reduction from second-order arithmetic to pure second-order logic, we have:

[7] We use N to avoid confusion with our earlier type N for natural numbers.

Proposition 7.8. *If* $\mu\mathsf{HA} \vdash \varphi$ *then* $\mu\mathsf{HA}^- \vdash \varphi^{\mathsf{N}}$.

Proof. The Robinson axioms are already part of $\mu\mathsf{HA}^-$ and, since they are universal statements, so too their relativisations. The same argument applies for the defining equations of primitive recursive function symbols.

Each number induction axiom,

$$\psi(\underline{0}) \to \forall x(\psi(x) \to \psi(\mathsf{s}x)) \to \forall x\psi(x) \tag{7.2}$$

has N-translation,

$$\psi^{\mathsf{N}}(\underline{0}) \to \forall x^{\mathsf{N}}(\psi^{\mathsf{N}}(x) \to \psi^{\mathsf{N}}(\mathsf{s}x)) \to \forall x^{\mathsf{N}}(\psi^{\mathsf{N}}(x))$$

which, by the $\mathsf{Pre}_{\mathsf{N}}$ axioms and pure logic, is equivalent to:

$$(\mathsf{N}\underline{0} \wedge \psi^{\mathsf{N}}(\underline{0}) \to \forall x((\mathsf{N}x \wedge \psi^{\mathsf{N}}(x)) \to (\mathsf{N}\mathsf{s}x \wedge \psi^{\mathsf{N}}(\mathsf{s}x))) \to \forall x^{\mathsf{N}}(\mathsf{N}x \wedge \psi^{\mathsf{N}}(x))$$

This is just an instance of $\mathsf{Ind}_{\mathsf{N}}$ with invariant $\mathsf{N}x \wedge \psi^{\mathsf{N}}(x)$.

Finally let us consider the fixed point axioms. First notice that a Pre axiom,

$$\forall y(\varphi(\mu\varphi, y) \to y \in \mu\varphi)$$

has N-relativisation,

$$\forall y \in \mathsf{N}(\varphi^{\mathsf{N}}((\mu\varphi)^{\mathsf{N}}, y) \to y \in (\mu\varphi)^{\mathsf{N}})$$

which is logically equivalent to the Pre axiom for $(\mu\varphi)^{\mathsf{N}}$.

Next, for an Ind_φ axiom,

$$\forall x(\varphi(\psi, x) \to \psi(x)) \to \forall x(x \in \mu\varphi \to \psi(x)) \tag{7.3}$$

we derive its N-relativisation as follows,

$$\begin{array}{lll} & \forall x((\mathsf{N}x \wedge \varphi^{\mathsf{N}}(\psi^{\mathsf{N}}, x)) \to \psi^{\mathsf{N}}(x)) \to \forall y(y \in (\mu\varphi)^{\mathsf{N}} \to \psi^{\mathsf{N}}(y)) & \text{by } \mathsf{Ind} \text{ for } (\mu\varphi)^{\mathsf{N}} \\ \implies & \forall x \in \mathsf{N}(\varphi^{\mathsf{N}}(\psi^{\mathsf{N}}, x) \to \psi^{\mathsf{N}}(x)) \to \forall y \in \mathsf{N}(y \in (\mu\varphi)^{\mathsf{N}} \to \psi^{\mathsf{N}}(y)) & \text{by pure logic} \end{array}$$

where the last line is just the N-relativisation of (7.3). □

7.4. **An abstract 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 4. For convenience we shall employ the following convention:

Convention 7.9. We henceforth identify terms of $\mathcal{L}_1$ with their corresponding definitions in $\mu\mathsf{LJ}^-$. We shall furthermore simply identify closed terms of arithmetic with the numerals they reduce to under $=_{\mathrm{r}}$. Note that, for s, t terms of $\mathcal{L}_1$, we indeed have $s =_{\mathrm{r}} t \implies \mu\mathsf{HA}^- \vdash s = t$.

At least one benefit of the convention above is that we avoid any confusion arising from metavariable clash between terms of arithmetic and terms of $\langle \mu\mathsf{LJ}^- \rangle$. This convention is motivated by the clauses of our realisability judgement for atomic formulas.

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 $\mathcal{L}_\mu^{\mathsf{N}}$ by construing each realisability candidate as a unary predicate symbol. The idea is that each $\cdot An$ consists of the just the realisers of the sentence $A\underline{n}$.

Definition 7.10 (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:

- $t\,\mathbf{r}\,(\underline{m} = \underline{n})$ if $\underline{m} =_{\mathrm{r}} t =_{\mathrm{r}} \underline{n}$.

- $t \mathbf{r} \mathsf{N}\underline{n}$ if $t =_{\mathrm{r}} \underline{n}$.
- $t \mathbf{r} A\underline{n}$ if tAn.
- $t \mathbf{r} \varphi_0 \wedge \varphi_1$ if $\mathsf{p}_0 t \mathbf{r} \varphi_0$ and $\mathsf{p}_1 t \mathbf{r} \varphi_1$.
- $t \mathbf{r} \varphi \to \psi$ if, whenever $s \mathbf{r} \varphi$, we have $ts \mathbf{r} \psi$.
- $t \mathbf{r} \exists x \varphi(x)$ if for some $n \in \mathbb{N}$ we have $t \mathbf{r} \varphi(\underline{n})$.
- $t \mathbf{r} \forall x \varphi(x)$ if for all $n \in \mathbb{N}$ we have $t \mathbf{r} \varphi(\underline{n})$.
- $t \mathbf{r} \underline{n} \in \mu X \lambda x \varphi(X, x)$ if $t \mathbf{r} \forall x(\varphi(A, x) \to Ax) \to A\underline{n}$ for all candidates A.

Definition 7.11 (Realising type)**.** The *realising type* of a (possibly open) formula φ without candidate symbols, written $\mathtt{t}(\varphi)$, is given by:

- $\mathtt{t}(s = t) := N$
- $\mathtt{t}(\mathsf{N}t) := N$
- $\mathtt{t}(Xt) := X$
- $\mathtt{t}(\varphi \wedge \psi) := \mathtt{t}(\varphi) \times \mathtt{t}(\psi)$
- $\mathtt{t}(\varphi \to \psi) := \mathtt{t}(\varphi) \to \mathtt{t}(\psi)$
- $\mathtt{t}(\exists x \varphi) := \mathtt{t}(\varphi)$
- $\mathtt{t}(\forall x \varphi) := \mathtt{t}(\varphi)$
- $\mathtt{t}(t \in \mu X \lambda x \varphi) := \mu X \mathtt{t}(\varphi)$

7.5. **Closure under $=_{\mathrm{r}}$ and realising induction.** Our realisability model is compatible with our notion of conversion from $\mu\mathsf{LJ}$:

Lemma 7.12 (Closure under $=_{\mathrm{r}}$)**.** *If $t \mathbf{r} \varphi$ and $t =_{\mathrm{r}} t'$ then $t' \mathbf{r} \varphi$.*

Proof. By induction on the structure of φ:

- if φ is $\underline{m} = \underline{n}$ then still $\underline{m} =_{\mathrm{r}} t' =_{\mathrm{r}} \underline{n}$ by symmetry and transitivity of $=_{\mathrm{r}}$.
- if φ is $\mathsf{N}\underline{n}$ then still $t' =_{\mathrm{r}} \underline{n}$ by symmetry and transitivity of $=_{\mathrm{r}}$.
- if φ is $A\underline{n}$ then still $t'An$ by definition of candidate.
- if φ is $\varphi_0 \wedge \varphi_1$ then we have $\mathsf{p}_0 t \mathbf{r} \varphi_0$ and $\mathsf{p}_1 t \mathbf{r} \varphi_1$, and so by IH and context-closure of $=_{\mathrm{r}}$ we have $\mathsf{p}_0 t' \mathbf{r} \varphi_0$ and $\mathsf{p}_1 t' \mathbf{r} \varphi_1$, thus indeed $t' \mathbf{r} \varphi$.
- if φ is $\varphi_0 \to \varphi_1$ then for any $s \mathbf{r} \varphi_0$ we have $ts \mathbf{r} \varphi_1$, and so also $t's \mathbf{r} \varphi_1$ by IH and context-closure of $=_{\mathrm{r}}$. Since choice of s was arbitrary we are done.
- if φ is $\exists x \varphi'(x)$ then there is some $n \in \mathbb{N}$ with $t \mathbf{r} \varphi'(\underline{n})$. So by IH we have $t' \mathbf{r} \varphi'(\underline{n})$, and so also $t' \mathbf{r} \varphi$.
- if φ is $\forall x \varphi'(x)$ then for each $n \in \mathbb{N}$ we have $t \mathbf{r} \varphi'(\underline{n})$, and so also $t' \mathbf{r} \varphi'(\underline{n})$ by IH. Since choice of n was arbitrary we are done.
- if φ is $\mu\varphi' \underline{n}$ then, for all candidates A we have $t \mathbf{r} \forall x(\varphi'(A, x) \to Ax) \to A\underline{n}$, and so also $t' \mathbf{r} \forall x(\varphi'(A, x) \to Ax) \to A\underline{n}$ by IH. Since choice of A was arbitrary we are done. □

Note that, despite its simplicity, the above lemma has the following consequence, by consideration of the candidate $\{(t, n) : t \mathbf{r} \psi(n)\}$:

Lemma 7.13. *If $t \mathbf{r} \mu\varphi \underline{n}$ then, for any formula $\psi(x)$, $t \mathbf{r} \forall x(\varphi(\psi, x) \to \psi(x)) \to \psi(\underline{n})$.*

This allows us to realise all the induction axioms for fixed points:

Proposition 7.14 (Realising Induction)**.** *Let $\sigma(X) = \mathtt{t}(\varphi(X, x))$ and $\tau = \mathtt{t}(\psi(x))$. Then $\mathsf{iter}_{\sigma,\tau} \mathbf{r} \mathsf{Ind}_{\varphi,\psi}$.*

Proof. We shall omit subscripts to lighten the syntax. Let $u \,\mathbf{r}\, \forall x(\varphi(\psi, x) \to \psi(x))$ and $v \,\mathbf{r}\, \mu\varphi\, \underline{n}$. We need to show that $\mathsf{iter}\, u\, v \,\mathbf{r}\, \psi(n)$:

$$\begin{array}{rll} v \;\mathbf{r}\; \forall x(\varphi(\psi, x) \to \psi(x)) \to \psi(\underline{n}) & \text{since } v \,\mathbf{r}\, \mu\varphi\, \underline{n} \text{ and by Lemma 7.13} \\ v\, u \;\mathbf{r}\; \psi(\underline{n}) & \text{by } \mathbf{r} \to \text{ since } u \,\mathbf{r}\, \forall x(\varphi(\psi, x) \to \psi(x)) \\ \mathsf{iter}\, u\, v \;\mathbf{r}\; \psi(\underline{n}) & \text{by } =_{\mathrm{r}} \text{ and Lemma 7.12} \end{array}$$ □

7.6. **Functoriality and realising prefix axioms.** As expected we will realise Pre by in. For this we need a 'functoriality lemma' establishing a semantics for functors of $\mu\mathsf{LJ}$ within our realisability model. This is because the reductions for in introduce functors.

In fact, due to the fact that we admit inductive types as primitive, with native μ_l and μ_r rules, we shall have to establish the realisability of Pre and the functoriality properties *simultaneously*.

Lemma 7.15 (Functoriality and realising Pre)**.** *Let* $\mathtt{t}(\varphi(\vec{Y}, Z, z)) = \sigma(\vec{Y}, Z)$. *Then for all* $\vec{\tau} = \mathtt{t}(\vec{\psi}(x))$ *we have:*

(1) $\mathsf{in}_{\sigma(\vec{\tau})} \,\mathbf{r}\, \mathsf{Pre}_{\varphi(\vec{\psi})}$

(2) *Let* $t \,\mathbf{r}\, \forall z(\chi(z) \to \chi'(z))$ *. Then:*

  (a) *if* $\varphi(\vec{Y}, Z, z)$ *is positive in* Z *then* $\sigma(\vec{\tau}, t) \,\mathbf{r}\, \forall z(\varphi(\vec{\psi}, \chi, z) \to \varphi(\vec{\psi}, \chi', z))$.

  (b) *if* $\varphi(\vec{Y}, Z, z)$ *is negative in* Z *then* $\sigma(\vec{\tau}, t) \,\mathbf{r}\, \forall z(\varphi(\vec{\psi}, \chi', z) \to \varphi(\vec{\psi}, \chi, z))$.

Before proving this, let us make some pertinent remarks:

Remark 7.16 (Comparison to F)**.** For the reader used to fixed points via second-order encodings, it is perhaps surprising that functoriality and realisability of Pre are mutually dependent. To recall, in system F, we usually set $\mu X\sigma(X) := \forall X((\sigma(X) \to X) \to X)$. From here we might set the appropriate functor as,

$$\mu X\sigma(X, t) := \Lambda X((\sigma(X, t) \to X) \to X)$$

whence the appropriate functoriality properties, analogous to (2) of Lemma 7.15, are readily established directly, without appeal to the realisation of Pre. However the above is not (explicitly) how $\mu\mathsf{LJ}$ defines the fixed point functors.

While a second-order calculus introduces $\forall$ by way of an eigenvariable representing an *arbitrary* prefixed point, our μ_r rule works precisely because $\mu X\sigma(X)$ is the *least* fixed point, and so is already sufficiently general. This can be realised by appropriate proof transformations, themselves relying on the μ_r rule. It is for this reason that the mutual dependency above presents in our setting.

Proof of Lemma 7.15. We proceed by induction on the structure of $\varphi(\vec{Y}, Z, z)$, proving both (1) and (2) simultaneously. More precisely, (1) will rely on instances of (2) for the same φ, and (2) will rely on smaller instances of both (1) and (2).

We have $\mathsf{Pre}_{\varphi(\vec{\psi})} = \forall z(\varphi(\vec{\psi}, \mu\varphi(\vec{\psi}), z) \to \mu\varphi(\vec{\psi})\, z)$. So to prove (1) let,

$$u \,\mathbf{r}\, \varphi(\vec{\psi}, \mu\varphi(\vec{\psi}), \underline{n}) \tag{7.4}$$

and, by $\mathbf{r}\forall$ and $\mathbf{r} \to$, we need to show $\mathsf{in}_{\sigma(\vec{\tau})}\, u \,\mathbf{r}\, \mu\varphi(\vec{\psi})\, \underline{n}$. Now, by $\mathbf{r}\mu$ and $\mathbf{r} \to$, let C be a candidate and,

$$v \,\mathbf{r}\, \forall z(\varphi(\vec{\psi}, C, z) \to Cz) \tag{7.5}$$

so that it suffices to show $\mathsf{in}_{\sigma(\vec{\tau})}\, u\, v\, \mathbf{r}\, C\underline{n}$. We have:[8]

$$\begin{array}{rll}
\mathsf{iter}\ v\ \mathbf{r}\ \forall z(\mu\varphi(\vec{\psi})\, z \to Cz) & & \text{by Proposition 7.14, (7.5) and } \mathbf{r}\to \\
\sigma(\vec{\tau}, \mathsf{iter}\ v)\ \mathbf{r}\ \varphi(\vec{\psi}, \mu\varphi(\vec{\psi}), \underline{n}) \to \varphi(\vec{\psi}, C, \underline{n}) & & \text{by IH for Item 2 and } \mathbf{r}\forall \\
\sigma(\vec{\tau}, \mathsf{iter}\ v)\, u\ \mathbf{r}\ \varphi(\vec{\psi}, C, \underline{n}) & & \text{by (7.4) and } \mathbf{r}\to \\
v(\sigma(\vec{\tau}, \mathsf{iter}\ v)\, u)\ \mathbf{r}\ C\underline{n} & & \text{by (7.5) and } \mathbf{r}\forall \text{ and } \mathbf{r}\to \\
\mathsf{in}_{\sigma(\vec{\tau})}\, u\, v\ \mathbf{r}\ C\underline{n} & & \text{by } =_{\mathrm{r}} \mathsf{in} \text{ and Lemma 7.12}
\end{array}$$

To prove (2) the critical case is when $\varphi(\vec{Y}, Z, z)$ is a fixed point formula $\underline{m} \in \mu X\lambda x\varphi'(\vec{Y}, Z, z, X, x)$. We consider only the case when Z is positive, (2a). As before we shall simply write, say $\mu\varphi'(\vec{Y}, Z, z)$ for $\mu X\lambda x\varphi'(\vec{\psi}, Z, z, X, x)$. To prove (2a) let $n \in \mathbb{N}$ and

$$u\ \mathbf{r}\ \varphi(\vec{\psi}, \chi, \underline{n}) \qquad (\text{i.e. } u\, \mathbf{r}\, \mu\varphi'(\vec{\psi}, \chi, \underline{n})\, \underline{m}) \tag{7.6}$$

so, by $\mathbf{r}\forall$ and $\mathbf{r}\to$ we need to show $\sigma(\vec{\tau}, t)\, u\, \mathbf{r}\, \varphi(\vec{\psi}, \chi', \underline{n})$, i.e. $\sigma(\vec{\tau}, t)\, u\, \mathbf{r}\, \mu\varphi'(\vec{\psi}, \chi', \underline{n})\, \underline{m}$. So, by $\mathbf{r}\mu$, let A be a candidate and:

$$v\ \mathbf{r}\ \forall x(\varphi'(\vec{\psi}, \chi', \underline{n}, A, x) \to Ax) \tag{7.7}$$

We need to show that $\sigma(\vec{\tau}, t)\, u\, v\, \mathbf{r}\, A\underline{m}$.

Now let $\sigma'(\vec{Y}, Z, X) = \mathtt{t}(\varphi'(\vec{Y}, Z, z, X, x))$. Again we may similarly write simply $\mu\sigma'(\vec{Y}, Z)$ for $\mu X\sigma'(\vec{Y}, Z, X)$ (which is just $\sigma(\vec{Y}, Z)$). First, by IH(1) for $\sigma'(\vec{Y}, Z, X)$, we have:

$$\mathsf{in}_{\sigma'(\vec{\tau},\gamma')}\, \mathbf{r}\, \forall x(\varphi'(\vec{\psi}, \chi', \underline{n}, \mu\varphi'(\vec{\psi}, \chi', \underline{n}), x) \to \mu\varphi'(\vec{\psi}, \chi', \underline{n})\, x) \tag{7.8}$$

We claim that:

$$\mu_r\, \sigma'(\vec{\tau}, t, \sigma(\vec{\tau}, \gamma'))\ \mathbf{r}\ \forall x(\varphi'(\vec{\psi}, \chi, \underline{n}, \mu\varphi'(\vec{\psi}, \chi', \underline{n}), x) \to \mu\varphi'(\vec{\psi}, \chi', \underline{n})\, x) \tag{7.9}$$

To prove this let $w\, \mathbf{r}\, \varphi'(\vec{\psi}, \chi, \underline{n}, \mu\varphi'(\vec{\psi}, \chi', \underline{n}), \underline{k})$ and we show $\mu_r\, \sigma'(\vec{\tau}, t, \sigma(\vec{\tau}, \gamma'))\, w\, \mathbf{r}\, \mu\varphi'(\vec{\psi}, \chi', \underline{n})\, \underline{k}$:

$$\begin{array}{rll}
\sigma'(\vec{\tau}, t, \mu X\sigma(\vec{\tau}, \gamma'))\, w\ \mathbf{r}\ \varphi'(\vec{\psi}, \chi', \underline{n}, \mu\varphi'(\vec{\psi}, \chi', \underline{n}), \underline{k}) & & \text{by IH(2) for } \varphi', \mathbf{r}\forall \text{ and } \mathbf{r}\to \\
\mathsf{in}_{\sigma'(\vec{\tau},\gamma')}(\sigma'(\vec{\tau}, t, \mu X\sigma(\vec{\tau}, \gamma'))\, w)\ \mathbf{r}\ \mu\varphi'(\vec{\psi}, \chi', \underline{n})\, \underline{k} & & \text{by (7.8) and } \mathbf{r}\forall \text{ and } \mathbf{r}\to \\
\mu_r\, \sigma'(\vec{\tau}, t, \sigma(\vec{\tau}, \gamma'))\, w\ \mathbf{r}\ \mu\varphi'(\vec{\psi}, \chi', \underline{n})\, \underline{k} & & \text{by } =_{\mathrm{r}}
\end{array}$$

Now by (7.9), $\mathbf{r}\mu$ and Lemma 7.13 we have,

$$u\, \mathbf{r}\, \forall x(\varphi'(\vec{\psi}, \chi, \underline{n}, \mu\varphi'(\vec{\psi}, \chi', \underline{n}), x) \to \mu\varphi'(\vec{\psi}, \chi', \underline{n})\, x) \to \mu\varphi'(\vec{\psi}, \chi', \underline{n})\, \underline{m} \tag{7.10}$$

Now we prove $\sigma(\vec{\tau}, t)\, u\, v\, \mathbf{r}\, A\underline{m}$ as follows:

$$\begin{array}{rll}
u\, (\mu_r\, \sigma'(\vec{\tau}, t, \sigma(\vec{\tau}, \gamma')))\ \mathbf{r}\ \mu\varphi'(\vec{\psi}, \chi', \underline{n})\, \underline{m} & & \text{by (7.9) and } \mathbf{r}\to \\
\mathsf{iter}\, (\mu_r\, \sigma'(\vec{\tau}, t, \sigma(\vec{\tau}, \gamma')))\, u\ \mathbf{r}\ \mu\varphi'(\vec{\psi}, \chi', \underline{n})\, \underline{m} & & \text{by } =_{\mathrm{r}} \\
\mathsf{iter}\, (\mu_r\, \sigma'(\vec{\tau}, t, \sigma(\vec{\tau}, \gamma')))\, u\, v\ \mathbf{r}\ A\underline{m} & & \text{by } \mathbf{r}\mu \text{ and (7.7)} \\
\mu\sigma'(\vec{\tau}, t)\, u\, v\ \mathbf{r}\ A\underline{m} & & \text{by defintion of } \mu\text{-functor} \\
\sigma(\vec{\tau}, t)\, u\, v\ \mathbf{r}\ A\underline{m} & &
\end{array}$$

The remaining steps for functoriality, (2), are routine, and for these we shall simply suppress the parameters $\vec{\psi}, \vec{\tau}$ henceforth, writing simply $\varphi(Z, z)$ instead of $\varphi(\vec{Y}, Z, z)$ and $\sigma(Z)$ instead of $\sigma(\vec{Y}, Z)$. This is because we shall never vary the parameters $\vec{\psi}, \vec{\tau}$ in the remaining cases. We give only the cases for Item 2a, the ones for Item 2b being similar. We

[8]The types for iter are omitted but determined by context.

need to show that $\sigma(t)\,\mathbf{r}\,\forall z(\varphi(\chi,z)\to\varphi(\chi',z))$ so let $n\in\mathbb{N}$ and $u\,\mathbf{r}\,\varphi(\chi,\underline{n})$ and let us show that $\sigma(t)\,u\,\mathbf{r}\,\varphi(\chi',\underline{n})$, under $\mathbf{r}\forall$ and $\mathbf{r}\to$.

- If $\varphi(Z,z)=Zs(z)$ then $\sigma(t)=t$. So we need $t\,\mathbf{r}\,\forall z(\chi(s(z))\to\chi'(s(z)))$. Let $n\in\mathbb{N}$ and, since already $t\,\mathbf{r}\,\forall z(\chi(z)\to\chi'(z))$ by assumption, we have $t\,\mathbf{r}\chi(\underline{s(n)})\to\chi(\underline{s(n)})$ as required.
- If $\varphi(Z,z)$ is any other atomic formula θ then $\forall z(\varphi(\chi,z)\to\varphi(\chi',z))$ is just $\forall z(\theta\to\theta)$ and $\sigma(t)$ is just $\mathsf{id}\,\mathbf{r}\,\forall z(\theta\to\theta)$ by $\mathbf{r}\forall$ and $\mathsf{id}=_{\mathrm{r}}$.
- Suppose $\varphi(Z,z)$ is $\varphi_0(Z,z)\wedge\varphi_1(Z,z)$. Write $\sigma_i(Z):=\mathtt{t}(\varphi_i(Z,z))$ for $i\in\{0,1\}$ so that $\sigma(Z)=\sigma_0(Z)\times\sigma_1(Z)$. We need to show that $\sigma(t)\,\mathbf{r}\,\forall z(\varphi(\chi,z)\to\varphi(\chi',z))$, so let $n\in\mathbb{N}$ and $u\,\mathbf{r}\,\varphi(\chi,\underline{n})$ and let us show that $\sigma(t)\,u\,\mathbf{r}\,\varphi(\chi',\underline{n})$:

$$\begin{array}{rl}
\mathsf{p}_i u\ \mathbf{r}\ \varphi_i(\chi,\underline{n}) & \text{for } i\in\{0,1\} \text{ by } \mathbf{r}\wedge\\
\sigma_i(t)(\mathsf{p}_i u)\ \mathbf{r}\ \varphi_i(\chi',\underline{n}) & \text{for } i\in\{0,1\} \text{ by IH and } \mathbf{r}\forall \text{ and } \mathbf{r}\to\\
\mathsf{p}_i(\sigma(t)\,u)\ \mathbf{r}\ \varphi_i(\chi',\underline{n}) & \text{for } i\in\{0,1\} \text{ by form of } \sigma(t) \text{ and } =_{\mathrm{r}}\\
\sigma(t)\,u\ \mathbf{r}\ \varphi(\chi',\underline{n}) & \text{by } \mathbf{r}\wedge
\end{array}$$

- Suppose $\varphi(Z,z)$ is $\varphi_0(Z,z)\to\varphi_1(Z,z)$ with $\varphi_0(Z,z)$ negative in Z and $\varphi_1(Z,z)$ positive in Z. Write $\sigma_i(Z):=\mathtt{t}(\varphi_i(Z,z))$ for $i\in\{0,1\}$ so that $\sigma(Z)=\sigma_0(Z)\to\sigma_1(Z)$. We need to show that $\sigma(t)\,\mathbf{r}\,\forall z(\varphi(\chi,z)\to\varphi(\chi',z))$ so let $n\in\mathbb{N}$ and $u\,\mathbf{r}\,\varphi(\chi,\underline{n})$ and let us show that $\sigma(t)\,u\,\mathbf{r}\,\varphi(\chi',\underline{n})$. By the form of $\varphi(Z,z)$, and by $\mathbf{r}\to$, let $v\,\mathbf{r}\,\varphi_0(\chi',\underline{n})$ so that it suffices to show $\sigma(t)\,u\,v\,\mathbf{r}\,\varphi_1(\chi',\underline{n})$:

$$\begin{array}{rl}
\sigma_0(t)\,v\ \mathbf{r}\ \varphi_0(\chi,\underline{n}) & \text{by IH and } \mathbf{r}\forall \text{ and } \mathbf{r}\to\\
u\,(\sigma_0(t)\,v)\ \mathbf{r}\ \varphi_1(\chi,\underline{n}) & \text{by } \mathbf{r}\to\\
\sigma_1(t)\,(u\,(\sigma_0(t)\,v))\ \mathbf{r}\ \varphi_1(\chi',\underline{n}) & \text{by IH and } \mathbf{r}\forall \text{ and } \mathbf{r}\to\\
\sigma(t)\,u\,v\ \mathbf{r}\ \varphi_1(\chi',\underline{n}) & \text{by form of } \sigma(t) \text{ and } =_{\mathrm{r}}
\end{array}$$

- Suppose $\varphi(Z,z)$ is $\exists y\varphi'(Z,z,y)$. Write $\sigma'(Z):=\mathtt{t}(\varphi'(Z,z,y))$ so that $\sigma(Z)=\sigma'(Z)$. We need to show that $\sigma(t)\,\mathbf{r}\,\forall z(\varphi(\chi,z)\to\varphi(\chi',z))$ so let $n\in\mathbb{N}$ and $u\,\mathbf{r}\,\varphi(\chi,\underline{n})$ and let us show that $\sigma(t)\,u\,\mathbf{r}\,\varphi(\chi',\underline{n})$, under $\mathbf{r}\forall$ and $\mathbf{r}\to$.

$$\begin{array}{rl}
u\ \mathbf{r}\ \exists y\varphi'(\chi,\underline{n},y) & \text{by assumption and form of } \varphi(Z,z)\\
u\ \mathbf{r}\ \varphi'(\chi,\underline{n},\underline{k}) & \text{for some } k\in\mathbb{N} \text{ by } \mathbf{r}\exists\\
\sigma'(t)\,u\ \mathbf{r}\ \varphi'(\chi',\underline{n},\underline{k}) & \text{for some } k\in\mathbb{N} \text{ by IH and } \mathbf{r}\forall \text{ and } \mathbf{r}\to\\
\sigma'(t)\,u\ \mathbf{r}\ \exists y\varphi'(\chi',\underline{n},y) & \text{by } \mathbf{r}\exists\\
\sigma(t)\,u\ \mathbf{r}\ \varphi(\chi',\underline{n}) & \text{by forms of } \sigma(Z) \text{ and } \varphi(Z,z)
\end{array}$$

- Suppose $\varphi(Z,z)$ is $\forall y\varphi'(Z,z,y)$. Write $\sigma'(Z):=\mathtt{t}(\varphi'(Z,z,y))$ so that $\sigma(Z)=\sigma'(Z)$. We need to show that $\sigma(t)\,\mathbf{r}\,\forall z(\varphi(\chi,z)\to\varphi(\chi',z))$ so let $n\in\mathbb{N}$ and $u\,\mathbf{r}\,\varphi(\chi,\underline{n})$ and let us show that $\sigma(t)\,u\,\mathbf{r}\,\varphi(\chi',\underline{n})$ under $\mathbf{r}\forall$ and $\mathbf{r}\to$.

$$\begin{array}{rl}
u\ \mathbf{r}\ \forall y\varphi'(\chi,\underline{n},y) & \text{by assumption and form of } \varphi(Z,z)\\
u\ \mathbf{r}\ \varphi'(\chi,\underline{n},\underline{k}) & \text{for all } k\in\mathbb{N} \text{ by } \mathbf{r}\forall\\
\sigma'(t)\,u\ \mathbf{r}\ \varphi'(\chi',\underline{n},\underline{k}) & \text{for all } k\in\mathbb{N} \text{ by IH and } \mathbf{r}\forall \text{ and } \mathbf{r}\to\\
\sigma'(t)\,u\ \mathbf{r}\ \forall y\varphi'(\chi',\underline{n},y) & \text{by } \mathbf{r}\forall\\
\sigma(t)\,u\ \mathbf{r}\ \varphi(\chi',\underline{n}) & \text{by forms of } \sigma(Z) \text{ and } \varphi(Z,z)
\end{array}$$

□

7.7. **Putting it all together.**

Theorem 7.17 (Soundness)**.** *If $\mu\mathsf{HA}^- \vdash \varphi$ then there is a typed $\langle \mu\mathsf{LJ}^- \rangle_{N,\times,\to,\mu}$ term $t : \mathtt{t}(\varphi)$ such that $t\,\mathbf{r}\,\varphi$.*

Proof. The axioms and rules of intuitionistic predicate logic are realised as usual, not using the fixed point rules. The basic arithmetical axioms of $\mu\mathsf{HA}^-$ are just universal closures of true equations between primitive recursive terms, which are realised by the corresponding derivations of those terms in $\mu\mathsf{LJ}^-$.[9] The axiom schemes Pre and Ind are realised by in and iter of appropriate types, by Lemma 7.15 and Proposition 7.14 respectively. It remains to consider the N-specific axioms, which are realised by the analogous N-specific rules of $\mu\mathsf{LJ}^-$:

- $\underline{0}\,\mathbf{r}\,\mathsf{Pre}^0_{\mathsf{N}}$, as this is just $\underline{0}\,\mathbf{r}\,\mathsf{N}\underline{0}$, which follows by definition of $\mathbf{r}\mathsf{N}$.
- $\mathsf{s}\,\mathbf{r}\,\mathsf{Pre}^{\mathsf{s}}_{\mathsf{N}}$. Let $n \in \mathbb{N}$ and $u\,\mathbf{r}\,\mathsf{N}\underline{n}$, i.e. $u =_{\mathrm{r}} \underline{n}$, so that, by $\mathbf{r}\forall$ and $\mathbf{r}\to$, it suffices to show that $\mathsf{s}u\,\mathbf{r}\,\mathsf{N}\mathsf{s}\underline{n}$. We have that $\mathsf{s}u =_{\mathrm{r}} \underline{n+1} = \mathsf{s}\underline{n}$, and so indeed $\mathsf{s}u\,\mathbf{r}\,\mathsf{N}\mathsf{s}\underline{n}$ as required.
- $\mathsf{iter}_N\,\mathbf{r}\,\mathsf{Ind}_{\mathsf{N}}$. Let $u\,\mathbf{r}\,\varphi(\underline{0})$ and $v\,\mathbf{r}\,\forall x(\varphi(x) \to \varphi(\mathsf{s}x))$ so that it suffices, by $\mathbf{r}\to$, to show that $\mathsf{iter}_N u\,v\,\mathbf{r}\,\forall x^{\mathsf{N}}\varphi(x)$. For this let $n \in \mathbb{N}$ and $w\,\mathbf{r}\,\mathsf{N}\underline{n}$, and we show $\mathsf{iter}_N u\,v\,w\,\mathbf{r}\,\varphi(\underline{n})$ by induction on n:
  - If $n = 0$ then $w =_{\mathrm{r}} \underline{0}$ so $\mathsf{iter}_N u\,v\,w =_{\mathrm{r}} \mathsf{iter}_N u\,v\,0 =_{\mathrm{r}} u\,\mathbf{r}\,\varphi(\underline{0})$ by assumption.
  - If $n = n' + 1$ then we have:

$$\begin{array}{rl} \mathsf{iter}_N u\,v\,\underline{n}'\ \mathbf{r}\ \varphi(\underline{n}') & \text{by IH} \\ v\,(\mathsf{iter}_N u\,v\,\underline{n}')\ \mathbf{r}\ \varphi(\mathsf{s}\underline{n}') & \text{by } \mathbf{r}\forall \text{ and } \mathbf{r}\to \\ \mathsf{iter}_N u\,v\,w\ \mathbf{r}\ \varphi(\underline{n}) & \text{by } =_{\mathrm{r}} \end{array}$$

□

Corollary 7.18. *Any provably total recursive function of $\mu\mathsf{PA}$ is representable in $\mu\mathsf{LJ}^-$.*

Proof. Assume $\mu\mathsf{PA} \vdash \forall\vec{x}\exists!y\, s(\vec{x},y) = t(\vec{x},y)$, without loss of generality.[10] So we have,

$$\begin{array}{rll} & \mu\mathsf{PA} \vdash \forall\vec{x}\exists y\, s(\vec{x},y) = t(\vec{x},y) & \\ \Longrightarrow & \mu\mathsf{HA} \vdash \forall\vec{x}\exists y\, s(\vec{x},y) = t(\vec{x},y) & \text{by Proposition 7.5} \\ \Longrightarrow & \mu\mathsf{HA}^- \vdash \forall\vec{x}^{\mathsf{N}}\exists y^{\mathsf{N}} s(\vec{x},y) = t(\vec{x},y) & \text{by Proposition 7.8} \\ \Longrightarrow & u\,\mathbf{r}\,\forall\vec{x}^{\mathsf{N}}\exists y^{\mathsf{N}} s(\vec{x},y) = t(\vec{x},y) & \end{array}$$

for some u of $\langle \mu\mathsf{LJ}^- \rangle_{N,\times,\to,\mu}$ (of appropriate type) by Theorem 7.17. Thus:

$$\begin{array}{rll} & \forall\vec{m} \in \mathbb{N}\ u\,\underline{\vec{m}}\,\mathbf{r}\,\exists y^{\mathsf{N}} s(\underline{\vec{m}},y) = t(\underline{\vec{m}},y) & \text{by } \mathbf{r}\forall,\ \mathbf{r}\to \text{ and } \mathbf{r}\mathsf{N} \\ \Longrightarrow & \forall\vec{m} \in \mathbb{N}\exists n \in \mathbb{N}\,(\mathsf{p}_0(u\,\underline{\vec{m}})\,\mathbf{r}\,\mathsf{N}\underline{n} \text{ and } \mathsf{p}_1(u\,\underline{\vec{m}})\,\mathbf{r}\,s(\underline{\vec{m}},\underline{n}) = t(\underline{\vec{m}},\underline{n})) & \text{by } \mathbf{r}\exists,\ \mathbf{r}\wedge \text{ and } \mathbf{r}\mathsf{N} \\ \Longrightarrow & \forall\vec{m} \in \mathbb{N}\exists n \in \mathbb{N}\,(\mathsf{p}_0(u\,\underline{\vec{m}}) =_{\mathrm{r}} \underline{n} \text{ and } s(\underline{\vec{m}},\underline{n}) =_{\mathrm{r}} t(\underline{\vec{m}},\underline{n})) & \text{by } \mathbf{r}\mathsf{N} \text{ and } \mathbf{r}= \end{array}$$

Thus, as the graph of $s(\vec{x},y) = t(\vec{x},y)$, in the standard model $\mathfrak{N}$ is functional, by assumption, it is computed by $\lambda\vec{x}^{\vec{N}}(\mathsf{p}_0(u\,\vec{x}))$. By Theorem 4.13 this is equivalent to some $\mu\mathsf{LJ}^-$ derivation too. □

[9] Recall that $\mu\mathsf{LJ}^-$ represents at least all the (even higher-order) primitive recursive functions.

[10] Recall we have included each primitive recursive function definition as a symbol of $\mathcal{L}_1$ and defining equations among the axioms of PA, HA and extensions.

8. 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 $\mu\mathsf{LJ}$ and $\mathsf{C}\mu\mathsf{LJ}$ as the functions provably recursive in the second-order theory Π^1_2-CA_0. This extends the tradition of such correspondences between arithmetic theories an type systems, in particular including between PA and system T due to Gödel [Gö58], and between $\mathsf{PA2}$ and F due to Girard [Gir72]. In a sense ours is a somewhat intuitive result in light of Möllerfeld's work [Mö02], identifying the latter's arithmetical theorems with the extension of ACA_0 by least and greatest general fixed points.

Our characterisation also applies to aforementioned (circular) systems of linear logic, namely $\mu\mathsf{MALL}$ and its circular counterpart, from [BM07,Bae12,BDS16,EJ21,EJS21,BDKS22, DS19, DPS21, DJS22]. This result was given the preliminary conference version of this paper [CD23], but will be expanded upon in a separate full paper.

Referring to Rathjen's ordinal notation system in [Rat95], this means that all these systems represent just the functions computable by recursion on ordinals of $\mathfrak{T}[\theta = \omega]$. 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 1.1.

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. [SY17] and the present work.

Acknowledgements

The authors would like to thank the anonymous referees for their diligent work in reviewing this paper, which has surely improved its presentation. The authors would like to thank Igor Walukiewicz, Alexis Saurin, Graham Leigh, Pierre Clairambault, Colin Riba, Ulrich Berger and Paul Levy for several insightful conversations around this work.

References

[AF98] Jeremy Avigad and Solomon Feferman. Gödel's functional ("dialectica") interpretation. *Handbook of proof theory*, 137:337–405, 1998.

[Bae12] David Baelde. Least and greatest fixed points in linear logic. *ACM Trans. Comput. Log.*, 13(1):2:1–2:44, 2012. doi:10.1145/2071368.2071370.

[BDKS22] 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: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 63:1–63:13. ACM, 2022. doi:10.1145/3531130.3533375.

[BDS16] David Baelde, Amina Doumane, and Alexis Saurin. Infinitary proof theory: the multiplicative additive case. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 42:1–42:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CSL.2016.42.

[BM07] David Baelde and Dale Miller. Least and greatest fixed points in linear logic. In Nachum Dershowitz and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 14th International Conference, LPAR 2007, Yerevan, Armenia, October 15-19, 2007, Proceedings*, volume 4790 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2007. doi:10.1007/978-3-540-75560-9_9.

[BT21] Ulrich Berger and Hideki Tsuiki. Intuitionistic fixed point logic. *Annals of Pure and Applied Logic*, 172(3):102903, 2021. doi:10.1016/j.apal.2020.102903.

[CD23] Gianluca Curzi and Anupam Das. Computational expressivity of (circular) proofs with fixed points. In *LICS*, pages 1–13, 2023. doi:10.1109/LICS56636.2023.10175772.

[Cla09] Pierre Clairambault. Least and greatest fixpoints in game semantics. In Ralph Matthes and Tarmo Uustalu, editors, *6th Workshop on Fixed Points in Computer Science, FICS 2009, Coimbra, Portugal, September 12-13, 2009*, pages 39–45. Institute of Cybernetics, 2009. URL: http://cs.ioc.ee/fics09/proceedings/contrib5.pdf.

[Cla13] Pierre Clairambault. Strong functors and interleaving fixpoints in game semantics. *RAIRO Theor. Informatics Appl.*, 47(1):25–68, 2013. doi:10.1051/ita/2012028.

[Das20a] Anupam Das. A circular version of Gödel's T and its abstraction complexity. *CoRR*, abs/2012.14421, 2020. URL: https://arxiv.org/abs/2012.14421, arXiv:2012.14421.

[Das20b] Anupam Das. On the logical complexity of cyclic arithmetic. *Logical Methods in Computer Science*, Volume 16, Issue 1, January 2020. doi:10.23638/LMCS-16(1:1)2020.

[Das21] Anupam Das. On the logical strength of confluence and normalisation for cyclic proofs. In Naoki Kobayashi, editor, *6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, 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. doi:10.4230/LIPIcs.FSCD.2021.29.

[DJS22] 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, *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022, December 18-20, 2022, IIT Madras, Chennai, India*, volume 250 of *LIPIcs*, pages 35:1–35:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSTTCS.2022.35.

[Dou17] 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. URL: https://tel.archives-ouvertes.fr/tel-01676953.

[DPS21] 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 2021: 23rd International Symposium on Principles and Practice of Declarative Programming, Tallinn, Estonia, September 6-8, 2021*, pages 7:1–7:15. ACM, 2021. doi:10.1145/3479394.3479402.

[DS19] Abhishek De and Alexis Saurin. Infinets: The parallel syntax for non-wellfounded proof-theory. In Serenella Cerrito and Andrei Popescu, editors, *Automated Reasoning with Analytic Tableaux and Related Methods - 28th International Conference, TABLEAUX 2019, London, UK, September 3-5, 2019, Proceedings*, volume 11714 of *Lecture Notes in Computer Science*, pages 297–316. Springer, 2019. doi:10.1007/978-3-030-29026-9_17.

[EJ21] Thomas Ehrhard and Farzad Jafarrahmani. Categorical models of linear logic with fixed points of formulas. In *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '21, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1109/LICS52264.2021.9470664.

[EJS21] Thomas Ehrhard, Farzad Jafarrahmani, and Alexis Saurin. On relation between totality semantic and syntactic validity. In *5th International Workshop on Trends in Linear Logic and Applications (TLLA 2021)*, Rome (virtual), Italy, June 2021. URL: https://hal-lirmm.ccsd.cnrs.fr/lirmm-03271408.

[Gir72] Jean-Yves Girard. *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur*. PhD thesis, Éditeur inconnu, 1972.

[Gö58] Kurt Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12(3-4):280–287, 1958.

[Hir05] Jeffry L. Hirst. A survey of the reverse mathematics of ordinal arithmetic. In Stephen G.Editor Simpson, editor, *Reverse Mathematics 2001*, Lecture Notes in Logic, page 222–234. Cambridge University Press, 2005. doi:10.1017/9781316755846.014.

[HS08] J. Roger Hindley and Jonathan P. Seldin. *Lambda-Calculus and Combinators: An Introduction*. Cambridge University Press, USA, 2 edition, 2008.

[KMPS19] Leszek Aleksander Kolodziejczyk, Henryk Michalewski, Pierre Pradic, and Michal Skrzypczak. The logical strength of Büchi's decidability theorem. *Log. Methods Comput. Sci.*, 15(2), 2019. doi:10.23638/LMCS-15(2:16)2019.

[KMV22] Clemens Kupke, Johannes Marti, and Yde Venema. Succinct Graph Representations of μ-Calculus Formulas. In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic (CSL 2022)*, volume 216 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2022.29.

[Koz83] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983. Special Issue Ninth International Colloquium on Automata, Languages and Programming (ICALP) Aarhus, Summer 1982. doi:10.1016/0304-3975(82)90125-6.

[KPP21] 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. doi:10.1145/3434282.

[Lub93] Robert S. Lubarsky. μ-definable sets of integers. *The Journal of Symbolic Logic*, 58(1):291–313, 1993. doi:10.2307/2275338.

[Men87] Nax Paul Mendler. Recursive types and type constraints in second-order lambda calculus. In *Logic in Computer Science*, 1987.

[Men91] 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. doi:10.1016/0168-0072(91)90069-X.

[Mö02] Michael Möllerfeld. *Generalized inductive definitions. The μ-calculus and Π^1_2-comprehension*. PhD thesis, University of Münster, 2002. University of Münster, https://nbn-resolving.de/urn:nbn:de:hbz:6-85659549572.

[NW96] Damian Niwinski and Igor Walukiewicz. Games for the μ-calculus. *Theor. Comput. Sci.*, 163(1&2):99–116, 1996. doi:10.1016/0304-3975(95)00136-0.

[PY17] Weiguang Peng and Takeshi Yamazaki. Two kinds of fixed point theorems and reverse mathematics. *Mathematical Logic Quarterly*, 63(5):454–461, 2017. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/malq.201600096, doi:10.1002/malq.201600096.

[Rat95] Michael Rathjen. Recent advances in ordinal analysis: Π^1_2-CA and related systems. *Bulletin of Symbolic Logic*, 1(4):468–485, 1995. doi:10.2307/421132.

[Rey74] 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. doi:10.1007/3-540-06859-7_148.

[RS22] 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.

[Sim99] Stephen G. Simpson. *Subsystems of second order arithmetic*. Perspectives in mathematical logic. Springer, 1999.

[Sim17] Alex Simpson. Cyclic arithmetic is equivalent to Peano arithmetic. In Javier Esparza and Andrzej S. Murawski, editors, *Foundations of Software Science and Computation Structures - 20th International Conference, FOSSACS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, volume 10203 of *Lecture Notes in Computer Science*, pages 283–300, 2017. doi:10.1007/978-3-662-54458-7_17.

[Stu08] Thomas Studer. On the proof theory of the modal μ-calculus. *Studia Logica: An International Journal for Symbolic Logic*, 89(3):343–363, 2008. URL: http://www.jstor.org/stable/40268983.

[SY17] Takashi Sato and Takeshi Yamazaki. Reverse mathematics and order theoretic fixed point theorems. *Arch. Math. Log.*, 56(3-4):385–396, 2017. doi:10.1007/s00153-017-0526-y.

[Tro98] 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. doi:10.1016/S0049-237X(98)80021-9.

[Tup04] Sergei Tupailo. On the intuitionistic strength of monotone inductive definitions. *J. Symb. Log.*, 69(3):790–798, 2004. doi:10.2178/jsl/1096901767.

[Ven08] Yde Venema. Lectures on the modal μ-calculus. *Renmin University in Beijing (China)*, 2008.

Appendix A. Weak and strong (co)iteration rules

Our presentation of the iteration rule (i.e., μ_l) and the coiteration rule (i.e., ν_r) for $\mu\mathsf{LJ}$ is the most permissive one, in that contexts are allowed to appear in both premises:

$$\mu_l \frac{\Gamma, \sigma(\rho) \Rightarrow \rho \quad \Delta, \rho \Rightarrow \tau}{\Gamma, \Delta, \mu X \sigma(X) \Rightarrow \tau} \qquad \nu_r \frac{\Delta \Rightarrow \tau \quad \Gamma, \tau \Rightarrow \sigma(\tau)}{\Delta, \Gamma \Rightarrow \nu X \sigma(X)} \tag{A.1}$$

Their cut-reduction rules are as in Figure 5. The reader should notice that the context Γ in both the rules of (A.1) is contracted during cut-reduction. For this reason, alternative presentations of μ_l and ν_r without the context Γ have been studied in the literature, especially in the setting of linear logic, where contraction cannot be freely used (see, e.g., [BM07])

$$\mu_l \frac{\sigma(\rho) \Rightarrow \rho \quad \Delta, \rho \Rightarrow \tau}{\Delta, \mu X \sigma(X) \Rightarrow \tau} \qquad \nu_r \frac{\Delta \Rightarrow \tau \quad \tau \Rightarrow \sigma(\tau)}{\Delta \Rightarrow \nu X \sigma(X)} \tag{A.2}$$

Single premise formulations of μ_l and ν_r for both (A.1) and (A.2) have been investigated in the literature as well (see, e.g., [Cla09]):

$$\mu_l \frac{\Gamma, \sigma(\tau) \Rightarrow \tau}{\Gamma, \mu X \sigma(X) \Rightarrow \tau} \qquad \nu_r \frac{\Gamma \Rightarrow \sigma(\tau)}{\Gamma \Rightarrow \nu X \sigma(X)} \tag{A.3}$$

$$\mu_l \frac{\sigma(\tau) \Rightarrow \tau}{\mu X \sigma(X) \Rightarrow \tau} \qquad \nu_r \frac{\tau \Rightarrow \sigma(\tau)}{\tau \Rightarrow \nu X \sigma(X)} \tag{A.4}$$

The corresponding cut-reduction rules for (A.2), (A.3), and (A.4) can be easily extracted from the ones in Figure 5.

It is worth mentioning, however, that cut-elimination fails in presence of the rules (A.3) and (A.4). By contrast, being essentially endowed with a built-in cut, the rules (A.1) and (A.2) allow cut-elimination results.

Perhaps surprisingly, all formulations of μ_l and ν_r are equivalent, meaning they can derive each other crucially using the cut rule.

Proposition A.1. *In* LJ *endowed with the rules* μ_r *and* ν_l *(Figure 3), the rules* (A.1)*,* (A.2)*,* (A.3)*, and* (A.4) *and their cut-reduction are inter-derivable.*

Proof. It suffices to show that the rules in (A.4) can derive the rules in (A.3). We only show the case of μ_l, as the case of ν_r is symmetric. W.l.o.g., we will treat Γ in (A.3) as a single formula. The case where Γ is an arbitrary context can be recovered using $\times_l$ and $\times_r$. The derivation is the following:

$$\mathsf{cut} \frac{\overset{P}{\triangledown} \quad \Gamma, \mu X.\sigma \Rightarrow \mu X.(\Gamma \times \sigma) \qquad \mu_l \dfrac{\times_l \dfrac{\Gamma, \sigma(\tau) \Rightarrow \tau}{\Gamma \times \sigma(\tau) \Rightarrow \tau}}{\mu X.(\Gamma \times \sigma) \Rightarrow \tau}}{\Gamma, \mu X.\sigma \Rightarrow \tau}$$

where P is the derivation in Figure 21. It is tedious but routine to show that the above derivation of μ_l allows us to derive the cut-reduction rule for (A.4). □

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \Rightarrow \Gamma}\,\mathsf{id}}{\Gamma \Rightarrow \Gamma}\,\mathsf{id} \quad \dfrac{}{\Gamma \Rightarrow \Gamma}\,\mathsf{id} \quad \sigma(\mu X.(\Gamma \times \sigma)) \Rightarrow \sigma(\mu X.(\Gamma \times \sigma))}{\Gamma, \Gamma \to \sigma(\mu X.(\Gamma \times \sigma)) \Rightarrow \sigma(\mu X.(\Gamma \times \sigma))}\,{\to_l}}{\Gamma, \Gamma, \Gamma \to \sigma(\mu X.(\Gamma \times \sigma)) \Rightarrow \Gamma \times \sigma(\mu X.(\Gamma \times \sigma))}\,{\times_r}}{\Gamma, \Gamma \to \sigma(\mu X.(\Gamma \times \sigma)) \Rightarrow \Gamma \times \sigma(\mu X.(\Gamma \times \sigma))}\,\mathsf{c}}{\Gamma, \Gamma \to \sigma(\mu X.(\Gamma \times \sigma)) \Rightarrow \mu X.(\Gamma \times \sigma)}\,{\mu_r}}{\Gamma, \sigma(\Gamma \to \sigma(\mu X.(\Gamma \times \sigma))) \Rightarrow \sigma(\mu X.(\Gamma \times \sigma))}\,\sigma}{\sigma(\Gamma \to \sigma(\mu X.(\Gamma \times \sigma))) \Rightarrow \Gamma \to \sigma(\mu X.(\Gamma \times \sigma))}\,{\to_r}}{\mu X.\sigma \Rightarrow \Gamma \to \sigma(\mu X.(\Gamma \times \sigma))}\,{\mu_l} \qquad \dfrac{\dfrac{}{\Gamma \Rightarrow \Gamma}\,\mathsf{id} \quad \sigma(\mu X.(\Gamma \times \sigma)) \Rightarrow \sigma(\mu X.(\Gamma \times \sigma))}{\Gamma \to \sigma(\mu X.(\Gamma \times \sigma)), \Gamma \Rightarrow \sigma(\mu X.(\Gamma \times \sigma))}\,{\to_l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \Rightarrow \Gamma}\,\mathsf{id} \qquad \Gamma, \mu X.\sigma \Rightarrow \sigma(\mu X.(\Gamma \times \sigma))}{\Gamma, \Gamma, \mu X.\sigma \Rightarrow \Gamma \times \sigma(\mu X.(\Gamma \times \sigma))}\,{\times_r}}{\Gamma, \Gamma, \mu X.\sigma \Rightarrow \Gamma \times \sigma(\mu X.(\Gamma \times \sigma))}\,\mathsf{c}}{\Gamma, \mu X.\sigma \Rightarrow \mu X.(\Gamma \times \sigma)}\,{\mu_r}}{\Gamma, \mu X.\sigma \Rightarrow \mu X.(\Gamma \times \sigma)}\,\mathsf{cut}$$

Figure 21: Deriving (A.3) from (A.4).

Appendix B. Proof of Proposition 3.12

We give a bespoke combination of a negative translation and a Friedman-Dragalin 'A translation' suitable for our purposes. Writing $N := \mu X(1 + X)$ and $\neg^N \sigma := \sigma \to N$. At the risk of ambiguity but to reduce syntax, we shall simply write $\neg\sigma$ for $\neg^N \sigma$ henceforth.[11] Also, we shall include the following derivable rules of $\mathsf{C}\mu\mathsf{LJ}$:

$$\neg_r \frac{\Gamma, \sigma \Rightarrow N}{\Gamma \Rightarrow \neg\sigma} \qquad \neg_l \frac{\Gamma \Rightarrow \sigma}{\Gamma, \neg\sigma \Rightarrow N} \tag{B.1}$$

$$\neg\neg_r \frac{\Gamma \Rightarrow \sigma}{\Gamma \Rightarrow \neg\neg\sigma} \qquad \neg\neg_l \frac{\Gamma, \sigma \Rightarrow N}{\Gamma, \neg\neg\sigma \Rightarrow N}$$

We define the translations $\cdot^N$ and $\cdot_N$ from arbitrary types to types over $\{N, \times, \to, \mu\}$ as follows:

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

Proposition B.1 (Substitution)**.**

- $(\sigma(\mu X.\sigma))^N = \sigma^N(\mu X.\sigma^N)$
- $(\sigma(\nu X.\sigma))^N = \sigma^N(\neg\mu X.\neg\sigma^N[\neg X/X])$
- $(\sigma(\nu X.\sigma))_N = \sigma_N(\neg\mu X.\neg\sigma^N[\neg X/X])$.

[11] Any confusion will be minimal as we shall not parametrise negation by any other A-translation in this work.

We shall extend the notations $\cdot^N$ and $\cdot_N$ to cedents, e.g. writing Σ^N and Σ_N, by distributing the translation over the list. For a sequent $\Sigma \Rightarrow \tau$, we also write $(\Sigma \Rightarrow \tau)^N$ for $\Sigma^N, \tau_N \Rightarrow N$

Definition B.2 (N-translation of steps)**.** For each inference step

$$\mathsf{r}\,\frac{\Sigma_1 \Rightarrow \tau_1 \quad \cdots \quad \Sigma_n \Rightarrow \tau_n}{\Sigma \Rightarrow \tau}$$

we define a gadget,

$$\begin{array}{c} (\Sigma_1 \Rightarrow \tau_1)^N \quad \cdots \quad (\Sigma_n \Rightarrow \tau_n)^N \\ \hline \mathsf{r}^N \\ (\Sigma \Rightarrow \tau)^N \end{array}$$

as in Figure 22. We lift this to a translation on coderivations $P \mapsto P^N$ in the obvious way.

Proposition B.3. *If P is progressing and regular then so is P^N.*

Proof. As for regularity, we observe that the translation maps any inference rule r to a gadget r^N with constantly many rules. Concerning progressiveness, it is easy to check that:

- for any infinite branch $(\Delta^j \Rightarrow \tau^j)_j$ of P' there is an infinite branch $(\Sigma^i \Rightarrow \sigma^i)_i$ of P such that $\Delta^{j_i} \Rightarrow \tau^{j_i} = (\Sigma^i \Rightarrow \sigma^i)^N$ for some sequence $j_0 < j_1 < \ldots < j_n < \ldots$
- for any thread in P connecting a formula γ in $\Sigma^i \Rightarrow \tau^i$ with a formula γ' in $\Sigma^{i+1} \Rightarrow \tau^{i+1}$ there is a thread in P' connecting γ^N in $(\Sigma^i \Rightarrow \tau^i)^N$ with a formula γ'^N in $(\Sigma^{i+1} \Rightarrow \tau^{i+1})^N$; moreover, if the former thread has progressing points then the latter has. □

Proposition B.4. *If $P_1 \rightsquigarrow_{\mathrm{cr}'} P_2$ then $P_1^N \rightsquigarrow^*_{\mathrm{cr}'} P_2^N$*

Proof. The proof is routine. We just consider the cut-reduction step eliminating a cut r of the form ν_r vs ν_l. W.l.o.g., we assume P ends with this cut. Then, P has the following shape:

$$\mathsf{cut}\,\frac{\nu_r\,\dfrac{\begin{array}{c}P_1\\ \hline \Gamma \Rightarrow \sigma(\nu X.\sigma)\end{array}}{\Gamma \Rightarrow \nu X.\sigma} \qquad \nu_l\,\dfrac{\begin{array}{c}P_2\\ \hline \Delta, \sigma(\nu X.\sigma) \Rightarrow \tau\end{array}}{\Delta, \nu X.\sigma \Rightarrow \tau}}{\Gamma, \Delta \Rightarrow \tau}$$

Which is translated into a cut between the following two coderivations:

$$\neg_r\,\dfrac{\mu_l\,\dfrac{\neg\neg_l\,\dfrac{=\,\dfrac{\begin{array}{c}{P_1}^N\\ \hline \Gamma^N, (\sigma(\nu X.\sigma))_N \Rightarrow N\end{array}}{\Gamma^N, \sigma_N(\neg\mu X.\neg\sigma^N(\neg X)) \Rightarrow N}}{\Gamma^N, \neg\sigma^N(\neg\mu X.\neg\sigma^N(\neg X)) \Rightarrow N}}{\Gamma^N, \mu X.\neg\sigma^N(\neg X) \Rightarrow N}}{\Gamma^N, (\nu X.\sigma)_N \Rightarrow N}}{\Gamma^N \Rightarrow (\nu X.\sigma)^N}$$

$$\mathsf{id}^N := \dfrac{}{\sigma^N, \sigma_N \Rightarrow N}\,{\scriptstyle \mathsf{id}}\,{\scriptstyle \neg_l} \qquad \mathsf{cut}^N := \dfrac{\Gamma^N, \sigma_N \Rightarrow N \quad \Delta^N, \sigma^N, \tau_N \Rightarrow N}{\Gamma^N, \Delta^N, \tau_N \Rightarrow N}\,{\scriptstyle \neg_r}\,{\scriptstyle \mathsf{cut}}$$

$${1_r}^N := \dfrac{}{1_N \Rightarrow N}\,{\scriptstyle \mathsf{id}} \qquad {1_l}^N := \dfrac{\Gamma^N, \sigma_N \Rightarrow N}{\Gamma^N, 1^N, \sigma_N \Rightarrow N}\,{\scriptstyle \mathsf{w}}$$

$${\to_r}^N := \dfrac{\dfrac{\Gamma^N, \sigma^N, \tau_N \Rightarrow N}{\Gamma^N, \sigma^N \Rightarrow \tau^N}\,{\scriptstyle \neg_r}}{\Gamma^N \Rightarrow \sigma^N \to \tau^N}\,{\scriptstyle \to_r}}{\Gamma^N, (\sigma \to \tau)_N \Rightarrow N}\,{\scriptstyle \neg_l} \qquad {\to_l}^N := \dfrac{\dfrac{\Gamma^N, \sigma_N \Rightarrow N}{\Gamma^N \Rightarrow \sigma^N}\,{\scriptstyle \neg_r} \quad \Delta^N, \tau^N, \gamma_N \Rightarrow N}{\Gamma^N, \Delta^N, \sigma^N \to \tau^N, \gamma_N \Rightarrow N}\,{\scriptstyle \to_l}}{\Gamma^N, \Delta^N, (\sigma \to \tau)^N, \gamma_N \Rightarrow N}\,{\scriptstyle \neg\neg_l}$$

$${\times_r}^N := \dfrac{\dfrac{\Gamma^N, \sigma_N \Rightarrow N}{\Gamma^N \Rightarrow \sigma^N}\,{\scriptstyle \neg_r} \quad \dfrac{\Delta^N, \tau_N \Rightarrow N}{\Delta^N \Rightarrow \tau^N}\,{\scriptstyle \neg_r}}{\Gamma^N, \Delta^N \Rightarrow \sigma^N \times \tau^N}\,{\scriptstyle \times_r}}{\Gamma^N, (\sigma \times \tau)_N \Rightarrow N}\,{\scriptstyle \neg_l} \qquad {\times_l}^N := \dfrac{\Gamma^N, \sigma^N, \tau^N, \gamma_N \Rightarrow N}{\Gamma^N, \sigma^N \times \tau^N, \gamma_N \Rightarrow N}\,{\scriptstyle \times_l}}{\Gamma^N, (\sigma \times \tau)^N, \gamma_N \Rightarrow N}\,{\scriptstyle \neg\neg_l}$$

$${+_r^0}^N := \dfrac{\dfrac{\dfrac{\Gamma^N, \sigma_N \Rightarrow N}{\Gamma^N, \neg\sigma^N \Rightarrow N}\,{\scriptstyle \neg\neg_l}}{\Gamma^N, \neg\sigma^N, \neg\tau^N \Rightarrow N}\,{\scriptstyle \mathsf{w}}}{\Gamma^N, (\sigma + \tau)_N \Rightarrow N}\,{\scriptstyle \times_l} \qquad {+_r^1}^N := \dfrac{\dfrac{\dfrac{\Gamma^N, \tau_N \Rightarrow N}{\Gamma^N, \neg\tau^N \Rightarrow N}\,{\scriptstyle \neg\neg_l}}{\Gamma^N, \neg\sigma^N, \neg\tau^N \Rightarrow N}\,{\scriptstyle \mathsf{w}}}{\Gamma^N, (\sigma + \tau)_N \Rightarrow N}\,{\scriptstyle \times_l}$$

$${+_l}^N := \dfrac{\dfrac{\dfrac{\Gamma^N, \sigma^N, \gamma_N \Rightarrow N}{\Gamma^N, \gamma_N \Rightarrow \neg\sigma^N}\,{\scriptstyle \neg_l} \quad \dfrac{\Gamma^N, \tau^N, \gamma_N \Rightarrow N}{\Gamma^N, \gamma_N \Rightarrow \neg\tau^N}\,{\scriptstyle \neg_l}}{\Gamma^N, \Gamma^N, \gamma_N \Rightarrow \neg\sigma^N \times \neg\tau^N}\,{\scriptstyle \times_l}}{\Gamma^N, \gamma_N \Rightarrow \neg\sigma^N \times \neg\tau^N}\,{\scriptstyle \mathsf{c}}}{\Gamma^N, (\sigma + \tau)^N, \gamma_N \Rightarrow N}\,{\scriptstyle \neg_l}$$

$${\mu_r}^N := \dfrac{\dfrac{\dfrac{\Gamma^N, (\sigma(\mu X.\sigma))_N \Rightarrow N}{\Gamma^N \Rightarrow (\sigma(\mu X.\sigma))^N}\,{\scriptstyle \neg_r}}{\Gamma^N \Rightarrow \sigma^N(\mu X.\sigma^N)}\,{=}}{\Gamma^N \Rightarrow \mu X.\sigma^N}\,{\scriptstyle \mu_r}}{\Gamma^N, (\mu X.\sigma)_N \Rightarrow N}\,{\scriptstyle \neg_l} \qquad {\mu_l}^N := \dfrac{\dfrac{\Gamma^N, (\sigma(\mu X.\sigma))^N, \gamma_N \Rightarrow N}{\Gamma^N, \sigma^N(\mu X.\sigma^N), \gamma_N \Rightarrow N}\,{=}}{\Gamma^N, \mu X.\sigma^N, \gamma_N \Rightarrow N}\,{\scriptstyle \mu_r}}{\Gamma^N, (\mu X.\sigma)^N, \gamma_N \Rightarrow N}\,{\scriptstyle \neg\neg_l}$$

$${\nu_r}^N := \dfrac{\dfrac{\dfrac{\Gamma^N, (\sigma(\nu X.\sigma))_N \Rightarrow N}{\Gamma^N, \sigma_N(\neg\mu X.\neg\sigma^N(\neg X)) \Rightarrow N}\,{=}}{\Gamma^N, \neg\sigma^N(\neg\mu X.\neg\sigma^N(\neg X)) \Rightarrow N}\,{\scriptstyle \neg\neg_l}}{\Gamma^N, \mu X.\neg\sigma^N(\neg X) \Rightarrow N}\,{\scriptstyle \mu_l}}{\Gamma^N, (\nu X.\sigma)_N \Rightarrow N}\,{\scriptstyle \neg\neg_l} \qquad {\nu_l}^N := \dfrac{\dfrac{\dfrac{\Gamma^N, (\sigma(\nu X.\sigma))^N, \gamma_N \Rightarrow N}{\Gamma^N, \sigma^N(\neg\mu X.\neg\sigma^N(\neg X)), \gamma_N \Rightarrow N}\,{=}}{\Gamma^N, \gamma_N \Rightarrow \neg\sigma^N(\neg\mu X.\neg\sigma^N(\neg X))}\,{\scriptstyle \neg_r}}{\Gamma^N, \gamma_N \Rightarrow \mu X.\neg\sigma^N(\neg X)}\,{\scriptstyle \mu_l}}{\Gamma^N, \neg\mu X.\neg\sigma^N(\neg X), \gamma_N \Rightarrow N}\,{\scriptstyle \neg_l}}{\Gamma^N, (\nu X.\sigma)^N, \gamma_N \Rightarrow N}\,{\scriptstyle \neg\neg_l}$$

Figure 22: Translation $(\_)^N$ on inference rules.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P_2{}^N}{\Delta^N, (\sigma(\nu X.\sigma))^N, \tau_N \Rightarrow N}}{\Delta^N, \sigma^N(\neg\mu X.\neg\sigma^N(\neg X)), \tau_N \Rightarrow N}=}{\Delta^N, \tau_N \Rightarrow \neg\sigma^N(\neg\mu X.\neg\sigma^N(\neg X))}\neg_r}{\Delta^N, \tau_N \Rightarrow \mu X.\neg\sigma^N(\neg X)}\mu_l}{\Delta^N, \neg\mu X.\neg\sigma^N(\neg X), \tau_N \Rightarrow N}\neg_l}{\Delta^N, (\nu X.\sigma)^N, \tau_N \Rightarrow N}\neg\neg_l$$

by applying a few cut-reduction steps we obtain the following:

$$\dfrac{\dfrac{\dfrac{P_1{}^N}{\Gamma^N, (\sigma(\nu X.\sigma))_N \Rightarrow N}}{\Gamma^N \Rightarrow (\nu X.\sigma)^N}\neg_r \qquad \dfrac{P_2{}^N}{\Delta^N, (\sigma(\nu X.\sigma))^N, \tau_N \Rightarrow N}}{\Gamma^N, \Delta^N, \tau_N \Rightarrow N}\mathsf{cut}$$

Which is the translation of the following:

$$\dfrac{\dfrac{P_1}{\Gamma \Rightarrow \sigma(\nu X.\sigma)} \qquad \dfrac{P_2}{\Delta, \sigma(\nu X.\sigma) \Rightarrow \tau}}{\Gamma, \Delta \Rightarrow \tau}\mathsf{cut}$$

□

Proposition B.5 ((De)coding for N)**.** *There are coderivations $P_{\mathsf{cod}} : N \Rightarrow N^N$ and $P_{\mathsf{dec}} : N^N \Rightarrow N$ over $\{N, \to, \times, \mu\}$ in $\mathsf{C}\mu\mathsf{LJ}$ such that, for any $n \in \mathbb{N}$:*

$$\dfrac{\dfrac{\underline{n}}{\Rightarrow N} \qquad \dfrac{P_{\mathsf{cod}}}{N \Rightarrow N^N}}{\Rightarrow N^N}\mathsf{cut} \quad \rightsquigarrow^*_{\mathrm{cr}'} \quad \dfrac{\underline{n}^N}{\Rightarrow N^N}$$

$$\dfrac{\dfrac{\underline{n}^N}{\Rightarrow N^N} \qquad \dfrac{P_{\mathsf{dec}}}{N^N \Rightarrow N}}{\Rightarrow N}\mathsf{cut} \quad \rightsquigarrow^*_{\mathrm{cr}'} \quad \dfrac{\underline{n}}{\Rightarrow N}$$

Proof. P_{cod} and P_{dec} are defined, respectively, as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{N \Rightarrow N}\,\mathsf{id}}{\Rightarrow 1^N}\,\neg_r}{\neg 1^N \Rightarrow N}\,\neg_l}{1, \neg 1^N, \neg N^N \Rightarrow N}\,\mathsf{w} \qquad \dfrac{\dfrac{\dfrac{}{N \Rightarrow N^N}\,\bullet}{N, \neg N^N \Rightarrow N}\,\neg_l}{N, \neg 1^N, \neg N^N \Rightarrow N}\,\mathsf{w}}{1 + N, \neg 1^N, \neg N^N \Rightarrow N}\,+_l}{N, \neg 1^N, \neg N^N \Rightarrow N}\,\mu_l}{N, \neg 1^N \times \neg N^N \Rightarrow N}\,\times_l}{N \Rightarrow \neg(\neg 1^N \times \neg N^N))}\,\neg_r}{N \Rightarrow \mu X.(\neg(\neg 1^N \times \neg X^N))}\,\mu r}{N \Rightarrow N^N}\,\neg\neg_r\,\bullet$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{N \Rightarrow N}\,N_r}{1^N \Rightarrow N}\,\mathsf{w}}{\Rightarrow \neg 1^N}\,\neg_r \qquad \dfrac{\dfrac{\dfrac{}{N^N \Rightarrow N}\,\neg\neg_l\,\bullet \qquad \dfrac{\dfrac{}{N \Rightarrow N}\,\mathsf{id}}{N \Rightarrow N}\,N_r^1}{N^N \Rightarrow N}\,\mathsf{cut}}{\Rightarrow \neg N^N}\,\neg_r}{\Rightarrow \neg 1^N \times \neg\neg N^N}\,\times_r}{\neg(\neg 1^N \times \neg\neg N^N) \Rightarrow N}\,\neg_l}{\mu X.(\neg(\neg 1^N \times \neg X^N)) \Rightarrow N}\,\mu_l}{N^N \Rightarrow N}\,\neg\neg_l\,\bullet$$

where 0 and s are derivations as in Figure 6. □

Theorem B.6. *If $P : \vec{N} \Rightarrow N$ in $\mathsf{C}\mu\mathsf{LJ}$ then there is coderivation $P' : \vec{N} \Rightarrow N$ over $\{N, \to, \times, \mu\}$ in $\mathsf{C}\mu\mathsf{LJ}$ such that P and P' represent the same functions on natural numbers.*

Proof. Let f be a numerical function of $\mathsf{C}\mu\mathsf{LJ}$, w.l.o.g. we can assume f is unary. This means that there is a derivation P of $N \Rightarrow N$ such that

$$\dfrac{\underline{n} \qquad P}{\Rightarrow N \qquad N \Rightarrow N}\,\mathsf{cut}{\Rightarrow N} \quad \rightsquigarrow^*_{\mathrm{cr}'} \quad \underline{f(n)} \Rightarrow N$$

By Proposition B.4 we have that, for any $n \in \mathbb{N}$:

$$\dfrac{\underline{n}^N \qquad \dfrac{P^N}{N^N, N_N \Rightarrow N}}{\Rightarrow N^N \qquad \dfrac{N^N, N_N \Rightarrow N}{N^N \Rightarrow N^N}\,\neg_r}\,\mathsf{cut}{\Rightarrow N^N} \quad \rightsquigarrow^*_{\mathrm{cr}'} \quad \underline{f(n)}^N \Rightarrow N^N$$

Then, f can be represented by the following coderivation P' over $\{N, \to, \times, \mu\}$:

$$\dfrac{\dfrac{P_{\mathsf{cod}} \quad \dfrac{P^N}{N^N, N_N \Rightarrow N}\,\neg_r}{N \Rightarrow N^N}\,\mathsf{cut} \qquad P_{\mathsf{dec}}}{N \Rightarrow N}\,\mathsf{cut}$$

□

Appendix C. Proof of Proposition 7.5

In what follows we prove Proposition 7.5.

We start with a definition a Gödel-Gentzen-style translation of formulas of $\mu\mathsf{PA}$ into formulas of $\mu\mathsf{HA}$:

$$\begin{aligned}
(t = u)^g &:= t = u \\
(t < u)^g &:= \neg\neg t < u \\
(t \in X)^g &:= \neg\neg t \in X \\
(t \in \mu X \lambda x \varphi)^g &:= \neg\neg t \in \mu X \lambda x \varphi^g \\
(\neg\varphi)^g &:= \neg\varphi^g \\
(\varphi \wedge \psi)^g &:= \varphi^g \wedge \psi^g \\
(\varphi \vee \psi)^g &:= \neg(\neg\varphi^g \wedge \neg\psi^g) \\
(\forall x \varphi)^g &:= \forall x \varphi^g \\
(\exists x \varphi)^g &:= \neg(\forall x \neg\varphi^g)
\end{aligned}$$

Some straightforward properties of the translation:

Lemma C.1.

(1) $(\varphi(\psi))^g = \varphi^g(\psi^g)$
(2) $\mu\mathsf{HA} \vdash \neg\neg\varphi^g \to \varphi^g$

Lemma C.2. *If $\mu\mathsf{PA} \vdash \varphi$ then $\mu\mathsf{HA} \vdash \varphi^g$.*

Proof. We show that any axiom of $\mu\mathsf{PA}$ is derivable in $\mu\mathsf{HA}$, where the translation on equations is justified by the fact that $\mu\mathsf{HA} \vdash \neg\neg t = s \to t = s$. We just check the axioms for the least fixed point operator μ. On the one hand, we have $\varphi^g(\mu X \lambda x \varphi^g, y) \to y \in \mu X \lambda x \varphi^g$ as an instance of (pre), and $y \in \mu X \lambda x \varphi^g \to \neg\neg y \in \mu X \lambda x \varphi^g$ by logic. We conclude by applying transitivity of implication and the generalisation rule introducing $\forall$. On the other hand, let us assume $\forall x(\varphi^g(\psi^g, x) \to \psi^g(x))$. By (ind) we obtain $y \in \mu X \lambda x \varphi^g \to \psi^g(y)$. By logic we have $\neg\neg y \in \mu X \lambda x \varphi^g \to \neg\neg\psi^g(y)$ and $\neg\neg\psi^g(y) \to \psi^g(y)$ (Lemma C.1.2). We conclude by applying transitivity of implication and deduction theorem, using Lemma C.1.1. □

We now use a standard trick called *Friedman's translation* to infer our conservativity result from the above lemma. Let ρ be a fixed formula of $\mu\mathsf{HA}$. For any formula φ of $\mu\mathsf{HA}$

we define φ^ρ as follows:

$$\begin{array}{rcll}
(t = u)^\rho & := & (t = u) \vee \rho & \\
(t < u)^\rho & := & (t < u) \vee \rho & \\
(t \in X)^\rho & := & (t \in X) \vee \rho & \\
(t \in \mu X \lambda x \varphi)^\rho & := & (t \in \mu X \lambda x \varphi^\rho) \vee \rho & \\
(\neg\varphi)^\rho & := & \neg\varphi^\rho & \\
(\varphi \circ \psi)^\rho & := & (\varphi^\rho \circ \psi^\rho) & \circ \in \{\wedge, \vee\} \\
\sigma x \varphi^\rho & := & (\sigma x.\varphi^\rho) & \sigma \in \{\forall, \exists\}
\end{array}$$

The replacement must be done without variable capture, so whenever we consider, e.g., $(\forall x\varphi)^\rho$, we must assume that x is not free in ρ.

Some straightforward properties of $(\_)^\rho$:

Lemma C.3.
(1) $(\varphi(\psi))^\rho = \varphi^\rho(\psi^\rho)$
(2) $\rho \vdash \varphi^\rho$

Lemma C.4 (Friedman's translation, adapted)**.** *If* $\mu\mathsf{HA} \vdash \varphi$ *then* $\mu\mathsf{HA} \vdash \varphi^\rho$.

Proof. It is easy to check that $\Gamma \vdash \varphi$ implies $\Gamma^\rho \vdash \varphi^\rho$, where $\Gamma^\rho := \{\gamma^\rho \mid \gamma \in \Gamma\}$, observing that $\rho \vdash \varphi^\rho$ by Lemma C.3.2. So, Friedman's translation preserves derivability. Therefore, to conclude, it suffices to show that $\mu\mathsf{HA}$ proves Friedman's translation of $\mu\mathsf{HA}$'s axioms. We only consider the cases of the axioms for the least fixed point operator μ. On the one hand, by instantiating (pre), we have $\varphi^\rho(\mu X\lambda x\varphi^\rho, y) \to y \in \mu X\lambda x\varphi^\rho$, with $\varphi^\rho(\mu X\lambda x\varphi^\rho, y) = (\varphi(\mu X\lambda x\varphi, y))^\rho$ (Lemma C.3.1), so that we can conclude $\forall y(\varphi(\mu X\lambda x\varphi, y) \to y \in \mu X\lambda x\varphi)^\rho$. On the other hand, we have to prove that $\forall x(\varphi^\rho(\psi^\rho, x) \to \psi^\rho(x))$ implies $y \in \mu X\lambda x\varphi^\rho \vee \rho \to \psi^\rho(y)$. By instantiating (ind) we have that the former implies $y \in \mu X\lambda x\varphi^\rho \to \psi^\rho(y)$. Also, from Lemma C.3.2 we infer $\rho \to \psi^\rho(y)$, so $\forall x(\varphi^\rho(\psi^\rho, x) \to \psi^\rho(x))$ implies $(y \in \mu X\lambda x\varphi^\rho) \vee \rho \to \psi^\rho(y)$. This concludes the proof. □

We can finally prove Proposition 7.5:

Proof of Proposition 7.5. Assume $\mu\mathsf{PA} \vdash \forall x\exists y A(x, y)$ for A quantifier-free. There is a primitive recursive function symbol f such that $\vdash A(x, y) \leftrightarrow f(x, y) = 0$ in both $\mu\mathsf{PA}$ and $\mu\mathsf{HA}$, so it suffices to show $\mu\mathsf{HA} \vdash \forall x\exists y f(x, y) = 0$. We set $\rho := \exists y.f(x, y) = 0$. Of course, $\mu\mathsf{PA} \vdash \rho$, so $\mu\mathsf{HA} \vdash \neg\neg\rho$ by Lemma C.2. By Lemma C.3 we have $\mu\mathsf{HA} \vdash (\rho^\rho \to \rho) \to \rho$, where $\rho^\rho = \exists y(f(x, y) = 0 \vee \exists y.(x, y) = 0)$, which is clearly equivalent to ρ. Thus, we have $\mu\mathsf{HA} \vdash (\rho \to \rho) \to \rho$, and so $\mu\mathsf{HA} \vdash \rho$. We conclude by generalising over x. □