On the logical strength of confluence and normalisation for cyclic proofs

Das, Anupam

DOI:
10.4230/LIPIcs.FSCD.2021.29

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (Harvard):
Das, A 2021, On the logical strength of confluence and normalisation for cyclic proofs. in N Kobayashi (ed.), *6th International Conference on Formal Structures for Computation and Deduction (FSCD 2021).*, 29, Leibniz International Proceedings in Informatics, LIPIcs, vol. 195, Schloss Dagstuhl, 6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, Virtual, Buenos Aires, Argentina, 17/07/21. https://doi.org/10.4230/LIPIcs.FSCD.2021.29

On the Logical Strength of Confluence and Normalisation for Cyclic Proofs

Anupam Das
University of Birmingham, UK

Abstract

In this work we address the logical strength of confluence and normalisation for non-wellfounded typing derivations, in the tradition of "cyclic proof theory" . We present a circular version CT of Gödel' s system T, with the aim of comparing the relative expressivity of the theories CT and T. We approach this problem by formalising rewriting-theoretic results such as confluence and normalisation for the underlying "coterm" rewriting system of CT within fragments of second-order arithmetic.

We establish confluence of CT within the theory RCA_0, a conservative extension of primitive recursive arithmetic and $I\Sigma_1$. This allows us to recast type structures of hereditarily recursive operations as "coterm" models of T. We show that these also form models of CT, by formalising a totality argument for circular typing derivations within suitable fragments of second-order arithmetic. Relying on well-known proof mining results, we thus obtain an interpretation of CT into T; in fact, more precisely, we interpret level-n-CT into level-$(n+1)$-T, an optimum result in terms of abstraction complexity.

A direct consequence of these model-theoretic results is weak normalisation for CT. As further results, we also show strong normalisation for CT and continuity of functionals computed by its type 2 coterms.

2012 ACM Subject Classification Theory of computation → Equational logic and rewriting; Theory of computation → Proof theory; Theory of computation → Higher order logic; Theory of computation → Lambda calculus

Keywords and phrases confluence, normalisation, system T, circular proofs, reverse mathematics, type structures

Digital Object Identifier 10.4230/LIPIcs.FSCD.2021.29

Related Version This work is based on part of the following preprint, where related results, proofs and examples may be found.
Extended Version: `https://arxiv.org/abs/2012.14421` [12]

Funding This work was supported by a UKRI Future Leaders Fellowship, *Structure vs. Invariants in Proofs*, project reference MR/S035540/1.

Acknowledgements I would like to thank Denis Kuperberg, Laureline Pinault and Damien Pous for several interesting discussions on this and related topics. I am also grateful to the anonymous reviewers for their helpful feedback and suggestions.

1 Introduction

Cyclic (or *circular*) proofs have attracted increasing attention in recent years, in settings including modal fixed point logics [28, 16, 35, 1, 18], predicate logic [8, 9, 7, 6], algebras [31, 14, 15, 13], arithmetic [33, 5, 11] and type systems [19, 4, 3]. In short, cyclic proofs are possibly non-wellfounded derivations ("coderivations") that have only finitely many distinct subderivations (and so are finitely presentable). That they are meaningful (i.e., sound, total, terminating, etc.) is usually guaranteed by some ω-regular correctness condition at the level of their infinite branches.

In this work we investigate the interface between theories of arithmetic and type systems. These two settings are fundamentally related by means of well-known *proof interpretations*, such as the functional and realisability interpretations (see, e.g., [2, 24]). In particular Gödel's system T, a simply typed classical quantifier-free theory with recursion and induction, is capable of interpreting all of Peano Arithmetic, effectively trading off quantifier complexity for abstraction complexity (i.e. type level).

Inspired by the aforementioned previous work on circular type systems, we present a circular version, CT, of T, and compare the relative expressivity of (fragments of) the two theories. More precisely, we show that the restriction of CT to level n (CT_n) is interpreted in the restriction of T to level $n+1$ (T_{n+1}). This result is optimal due to a converse result in parallel work [12] (that is beyond the scope of the present paper).[1]

Since non-wellfounded derivations do not directly admit inductive arguments and their correctness relies on nontrivial infinitary combinatorics, we employ a "proof mining" approach towards establishing this interpretation. More precisely, we formalise models of CT_n within fragments of (second-order) arithmetic, and rely on the aforementioned proof interpretations to extract corresponding terms of T_{n+1}. This builds on analogous aforementioned work in the arithmetic setting, namely [33, 11], also taking advantage of second-order theories.

Our formalisation requires us to establish a form of confluence for the underlying rewrite system of CT, which we show holds in one of the weakest second-order theories RCA_0, essentially a form of primitive recursive arithmetic with quantification over sets. Showing that these structures indeed constitute models of CT requires a formalisation of the totality argument for circular derivations, with quantifiers relativised to this structure.

A direct consequence of these model-theoretic results is weak normalisation for coterms of CT. As further results, we also show strong normalisation for CT and continuity of functionals computed by its type 2 coterms.

Relation to other work. In [26] the authors present a circular version of the underlying type system of T, using a slightly different type language including a Kleene $*$. In particular, they show that circular derivations compute, in the standard model, just the primitive recursive functionals at type 1, i.e. the natural number functions computed by terms of T, also using a formalisation within second-order theories of arithmetic. We generalise that result in several ways: (a) we optimise the result with respect to abstraction complexity; (b) we give a *logical* correspondence, at the level of theories, not just the standard model; (c) we give bona fide confluence and normalisation results for the underlying rewrite system on coterms.

This work is based on part of the (unpublished) preprint [12], where related results, proofs and examples may be found.

Preliminaries. We shall assume some basic familiarity with the underlying technical disciplines of this work, which are now well-established and form the subjects of multiple monographs. In particular, these include rewriting theory [37], subsystems of second-order arithmetic [34, 22], and Gödel's system T and program extraction [2, 24]. Some familiarity with higher-order computability [27] and metamathematics [20, 23, 38] is also helpful.

[1] It is easy, however, to see that T_n is interpreted in CT_n, as we will see in Example 2.5.

$$\mathsf{ex}\ \frac{\vec{\rho}, \sigma, \rho, \vec{\sigma} \Rightarrow \tau}{\vec{\rho}, \rho, \sigma, \vec{\sigma} \Rightarrow \tau} \qquad \mathsf{wk}\ \frac{\vec{\sigma} \Rightarrow \tau}{\vec{\sigma}, \sigma \Rightarrow \tau} \qquad \mathsf{cntr}\ \frac{\vec{\sigma}, \sigma, \sigma \Rightarrow \tau}{\vec{\sigma}, \sigma \Rightarrow \tau} \qquad \mathsf{cut}\ \frac{\vec{\sigma} \Rightarrow \sigma \quad \vec{\sigma}, \sigma \Rightarrow \tau}{\vec{\sigma} \Rightarrow \tau}$$

$$\mathsf{id}\ \frac{}{\sigma \Rightarrow \sigma} \qquad \mathsf{L}\ \frac{\vec{\sigma} \Rightarrow \rho \quad \vec{\sigma}, \sigma \Rightarrow \tau}{\vec{\sigma}, \rho \to \sigma \Rightarrow \tau} \qquad \mathsf{R}\ \frac{\vec{\sigma}, \sigma \Rightarrow \tau}{\vec{\sigma} \Rightarrow \sigma \to \tau}$$

$$0\ \frac{}{\Rightarrow N} \qquad \mathsf{s}\ \frac{}{N \Rightarrow N} \qquad \mathsf{cond}\ \frac{\vec{\sigma} \Rightarrow \tau \quad \vec{\sigma}, N \Rightarrow \tau}{\vec{\sigma}, N \Rightarrow \tau} \qquad \mathsf{rec}_\tau\ \frac{\vec{\sigma} \Rightarrow \tau \quad \vec{\sigma}, N, \sigma \Rightarrow \tau}{\vec{\sigma}, N \Rightarrow \tau}$$

Figure 1 Sequent style typing rules for T.

2 A circular version of Gödel's T

Throughout this work we shall work with theories that are *simply* or *finitely* typed. Namely **types**, written σ, τ etc., are generated by the following grammar:

$$\sigma, \tau \quad ::= \quad N \quad | \quad (\sigma \to \tau)$$

A simply typed **theory** is a multi-sorted (classical) first-order theory, whose sorts are just the simple types, equipped with **application** operators $\circ_{\sigma,\sigma\to\tau}$ for each pair $\sigma, \sigma \to \tau$ of types, as usual. (Typed) **terms**, written s, t etc., are formed from constants of a simply typed language under typed application. We simply write $t\,s$ for the application of a term t of type $\sigma \to \tau$ to a term s of type σ. As usual we may sometimes omit parentheses, e.g. writing $r\,s\,t$ instead of $((r\,s)\,t)$.

In this work, we always assume **intensional equality** for simply typed theories. Namely we have binary relation symbols $=_\sigma$ for each type σ, axiomatised by reflexivity, $t =_\sigma t$, and the Leibniz property, $(s =_\sigma t \wedge \varphi(s)) \supset \varphi(t)$, for each formula φ and terms s, t of type σ.

2.1 Sequent calculus presentation of T terms

Sequent calculi give us a way to write typed terms that are more succinct with respect to type level, and also enjoy elegant proof theoretic properties, e.g. cut-elimination. Importantly, the induced relations between type occurrences makes it easier to define our correctness criterion for non-wellfounded derivations later.

▶ **Definition 2.1** (Sequent calculus). *Sequents are expressions $\vec{\sigma} \Rightarrow \tau$, where $\vec{\sigma}$ is a list of types and τ is a type. The typing rules for T are given in Figure 1.*

Here, and throughout this subsection, colours of each type occurrence in typing rules may be ignored for now and will become relevant later in Section 2.2.

Each rule instance (or **step**) determines a constant of the appropriate type. E.g., a step $\dfrac{\vec{\rho} \Rightarrow \rho \quad \vec{\sigma} \Rightarrow \sigma}{\vec{\tau} \Rightarrow \tau}$ is a constant of type $(\vec{\rho} \to \rho) \to (\vec{\sigma} \to \sigma) \to \vec{\tau} \to \tau$.[2] In this way, we may identify each derivation with a term obtained by just repeatedly applying rule instances, starting from the conclusion, to its subderivations. Note that this "combinatory" approach, treating rule instances as constants rather than, say, meta-level operations on λ-terms, ensures that this association of a term to a derivation is *continuous*. This is important for our later association of "coterm" to a "coderivation" .

[2] Here and elsewhere we freely write, say, $\vec{\rho} \to \rho$ for $\rho_1 \to \cdots \to \rho_n \to \rho$ when $\vec{\rho}$ is a list $(\rho_1, \ldots, \rho_n)$.

$$\begin{aligned}
\mathsf{id}\ x &= x \\
\mathsf{ex}\ t\ \vec{x}\ x\ y\ \vec{y} &= t\ \vec{x}\ y\ x\ \vec{y} \\
\mathsf{wk}\ t\ \vec{x}\ x &= t\ \vec{x} \\
\mathsf{cntr}\ t\ \vec{x}\ x &= t\ \vec{x}\ x\ x
\end{aligned}
\qquad
\begin{aligned}
\mathsf{cut}\ s\ t\ \vec{x} &= t\ \vec{x}\ (s\ \vec{x}) \\
\mathsf{L}\ s\ t\ \vec{x}\ y &= t\ \vec{x}\ (y\ (s\ \vec{x})) \\
\mathsf{R}\ t\ \vec{x}\ x &= t\ \vec{x}\ x
\end{aligned}$$

$$\begin{aligned}
\mathsf{rec}\ s\ t\ \vec{x}\ 0 &= s\ \vec{x} \\
\mathsf{rec}\ s\ t\ \vec{x}\ \mathsf{s}z &= t\ \vec{x}\ z\ (\mathsf{rec}\ s\ t\ \vec{x}\ z)
\end{aligned}
\qquad
\begin{aligned}
\mathsf{cond}\ s\ t\ \vec{x}\ 0 &= s\ \vec{x} \\
\mathsf{cond}\ s\ t\ \vec{x}\ \mathsf{s}z &= t\ \vec{x}\ z
\end{aligned}$$

Figure 2 Equational axiomatisation of T, where z is a variable of type N.

1. $\neg\, \mathsf{s}x = 0$
2. $\mathsf{s}x = \mathsf{s}y \supset x = y$
(Ind) If $\vdash \varphi(0)$ and $\vdash \varphi(x) \supset \varphi(\mathsf{s}x)$ then $\vdash \varphi(t)$, for φ quantifier-free.

Figure 3 Number-theoretic axioms for T, where x, y and t are variables/a term of type N.

A term of the form $\overbrace{\mathsf{s}\cdots\mathsf{s}}^{n}0$ is called a **numeral**, and is more succinctly written just $\underline{n}$.

▶ **Definition 2.2** (System T)**.** *T is the simple type theory over the language given by Figure 1, axiomatised by the formulas and rules from Figure 2 and Figure 3.*

▶ Remark 2.3 (Standard model). We may consider usual Henkin structures for simply typed theories, called **type structures**. One particular structure, the "standard" or "full set-theoretic" model $\mathfrak{N}$, is given by the following interpretation:

- $N^{\mathfrak{N}}$ is $\mathbb{N}$ and $(\sigma \to \tau)^{\mathfrak{N}}$ is the set of functions $\sigma^{\mathfrak{N}} \to \tau^{\mathfrak{N}}$.
- $0^{\mathfrak{N}} := 0 \in \mathbb{N}$ and $\mathsf{s}^{\mathfrak{N}}(n) := n + 1$.
- The other constants of T are interpreted by (higher-order) functionals by taking the equations from Figure 2 as definitions, left-to-right.
- Given $f \in \sigma^{\mathfrak{N}}$ and $g \in (\sigma \to \tau)^{\mathfrak{N}}$, $g \circ^{\mathfrak{N}} f \in \tau^{\mathfrak{N}}$ is defined as $g(f)$.
- For each type σ, we have an *extensional* equality relation $=^{\mathfrak{N}}_{\sigma}$:
  - $=^{\mathfrak{N}}_{N}$ is just equality of natural numbers;
  - for $f, g \in (\sigma \to \tau)^{\mathfrak{N}}$, we have $f =^{\mathfrak{N}}_{\sigma\to\tau} g$ just if $\forall x \in \sigma^{\mathfrak{N}}.f(x) =^{\mathfrak{N}}_{\tau} g(x)$.

It is clear, by reduction to induction at the meta-level, that the interpretations of the constants above are well-defined, and that the axioms of Figure 3 (as well as Figure 2) are satisfied in $\mathfrak{N}$. Thus $\mathfrak{N}$ constitutes a bona fide model of T.

2.2 "Coderivations" and a correctness condition

Coterms are generated *coinductively* from constants and variables under typed application. Formally, we may construe a coterm as a possibly infinite binary tree (of height $\leq \omega$) where each leaf (if any) is labelled by a typed variable or constant and each interior node is labelled by a typed application operation, having type consistent with the types of its children. I.e., an interior node with children of types σ and $\sigma \to \tau$, respectively, must have type τ.

Similarly, a **coderivation**, is a possibly non-wellfounded tree built from the derivation rules of Figure 1. As for (well-founded) derivations and terms, we treat coderivations as coterms in the natural way. We say that a coderivation or coterm is **regular** (or **circular**) if it has finitely many distinct sub-coderivations or sub-coterms, respectively. Note that a regular coderivation or coterm is indeed finitely presentable, e.g. as a finite directed graph, possibly with cycles, or a finite binary tree with "backpointers".

Note that the equational theory induced by Figure 2 forms a Kleene-Herbrand-Gödel style equational specification for regular coterms (cf., e.g., [23]). This allows us to view coterms as *partial recursive functionals* in the standard model $\mathfrak{N}$ of the appropriate type, though a full exposition is beyond the scope of this paper. Instead we will give a more formal (and, indeed, formal*ised*) treatment of "regular" coterms and their computational interpretations in Section 3. We point the reader to the excellent book [27] for further details on models of (partial) (recursive) function(al)s.

Nonetheless, let us temporarily adopt the notation $t^{\mathfrak{N}}$ for the partial functional "computed" by a coterm t in $\mathfrak{N}$, and present some examples, at the same time establishing some foundational results. As before, the reader may safely ignore the colouring of type occurrences in what follows. That will become meaningful later in the section.

▶ **Example 2.4** (Extensional completeness at type 1)**.** For *any* $f : \mathbb{N}^k \to \mathbb{N}$, there is a coderivation $t : N^k \Rightarrow N$ s.t. $t^{\mathfrak{N}} = f$. To demonstrate this, we proceed by induction on k.[3] If $k = 0$ then the numerals clearly suffice. Otherwise, suppose $f : \mathbb{N} \times \mathbb{N}^k \to \mathbb{N}$ and write f_n for the projection $\mathbb{N}^k \to \mathbb{N}$ by $f_n(\vec{x}) = f(n, \vec{x})$. We define the coderivation for f as follows:

$$\cfrac{f_0 \quad \cfrac{f_1 \quad \cfrac{f_2 \quad \cfrac{\vdots}{\vec{N} \Rightarrow N}\,\mathsf{cond}}{N, \vec{N} \Rightarrow N}\,\mathsf{cond}}{N, \vec{N} \Rightarrow N}\,\mathsf{cond}}{N, \vec{N} \Rightarrow N} \tag{1}$$

where the derivations for each f_n are obtained by the inductive hypothesis. It is not difficult to see that the interpretation of this coderivation in the standard model indeed coincides with f.

Notice that, while we have extensional completeness at type 1, we cannot possibly have such a result for higher types by a cardinality argument: there are only continuum many coderivations.

▶ **Example 2.5** (Naïve simulation of primitive recursion)**.** Terms of T may be interpreted as coterms without the rec combinators in a straightforward manner, by the following translation:

$$\cfrac{\vec{\sigma} \Rightarrow \sigma \quad \vec{\sigma}, N, \sigma \Rightarrow \sigma}{\vec{\sigma}, N \Rightarrow \sigma}\,\mathsf{rec} \quad \rightsquigarrow \quad \cfrac{\vec{\sigma} \Rightarrow \sigma \quad \cfrac{\cfrac{\vdots}{\vec{\sigma}, N \Rightarrow \sigma}\,\mathsf{cond}\,\bullet \quad \vec{\sigma}, N, \sigma \Rightarrow \sigma}{\vec{\sigma}, N \Rightarrow \sigma}\,\mathsf{cut}}{\vec{\sigma}, N \Rightarrow \sigma}\,\mathsf{cond}\,\bullet \tag{2}$$

where the occurrences of $\bullet$ indicate roots of identical coderivations.

[3] While we may assume $k = 1$ WLoG by the availability of sequence (de)coding, the current argument is both more direct and avoids the use of cuts (on non-numerals).

Denoting the RHS of (2) above as rec', we can check that the two sides of (2) are equivalent under Figures 2 and 3. Formally, we show $\mathsf{rec}'\, s\, t\, \vec{x}\, y \;=\; \mathsf{rec}\, s\, t\, \vec{x}\, y$ by induction on y:

$$\begin{array}{rcll}
\mathsf{rec}'\, s\, t\, \vec{x}\, 0 & = & \mathsf{cond}\, s\, (\mathsf{cut}\, (\mathsf{rec}'\, s\, t)\, t)\, \vec{x}\, 0 & \text{by definition of } \mathsf{rec}' \text{ above} \\
& = & s\, \vec{x} & \text{by } \mathsf{cond} \text{ axioms} \\
& = & \mathsf{rec}\, s\, t\, \vec{x}\, 0 & \text{by } \mathsf{rec} \text{ axioms} \\
\mathsf{rec}'\, s\, t\, \vec{x}\, \mathsf{s}y & = & \mathsf{cond}\, s\, (\mathsf{cut}\, (\mathsf{rec}'\, s\, t)\, t)\, \vec{x}\, \mathsf{s}y & \text{by definition of } \mathsf{rec}' \text{ above} \\
& = & \mathsf{cut}\, (\mathsf{rec}'\, s\, t)\, t\, \vec{x}\, y & \text{by } \mathsf{cond} \text{ axioms} \\
& = & t\, \vec{x}\, y\, (\mathsf{rec}'\, s\, t\, \vec{x}\, y) & \text{by } \mathsf{cut} \text{ axiom} \\
& = & t\, \vec{x}\, y\, (\mathsf{rec}\, s\, t\, \vec{x}\, y) & \text{by inductive hypothesis} \\
& = & \mathsf{rec}\, s\, t\, \vec{x}\, \mathsf{s}y & \text{by } \mathsf{rec} \text{ axioms}
\end{array}$$

▶ **Example 2.6** (Turing completeness). The set of regular coderivations is Turing-complete,[4] i.e. $\{t^{\mathfrak{N}} \mid t : N^k \Rightarrow N \text{ regular}\}$ includes all partial recursive functions on $\mathbb{N}$. We have already seen in Example 2.5 that we can encode the primitive recursive functions, so it remains to simulate minimisation, i.e. the operation $\mu x(fx = 0)$, for a given function f, returning the least natural number x s.t. $fx = 0$ (if it exists). For this, we observe that $\mu x(fx = 0)$ is equivalent to $H\, 0$ where:

$$H\, x \;=\; \mathsf{cond}\; (f\, x)\; x\; (H\, \mathsf{s}x) \tag{3}$$

Note that H is computed by the following coderivation:

$$\dfrac{\dfrac{\triangledown_f}{N \Rightarrow N} \quad \dfrac{\dfrac{\mathsf{id}\ \overline{N \Rightarrow N} \qquad \mathsf{wk}\ \dfrac{\vdots}{\underline{N}, N \Rightarrow N}}{\underline{N}, N \Rightarrow N}\ \mathsf{cond}}{N \Rightarrow N}\ \mathsf{cut}}{N \Rightarrow N}\ \bullet \qquad \dfrac{\dfrac{\mathsf{s}\ \overline{N \Rightarrow N} \qquad \mathsf{cut}\ \dfrac{\vdots}{N \Rightarrow N}\ \bullet}{N \Rightarrow N}\ \mathsf{cut}}{N \Rightarrow N} \tag{4}$$

It is intuitive here to think of the blue N standing for x, the red N standing for $f(x)$, and the purple N standing for $\mathsf{s}x$. Again, the reader may verify that this coderivation indeed satisfies Equation (3) in the standard model $\mathfrak{N}$. Note that we only used the type N above, and no higher-order types, so Turing-completeness holds already for N-only regular coderivations.

▶ **Definition 2.7** (Immediate ancestry). *Let t be a (co)derivation. A type occurrence σ^1 is an **immediate ancestor**[5] of a type occurrence σ^2 in t if σ^1 and σ^2 appear in the LHSs of a premiss and conclusion, respectively, of a rule instance and have the same colour in the corresponding rule typeset in Figure 1. If σ^1 and σ^2 are elements of an indicated list, say $\vec{\sigma}$, we also require that they are at the same position of the list in the premiss and the conclusion. Note that, if σ^1 is an immediate ancestor of σ^2, they are necessarily occurrences of the same type.*

[4] For a model of program execution, we may simply take the aforementioned Kleene-Herbrand-Gödel model with *equational derivability*, cf. [23]. Note that this coincides with derivability by the axioms thus far presented.

[5] This terminology is standard in proof theory, e.g. as in [10].

The notion of immediate ancestor thus defined, being a binary relation, induces a directed graph whose paths will form the basis of our termination criterion.

▶ **Definition 2.8** (Threads and progress)**.** *A* ***thread*** *is a maximal path in the graph of immediate ancestry. A* ***σ-thread*** *is a thread whose elements are occurrences of the type σ. We say that a N-thread* ***progresses*** *when it is principal for a* cond *step (i.e. it is the indicated blue N in the* cond *rule typeset in Figure 1). A (infinitely)* ***progressing*** *thread is a N-thread that progresses infinitely often (i.e. it is infinitely often the indicated blue N in the* cond *rule typeset in Figure 1.)*

A coderivation is ***progressing*** *if every infinite branch has a progressing thread.*

Note that progressing threads do not necessarily begin at the root of a coderivation, they may begin arbitrarily far into a branch. In this way, the progressing coderivations are closed under all typing rules. Note also that arbitrary coderivations may be progressing, not only the regular ones.

▶ **Example 2.9** (Extensional completeness at type 1, revisited)**.** Recalling Example 2.4, note that the infinite branch marked $\cdots$ in (1) has a progressing thread along the red Ns. Other infinite branches, say through f_0, f_1, etc., will have progressing threads along their infinite branches by an appropriate inductive hypothesis, though these may progress for the first time arbitrarily far from the root of (1).

As previously mentioned, we shall focus our attention in this work on the regular coderivations. Let us take a moment to appreciate some previous (non-)examples of regular coderivations with respect to the progressing criterion.

▶ **Example 2.10** (Primitive recursion and Turing-completeness, revisited)**.** Recalling Example 2.5, notice that the RHS of (2) is a progressing coderivation: there is precisely one infinite branch (that loops on $\bullet$) and it has a progressing thread on the blue N indicated there.

Now recalling Example 2.6, notice that the coderivation given for H in (4) is *not* progressing: the only infinite branch loops on $\bullet$ and immediate ancestry, as indicated by the colouring, admits no thread along the $\bullet$-loop.

One of the most appealing features of the progressing criterion is that it is decidable (for regular coderivations) by a well-known reduction to unviersality of Büchi automata (see, e.g., [17] for an exposition for a similar circular system). On the semantic side, we duly have:

▶ **Proposition 2.11.** *If $t : \vec{\sigma} \Rightarrow \tau$ is a progressing coderivation, then $t^{\mathfrak{N}}$ is a well-defined total functional in $(\vec{\sigma} \to \tau)^{\mathfrak{N}}$.*

Proof sketch. First, observe that each constant (i.e. rule instance) computes a total functional of corresponding type. Thus, contrapositively, if $t^{\mathfrak{N}}$ is non-total then so is one of its immediate sub-coderivations. Continuing this reasoning yields an infinite branch $(t_i : \vec{\sigma}_i \Rightarrow \tau_i)_i$ of non-total coderivations. Now, by the progressing criterion, there must be a progressing thread $(N_i)_{i \geq k}$ along this branch. Assigning to each occurrence N_i the least natural number n_i on which t_i is non-total yields a monotone non-increasing sequence $(n_i)_{i \geq k}$ that does not converge (by definition of progressing thread), giving the required contradiction. ◀

2.3 Some fragments and program extraction

Let us write T^- for the restriction of T to the language without the rec constants from Figure 1, and so also without the rec axioms from Figure 2.

▶ **Definition 2.12** (Circular version of T)**.** *The language of CT contains every regular progressing coderivation of T^- as a symbol. We identify "terms" of this language (i.e. finite applications of regular progressing coderivations, constants and variables) with coterms in the obvious way, and call them* ***regular progressing coterms****. CT itself is axiomatised by the schemata from Figures 2 and 3, now interpreting the metavariables s, t etc. there as ranging over (regular progressing) coterms.*

The aim of this work is to compare fragments of CT and fragments of T delineated by type level. Recall that the **level** of a type σ, written $\mathsf{lev}(\sigma)$ is given by: $\mathsf{lev}(N) := 0$ and $\mathsf{lev}(\sigma \to \tau) := \max(1 + \mathsf{lev}(\sigma), \mathsf{lev}(\tau))$.

▶ **Definition 2.13** (Type level restricted fragments of T and CT)**.** *T_n is the restriction of T to the language containing only recursors rec_σ where $\mathsf{lev}(\sigma) \leq n$.*

CT_n is the restriction of CT to the language containing only coderivations where all types occurring have level $\leq n$. CT_n still has symbols for each constant of T^-.

Note that this definition of CT_n is quite natural, since it is known that T_n derivations (of level $n+1$ functionals) can be put into an analogous form (see, e.g., [12]). For instance, the coderivation in Equation (4) has level 0 (though it is not an element of CT_0 since it is not progressing). Note that CT itself is just the union of all CT_n, since regular coderivations have only finitely many type occurrences and so exhibit a maximum type level.

The significance of the fragments T_n, in terms of quantifier-restricted fragments of arithmetic, was investigated in the seminal work of Parsons [29]. Let us first recall such fragments in a two-sorted framework.

RCA_0 is a second-order[6] theory in the language of arithmetic (i.e. with symbols $0, \mathsf{s}, +, \times, <$). It is axiomatised by an appropriate extension of Robinson's Q to the second-order setting, along with comprehension for (provably) Δ^0_1 predicates and induction for Σ^0_1 formulas. A comprehensive presentation of RCA_0 and related theories can be found in, e.g., [34, 22].

Writing $I\Sigma^0_n$ for the induction scheme for Σ^0_n formulas we have:

▶ **Proposition 2.14** ([29])**.** *If $\mathsf{RCA}_0 + I\Sigma^0_{n+1} \vdash \forall \vec{x} \exists y A(\vec{x}, y)$, where A is Δ^0_0, then there is a T_n term t with $T_n \vdash A(\vec{x}, t\,\vec{x})$.*[7]

Since we use it later, let us note that $I\Sigma^0_n$ is equivalent, over a weak base theory (certainly RCA_0), to induction on Boolean combinations of Σ^0_n formulas, cf., e.g., [20]. The theory ACA_0 is obtained from RCA_0 by adding comprehension for arithmetical predicates, and is equivalent, over arithmetical theorems, to the extension of RCA_0 by arithmetical induction.

Let us also mention a nontrivial result from previous work that we shall make use of:

▶ **Proposition 2.15** ([11])**.** *For any regular progressing coderivation t, RCA_0 proves that t is progressing.*

Since progressiveness is, a priori, a Π^1_2 property, the above result is not at all immediate and relies on a formalisation of Büchi automaton theory that is implicit in [25]. Note that this result is "non-uniform" , in that the quantification over coderivations t takes place at the meta-level. As noted in [11], the above result cannot be strengthened to a uniform one unless RCA_0 (and so PRA) is inconsistent, by a reduction to Gödel-incompleteness.

---

[6] As for simple type theories, all references to "second" or "higher" order are purely due to convention. Strictly speaking, these are multi-sorted first-order theories.

[7] We assume here some standard encoding of Δ_0 formulas into quantifier-free formulas of T_0. Alternatively we could admit bounded quantifiers into the language of T, on which induction is allowed, without affecting expressivity. We shall gloss over this technicality here.

3 Confluence and models of T

We cannot formalise the standard model $\mathfrak{N}$ in arithmetic for cardinality reasons, however there are natural models of partial recursive functionals that can be formalised, namely the *hereditarily recursive* operations of finite type (see, e.g., [27]). We shall recast this type structure using regular coterms, in light of Example 2.6 and Example 2.10.

3.1 Reduction sequences and their logical complexity

▶ **Definition 3.1.** *The **reduction** relation $\rightsquigarrow$ on coterms is defined by orienting all the equations in Figure 2 left-to-right and taking closure under substitution and contexts. We write $\approx$ for the reflexive, symmetric, transitive closure of $\rightsquigarrow$, and freely use standard rewriting theoretic terminology and notations for these relations.*

Since coterms are potentially infinite, equality for them is a Π^0_1 predicate. Thus, for the sake of simplicity, we shall henceforth deal with only regular coterms, which are finite so may be coded by natural numbers. Representing regular coterms as finite directed graphs, note that equality now reduces to checking bisimilarity, which is provably recursive in RCA_0.

In fact, throughout this section, we will only deal with coterms that are finite applications of regular coderivations, variables and constants ("FARs" for short). We better show that these are at least closed under reduction. To this end, let us write, for $v \in \{0,1\}^*$, t_v for the sub-coterm of t rooted at position v. We have:

▶ **Proposition 3.2** (RCA_0). *If $s \rightsquigarrow t$ then t is finitely composed of sub-coterms of s:*

$$\exists \text{ a finite term } r(x_1,\dots,x_n).\, \exists \langle v_1,\dots,v_n\rangle.\, t = r(s_{v_1},\dots,s_{v_n}) \tag{5}$$

We can take $s_{v_1},\dots,s_{v_n}$ to include the coderivations indicated in the contractum of a reduction, as well as the "comb" of the redex of the reduction in s, i.e. the siblings of all the nodes in the path leading to the redex. $r(\vec{x})$ is now the finite term induced by the contracta and this comb.

Naturally, this property also holds for $\rightsquigarrow^*$ and $\approx$, by Σ^0_1-induction. As a consequence:

▶ **Corollary 3.3** (RCA_0). *If s is a FAR and $s \rightsquigarrow t$ or $s \rightsquigarrow^* t$ or $s \approx t$, then t is a FAR.*

Note, in particular, that $\rightsquigarrow$, $\rightsquigarrow^*$ and $\approx$, restricted to FARs, are Σ^0_1-relations.

3.2 Confluence of reduction

In order to obtain basic metamathematical properties of the coterm models we later consider, we need to know that our model of computation is *deterministic*, so that coterms have unique interpretations. There are various ways to prove this in arithmetic, but we will approach it in terms of *confluence* in rewriting theory.

Throughout this subsection we continue to deal only with FARs, i.e. coterms that are finite applications of regular coderivations, variables and constants. The main goal of this subsection is to prove the following:

▶ **Theorem 3.4** (Church-Rosser, RCA_0). *Let $t : \sigma$ be a FAR. If $t_0 \,{}^*\!\!\leftsquigarrow t \rightsquigarrow^* t_1$ then there is $t' : \sigma$ such that $t_0 \rightsquigarrow^* t' \,{}^*\!\!\leftsquigarrow t_1$.*

To some extent, we follow a standard approach to proving this result. However, since coterms are infinite (and, moreover, non-wellfounded), we must carry out our argument without appeal to induction on *term structure*, as is usual in presentations of arguments due to Tait and Martin-Löf (cf., e.g., [21]). Instead, we perform an argument by induction on reduction *length*, as in, e.g., [30].

▶ **Definition 3.5** (Parallel reduction). *We define the relation $\rhd$ on FARs as follows:*

1. *$t \rhd t$ for any FAR t.*[8]
2. *For a reduction step $\mathsf{r}\,\vec{t} \rightsquigarrow r(\vec{t})$, if each $t_i \rhd t_i'$ then we have $\mathsf{r}\,\vec{t} \rhd r(\vec{t'})$.*
3. *For a reduction step $\mathsf{r}\,\vec{t}\,\mathsf{s}s \rightsquigarrow r(\vec{t}, s)$ (i.e. a* rec *or* cond *successor step), if each $t_i \rhd t_i'$ and $s \rhd s'$ then we have $\mathsf{r}\,\vec{t}\,\mathsf{s}s \rhd r(\vec{t'}, s')$.*
4. *If $s \rhd s'$ and $t \rhd t'$ then $s\,t \rhd s'\,t'$.*

▶ **Proposition 3.6** (RCA_0). *$s \rightsquigarrow t \implies s \rhd t$ and $s \rhd t \implies s \rightsquigarrow^* t$.*

The proof of this result is not difficult, but before giving an argument let us point out a particular consequence that we will need, obtained by Σ^0_1-induction on the length of reduction sequences:

▶ **Corollary 3.7** (RCA_0). *$s \rightsquigarrow^* t \iff s \rhd^* t$*

Even though it is not necessary to prove the proposition above, we shall first prove the following useful lemma since we will use it later:

▶ **Lemma 3.8** (Substitution, RCA_0). *Suppose $t \rhd t'$. If $s \rhd s'$ then $s[t/x] \rhd s'[t'/x]$, for a variable x of the same type as t and t'.*

Writing, say, $d : s \rightsquigarrow^* t$ for the (provably) Δ^0_1 predicate "d is a $\rightsquigarrow$-derivation from s to t", the above result is shown by proving

$$d : s \rhd s' \implies s[t/x] \rhd s'[t'/x]$$

by Σ^0_1-induction on the structure of the derivation $d : s \rhd s'$. We crucially use the fact that we are dealing with FARs for the base case when $s' = s$, using a subinduction on the maximum depth of an x-occurrence in s.

Notice that Proposition 3.6 now follows immediately, by simply instantiating the Lemma above with $s = s'$ to deduce context-closure of $\rhd$.

▶ **Lemma 3.9** (Diamond property of $\rhd$, RCA_0). *Suppose $t_0 \lhd s \rhd t_1$. Then there is some u with $t_0 \rhd u \lhd t_1$.*

Before giving the proof, it will be useful to have the following intermediate result, which follows by Σ^0_1-induction:

▶ **Proposition 3.10** (RCA_0). *Suppose $d : \mathsf{r}\,\vec{s} \rhd t$, and there is no redex in $\mathsf{r}\,\vec{s}$ involving r. There are some $\vec{t}$ s.t. $t = \mathsf{r}\,\vec{t}$ and, for each i, some $d_i : s_i \rhd t_i$ for some $d_i < d$.*

The diamond property, Lemma 3.9, now follows by proving

$$\exists s'.\,((d_0 : s \rhd t_0 \text{ and } d_1 : s \rhd t_1) \implies (t_0 \rhd s' \text{ and } t_1 \rhd s'))$$

by Σ^0_1-induction on $\min(|d_0|, |d_1|)$. We use Lemma 3.8 for the case when both d_0 and d_1 end by clause (2), and we use Proposition 3.10 when d_0 ends by clause (2) and d_1 ends by clause (4) or vice-versa.

[8] Note that we really do seem to require $t \rhd t$ for arbitrary FARs t, not just variables and constants, since we cannot finitely derive the former from the latter.

▶ **Proposition 3.11** (Weighted CR for $\rhd$, RCA_0). *If $t_0 \lhd^m t \rhd^n t_1$ then there is some t' with $t_0 \rhd^n t' \lhd^m t_1$.*

The argument for this follows by proving

$$(d_0 : t \rhd^m t_0 \text{ and } d_1 : t \rhd^n t_1) \implies \exists t'(t_0 \rhd^n t' \text{ and } d_1' : t_1 \rhd^m t')$$

by Σ^0_1-induction on $m = |d_0|$. The following corollary is immediate:

▶ **Corollary 3.12** (CR for $\rhd$, RCA_0). *If $t_0 \lhd^* t \rhd^* t_1$ then there is t' s.t. $t_0 \rhd^* t' \lhd^* t_1$.*

We may finally conclude the main result of this subsection:

Proof of Theorem 3.4. Suppose $t_0 \leftsquigarrow^* s \rightsquigarrow^* t_1$. Then, by Corollary 3.7 we have $t_0 \lhd^* s \rhd^* t_1$. By Corollary 3.12 above, we have some s' with $t_0 \rhd^* s' \lhd^* t_1$, whence $t_0 \rightsquigarrow^* s' \leftsquigarrow t_1$ by Corollary 3.7 again. ◀

3.3 Hereditarily total coterms under conversion

We are now ready to present a type structure that will allow us to obtain an interpretation of CT_n within T_{n+1}. The structure that we present in this subsection is essentially the *hereditrarily recursive operations* of finite type, but where we adopt FARs under conversion as the underlying model of computation, cf. Example 2.6 and Example 2.10.

▶ **Definition 3.13.** *We define the following sets of FARs:*
- $\mathsf{HR}_N := \{t : N \mid \exists n \in \mathbb{N}.\ t \approx \underline{n}\}$
- $\mathsf{HR}_{\sigma\to\tau} := \{t : \sigma \to \tau \mid \forall s \in \mathsf{HR}_\sigma.\ t\,s\ \in \mathsf{HR}_\tau\}$

We write HR_n for the union of all HR_σ with $\mathsf{lev}(\sigma) \leq n$.

Note that it is immediate from the definition that each HR_σ contains only closed FARs of type σ. Notice that, by the confluence result of the previous subsection, Theorem 3.4, if $t \approx \underline{n}$ then $n \in \mathbb{N}$ is unique and in fact $t \rightsquigarrow^* \underline{n}$ (provably in RCA_0). In this way we can view every element of HR_N as computing a unique natural number by means of reduction.

▶ **Fact 3.14.** *HR_N is Σ^0_1, and if $\mathsf{lev}(\sigma) = n > 0$ then HR_σ is Π^0_{n+1}.*

This is obtained by a (meta-level) induction on the type σ. The same induction also yields:

▶ **Proposition 3.15** (Closure properties of HR). *Fix types σ and τ. RCA_0 proves the following:*
1. *If $s \in \mathsf{HR}_\sigma$ and $t \in \mathsf{HR}_{\sigma\to\tau}$ then $ts \in \mathsf{HR}_\tau$. (HR closed under application)*
2. *If $t \in \mathsf{HR}_\tau$ and $t \approx t'$ then $t' \in \mathsf{HR}_\tau$. (HR closed under conversion)*

Note that provability within RCA_0 above is *non-uniform* in σ and τ, i.e. RCA_0 proves the statements for each particular σ and τ. These properties justify defining the following type structure:

▶ **Definition 3.16** (HR structure). *We write HR for the type structure defined as follows:*
- *σ^{HR} is HR_σ.*
- *r^{HR} is just r for each constant r.*
- *$t \circ^{\mathsf{HR}} s$ is just ts.*
- *$=^{\mathsf{HR}}_\sigma$ is $\approx_\sigma$.*

Ultimately we will show that this structure constitutes a model of CT. For this the following lemma will be key:

▶ **Lemma 3.17** (Induction for HR, RCA_0). *Suppose $r(x)$ and $s(x)$ are FARs. If $r(0) \approx s(0)$ and $\forall t \in \mathsf{HR}_N.(r(t) \approx s(t) \implies r(\mathsf{s}t) \approx s(\mathsf{s}t))$, then $\forall t \in \mathsf{HR}_N.r(t) \approx s(t)$.*

This result is essentially "forced" by the definition of HR_N, reducing induction in HR to induction in RCA_0. We also rely on the Leibniz property of equality in the structure (i.e. if $s \approx t$ and $\varphi(s)$ then $\varphi(t)$), which is facilitated by the symmetry and transitivity of $\approx$.

Note that the axioms governing the constants are immediate given that our reduction relation is obtained from them. The remaining number-theoretic axioms follow from confluence (for $\neg \mathsf{s}0 \approx 0$, by uniqueness of normal forms) and the fact that no reduction rule has s at the head (for $\mathsf{s}s \approx \mathsf{s}t$ implies $s \approx t$, requiring a Σ^0_1-induction).

Thus to conclude that HR actually constitutes a model of T (or CT) it remains to show that it interprets each term t of T (or coterm of CT), i.e. that indeed $t \in \mathsf{HR}$. For T, this follows from Tait's seminal normalisation result [36]:

▶ **Proposition 3.18.** HR *is a model of* T.

In fact this result can be formalised non-uniformly in the following sense: for each term t of type τ with $\mathsf{lev}(\sigma) \leq n$, we have $\mathsf{RCA}_0 + I\Sigma^0_{n+1} \vdash \mathsf{HR}_\tau(t)$. We will see a similar situation for membership of CT_n coderivations in HR_{n+1} later, but with the quantifier complexity of induction increased by 1.

4 Interpretation of CT into T

In this section we show that the type structure HR introduced in the previous section indeed constitutes a model of CT. In fact, we will formalise the membership of CT_n coderivations in HR_{n+1} within the theory $\mathsf{RCA}_0 + I\Sigma^0_{n+2}$ (non-uniformly), whence we obtain explicit equivalent terms of T_{n+1} by program extraction. Throughout this section we continue to work only with regular coterms that are finite applications of coderivations, variables and constants (i.e. FARs).

4.1 Canonical branches of non-total coterms

In this section we give a formalised proof of the totality of CT-coterms. Our approach will be to import a suitable version of the proof of Proposition 2.11 but relativise all the quantifiers, there in the standard model, to their respective domains in HR.

First let us note that HR is closed under the typing rules of CT:

▶ **Observation 4.1.** *Consider a rule instance* $\mathsf{r}\,\dfrac{\vec{\sigma}_0 \Rightarrow \tau_0 \quad \cdots \quad \vec{\sigma}_k \Rightarrow \tau_k}{\vec{\sigma} \Rightarrow \tau}$ *for some* $k < 2$. *If* $t_i \in \mathsf{HR}_{\vec{\sigma}_i \to \tau_i}$ *for* $i < k$ *then* $\mathsf{r}\ t_0\ \cdots\ t_k \in \mathsf{HR}_{\vec{\sigma} \to \tau}$.

This follows by simple inspection of the rules of CT. By contraposition, any coderivation $\notin \mathsf{HR}$ must induce an infinite branch of coderivations $\notin \mathsf{HR}$, similarly to the proof of Proposition 2.11. The next definition formalises a canonical such branch, as induced by an input on which a coderivation is non-hereditarily-total. We shall present just the definition of the branch first, and then argue that it is well-defined, for each explicit CT_n coderivation, in $\mathsf{RCA}_0 + I\Sigma^0_{n+2}$.

▶ **Definition 4.2** (Branch generated by a non-total input)**.** *Let* $t_0 : \vec{\sigma}_0 \Rightarrow \tau_0$ *be a coderivation and let* $\vec{s}_0 \in \mathsf{HR}_{\vec{\sigma}_0}$ *s.t.* $t_0\,\vec{s}_0 \notin \mathsf{HR}_\tau$. *We define the branch* $(t_i : \vec{\sigma}_i \Rightarrow \tau_i)_{i \geq 0}$ *and inputs* $\vec{s}_i \in \mathsf{HR}_{\vec{\sigma}_i}$, generated *by* t_0 *and* $\vec{s}_0$ *below. Each rule instance is as typeset in Figure 1, with immediate sub-coderivations* t *and* t' *respectively. Furthermore, we preserve the invariant* $t_i\,\vec{s}_i \notin \mathsf{HR}_{\tau_i}$ *throughout the definition.*

1. *(t_i cannot be an initial sequent).*
2. *Suppose t_i ends with* wk *and $\vec{s}_i = (\vec{s}, s)$. Then $t_{i+1} := t$ and $\vec{s}_{i+1} := \vec{s}$.*
3. *Suppose t_i ends with* ex *and $\vec{s}_i = (\vec{r}, r, s, \vec{s})$. Then $t_{i+1} := t$ and $\vec{s}_{i+1} := (\vec{r}, s, r, \vec{s})$.*
4. *Suppose t_i ends with* cntr *and $\vec{s}_i = (\vec{s}, s)$. Then $t_{i+1} := t$ and $\vec{s}_{i+1} := (\vec{s}, s, s)$.*
5. *Suppose t_i ends with* cut *and $\vec{s}_i = \vec{s}$. Then if $t\,\vec{s} \in \mathsf{HR}_\sigma$ then $t_{i+1} := t'$ and $\vec{s}_{i+1} := (\vec{s}, t\,\vec{s})$. Otherwise, $t_{i+1} := t$ and $\vec{s}_{i+1} := \vec{s}$.*
6. *Suppose t_i ends with* L *and $\vec{s}_i = (\vec{s}, s)$. If $t\,\vec{s} \in \mathsf{HR}_\rho$ then $t_{i+1} := t'$ and $\vec{s}_{i+1} := (\vec{s}, s\,(t\,\vec{s}))$. Otherwise $t_{i+1} := t$ and $\vec{s}_{i+1} := \vec{s}$.*
7. *Suppose t_i ends with* R *and $\vec{s}_i = \vec{s}$. Let s be the least[9] element of HR_σ such that $t\,\vec{s}\,s \notin \mathsf{HR}_\tau$. We set $t_{i+1} := t$ and $\vec{s}_{i+1} := (\vec{s}, s)$.*
8. *Suppose t_i ends with* cond *and $\vec{s}_i = (\vec{s}, r)$. If $r \approx 0$ then $t_{i+1} := t$ and $\vec{s}_{i+1} := \vec{s}$. Otherwise, if $r \approx \mathsf{s}\underline{n}$, then $t_{i+1} := t'$ and $\vec{s}_{i+1} := (\vec{s}, \underline{n})$.*

The main result of this subsection is:

▶ **Proposition 4.3.** *Let $t_0 : \vec{\sigma}_0 \Rightarrow \tau_0$ be a fixed coderivation in which all types occurring have level $\leq n$. $\mathsf{RCA}_0 + I\Sigma^0_{n+2}$ proves the following: if $\vec{s}_0 \in \mathsf{HR}_{\vec{\sigma}_0}$ s.t. $t_0\,\vec{s}_0 \notin \mathsf{HR}_{\tau_0}$ then the branch $(t_i)_i$ and inputs $(\vec{s}_i)_i$ generated by t_0 and s_0 are Δ^0_{n+2}-well-defined.*

Most of the cases follow by the inductive hypothesis and the closure of HR under $\approx$. Crucially, for the R case, we must use the Σ^0_{n+1}-minimisation principle, a consequence of $I\Sigma^0_{n+1}$ cf. [20], to find the "least" FAR s satisfying a Σ^0_{n+1} property. We also use confluence to ensure that the cond-case is well-defined.

4.2 Progressing coterms are hereditarily total

We are now ready to show that CT-coterms are hereditarily total, i.e. that they belong to HR. Now that we have formalised the infinite "non-total" branches of the proof of Proposition 2.11, relativised to the type structure HR, we continue to formalise the remainder of the argument. First, again by confluence, we have:

▶ **Lemma 4.4** (RCA_0). *Let $t_0 : \vec{\sigma}_0 \Rightarrow \tau_0$ and $\vec{s}_0 \in \mathsf{HR}_{\vec{\sigma}_0}$ be a coderivation and inputs s.t. $t_0\,\vec{s}_0 \notin \mathsf{HR}_{\tau_0}$. Furthermore let $(t_i : \vec{\sigma}_i \Rightarrow \tau_i)_i$ and $\vec{s}_i \in \mathsf{HR}_{\vec{\sigma}_i}$ be a branch and inputs generated by t_0 and $\vec{s}_0$, satisfying Definition 4.2.*

Suppose some N-occurrence $N^{i+1} \in \vec{\sigma}_{i+1}$ is an immediate ancestor of some N-occurrence $N^i \in \vec{\sigma}_i$. Write $s_i \in \vec{s}_i$ for the coterm in HR_N corresponding to N^i, and similarly $s_{i+1} \in \vec{s}_{i+1}$ for the coterm $s_{i+1} \in \mathsf{HR}_N$ corresponding to N^{i+1}. If $s_i \approx \underline{n}_i$ and $s_{i+1} \approx \underline{n}_{i+1}$, for $n_i, n_{i+1} \in \mathbb{N}$, then:

1. *$n_i \geq n_{i+1}$.*
2. *If N^i is principal for a* cond *step, then $n_i > n_{i+1}$.*

In order to complete our formalisation of the totality argument, we actually have to use an "arithmetical approximation" of thread progression that nonetheless suffices for our purposes, similarly to [11]. The reason for this is that, even though non-total branches are well-defined by Proposition 4.3, we do not a priori have access to them as *sets* in extensions of RCA_0 by induction principles, and so the lack of progressing threads along them does not directly contradict the fact that a coderivation is progressing.[10]

[9] Recall that, strictly speaking, we assume all our objects are coded by natural numbers in the ambient theory (here fragments of second-order arithmetic). Thus we may always find a "least" object satisfying a property when one exists. Naturally this will correspond to a form of induction in the proof of well-definedness.

[10] Notice that this is not an issue in the presence of arithmetical comprehension, i.e. in ACA_0, but in that case logical complexity of defined sets is not a stable notion: all of arithmetical comprehension reduces to Π^0_1-comprehension.

▶ **Proposition 4.5** (RCA_0). *Suppose t_i and $\vec{s}_i$ are as in Lemma 4.4. Any N-thread along $(t_i)_i$ is not progressing. Moreover, $\forall k.\exists m.$ any N-thread from t_k progresses $\leq m$ times.*

The main result of this subsection is:

▶ **Theorem 4.6.** *Let $t : \vec{\sigma} \Rightarrow \tau$ be a CT_n-coderivation. Then $\mathsf{RCA}_0 + I\Sigma^0_{n+2} \vdash t \in \mathsf{HR}_{\vec{\sigma}\to\tau}$.*

As well as using Proposition 4.5, this result relies crucially on the fact that we prove that CT-coderivations progress in RCA_0, Proposition 2.15 (itself from [11], allowing us to "substitute" the Δ^0_{n+2}-definition of a non-hereditarily-total branch from Definition 4.2 to obtain an argument using $I\Sigma^0_{n+2}$ overall.

▶ **Corollary 4.7.** HR *is a model of CT.*

4.3 Interpretation of CT_n into T_{n+1}

We may now realise our model-theoretic results as bona fide interpretations of fragments of CT into fragments of T. As a word of warning, coterms of CT in this section, when operating inside T, should formally be understood by their Gödel codes, i.e. in this section T is "one meta-level higher" than CT. Until now we have been formalising the metatheory of CT within second-order arithmetic, and so arithmetising its syntax as natural numbers. Since we will here invoke program extraction from these fragments of arithmetic to fragments of T to interpret CT, the same coding carries over. At the risk of confusion, we shall suppress this formality henceforth.

▶ **Theorem 4.8.** *If $CT_n \vdash s = t$ then $T_{n+1} \vdash s \approx t$.*

The main idea here is that our formalisation of the HR model within arithmetic allows us to prove the following *reflection principle* in $\mathsf{RCA}_0 + I\Sigma^0_2$:

$$\forall P\,(\text{if } \text{“}P \text{ is a } CT_n \text{ proof of } s = t\text{”} \text{ then } \exists d : s \approx t)$$

Since this statement is Π^0_2, we may apply program extraction, Proposition 2.14, to indeed witness the required derivation d within T_{n+1}, as required.

▶ **Corollary 4.9.** *If $t : \vec{N} \Rightarrow N$ is a progressing coterm of CT_n, then there is a T_{n+1}-term $t : \vec{N} \to N$ such that $t'^{\mathfrak{N}} = t^{\mathfrak{N}}$.*

5 Further results

In this section we shall give some further rewriting-theoretic results related to the system CT we have presented.

5.1 Continuity at type 2

It is well-known that the type 2 functionals of T are *continuous*, in the sense that any type 1 function input is only queried a finite number of times, e.g. [38, 32, 39]. For the case of CT, we may actually formalise a variation of the classical argument of [38] within second-order arithmetic, extending the simulation of CT coterms within T to type 2 functionals. For the sake of brevity, we shall not refine our exposition by type level in this subsection.

Let us fix a CT coderivation $t : \vec{\sigma} \Rightarrow N$ s.t. each $\sigma_i = N_1 \to \cdots \to N_{k_i} \to N$, and let us henceforth work in ACA_0, distinguishing second-order variables $f_i : \mathbb{N}^{k_i} \to \mathbb{N}$, intuitively representing the inputs for t. Within CT, introduce new (uninterpreted) constant symbols $\underline{f}_i : N_1 \to \cdots \to N_{k_i} \to N$ for each σ_i, and new reduction steps:

$$\underline{f}_i\, \underline{n}_1 \ \dots\ \underline{n}_{k_i} \rightsquigarrow \underline{f_i(n_1, \dots, n_{k_i})} \tag{6}$$

Notice that reduction is now still semi-recursive in the oracles $\vec{f}$, i.e. $\rightsquigarrow, \rightsquigarrow^*, \approx$ are now $\Sigma^0_1(\vec{f})$. To save the effort of reproving our confluence results from Section 3 with these new oracle symbols, we shall simply henceforth assume a suitable consistency principle:

$$\mathrm{UNF}_N \quad : \quad \forall m, n.\, (\underline{m} \approx \underline{n} \supset m = n)$$

Note that, since this is a true Π^0_1 statement (by meta-level reasoning), it carries no computational content and adding it to ACA_0 still admits extraction into T (see, e.g., [24]).[11] From here, we define $\mathsf{HR}^{\vec{f}}_\sigma$ just as HR_σ, but allowing coterms to include the symbols $\underline{\vec{f}}$. Since each HR_σ is arithmetical in $\rightsquigarrow$, we have that each $\mathsf{HR}^{\vec{f}}_\sigma$ is arithmetical in our extended reduction relation, so with free second-order variables $\vec{f}$. Note in particular that we have that each $\underline{f_i} \in \mathsf{HR}^{\vec{f}}_{\sigma_i}$, thanks to (6) above. By adapting our approach from Section 4, we may show:

▶ **Theorem 5.1** ($\mathsf{ACA}_0 + \mathrm{UNF}_N$). $\forall \vec{f}.\, t\, \underline{\vec{f}} \in \mathsf{HR}^{\vec{f}}_N$

Expanding out this result we have that $\mathsf{ACA}_0 + \mathrm{UNF}_N \vdash \forall \vec{f}. \exists n.\, t\, \vec{f} \approx \underline{n}$. Note that this yields the required syntactic continuity property: since any $\approx$-sequence is finite, we may compute $t(\vec{f})$ by querying each f_i only finitely many times. From here, by applying a relativised version of program extraction (see, e.g., [24]), we obtain a strengthening of our simulation of CT-coterms by T terms to type 2 (stated without refinement to type level):

▶ **Corollary 5.2.** *If t is a level 2 coterm of CT, then there is a T term t' s.t. $t'^{\mathfrak{N}} = t^{\mathfrak{N}}$.*

5.2 A "term model" à la Tait and strong normalisation

It is an immediate consequence of our results that CT-coterms are *weakly normalising*. Namely, by an induction on type (using confluence for the base case, at type N), we may show that each $t \in \mathsf{HR}$ is weakly normalising. Thus, by Theorem 4.6, we have:

▶ **Proposition 5.3.** *Each closed CT coterm is weakly normalising. Moreover, any CT_n coterm is provably weakly normalising inside $\mathsf{RCA}_0 + I\Sigma^0_{n+2}$.*

In this section we will go further and show that CT-coterms are actually *strongly normalising*, just like T-terms. For the sake of brevity, we will not formalise our exposition within arithmetic. We will define a minimal "coterm model" in a similar way to Tait's term models of sytem T [36]. This is complementary to our development of HR: while that structure was an "over-approximation" of the language of CT, the structure we are about to define is an "under-approximation" , by virtue of its definition. Naturally, the point is to show that the approximation is, in fact, tight.

▶ **Definition 5.4** (Convertibility). *We define the following sets of closed CT-coterms:*

- $\mathcal{C}_N := \{t : N \mid t \text{ is strongly normalising}\}$.
- $\mathcal{C}_{\sigma\to\tau} := \{t : \sigma \to \tau \mid \forall s \in \mathcal{C}_\sigma.\, ts \in \mathcal{C}_\tau\}$.

By an induction on type, we establish suitable versions of Proposition 3.15 and the normalisation property for $\mathcal{C}$:

▶ **Proposition 5.5.** *We have the following:*

1. *If $t \in \mathcal{C}_{\sigma\to\tau}$ and $s \in \mathcal{C}_\sigma$ then $ts \in \mathcal{C}_\tau$. ($\mathcal{C}$ closed under application)*
2. *If $t \in \mathcal{C}_\tau$ and $t \rightsquigarrow t'$ then $t' \in \mathcal{C}_\tau$. ($\mathcal{C}$ closed under reduction)*
3. *If $t \in \mathcal{C}_\tau$ then t is strongly normalising. ($\mathcal{C} \subseteq \mathrm{SN}$)*

[11] The drawback of this approach is that it does not yield any bona fide interpretation of CT into T, which is why we chose to formalise a confluence argument for our main interpretation result.

Closure of $\rightsquigarrow$ under contexts is required for 2 and 3. Note that the strong normalisation condition for $\mathcal{C}_N$ is crucial to justify closure under reduction, (2), at base type N. In contrast, for HR_N we only asked for *conversion* to a numeral, and so the analogous property of closure under conversion was a consequence of symmetry.

Let us call a coterm t **neutral** if, for any s, any redex of ts is either entirely in t or entirely in s. We also have the following expected characterisation of convertibility by induction on type:

▶ **Lemma 5.6** (Convertibility lemma). *Let t be neutral. If $\forall t' \leftsquigarrow t.\ t' \in \mathcal{C}_\tau$, then $t \in \mathcal{C}_\tau$.*

As for classical proofs of strong normalisation for T, we must also make use of a sub-induction on the size of the complete reduction trees of elements of $\mathcal{C}$; recall that they are strongly normalising, by Proposition 5.5, and so have finite reduction trees by König's lemma,[12] since there are always only finitely many redexes.

Now we can go on to define a non-converting branch, just like we did for the standard model $\mathfrak{N}$ in Proposition 2.11 (non-total branch), and for HR in Definition 4.2 (non-hereditarily-total branch). As in the latter case, we need to prove well-definedness of such a branch, cf. Observation 4.1 and Proposition 4.3.

▶ **Proposition 5.7** (Preservation of convertibility). *Let $\vec{r} \in \mathcal{C}_{\vec{\rho}}$ and $\vec{s} \in \mathcal{C}_{\vec{\sigma}}$. We have:*[13]

- *If $s \in \mathcal{C}_\sigma$ then $\mathsf{id}\, s \in \mathcal{C}_\sigma$.*
- *If $r \in \mathcal{C}_\rho, s \in \mathcal{C}_\sigma$ and $t\,\vec{r}\,s\,r\,\vec{s} \in \mathcal{C}_\tau$ then $\mathsf{ex}\, t\,\vec{r}\,r\,s\,\vec{s} \in \mathcal{C}_\tau$.*
- *If $s \in \mathcal{C}_\sigma$ and $t\,\vec{s} \in \mathcal{C}_\tau$ then $\mathsf{wk}\, t\,\vec{s}\,s \in \mathcal{C}_\tau$.*
- *If $s \in \mathcal{C}_\sigma$ and $t\,\vec{s}\,s\,s \in \mathcal{C}_\tau$ then $\mathsf{cntr}\, t\,\vec{s}\,s \in \mathcal{C}_\tau$.*
- *If $t_0\,\vec{s} \in \mathcal{C}_\sigma$ and $\forall s \in \mathcal{C}_\sigma.\ t_1\,\vec{s}\,s \in \mathcal{C}_\tau$ then $\mathsf{cut}\, t_0\, t_1\,\vec{s} \in \mathcal{C}_\tau$.*
- *If $r \in \mathcal{C}_{\rho\to\sigma}$ and $t_0\,\vec{s} \in \mathcal{C}_\rho$ and $\forall s \in \mathcal{C}_\sigma.\ t_1\,\vec{s}\,s \in \mathcal{C}_\tau$ then $\mathsf{L}\, t_0\, t_1\,\vec{s}\,r \in \mathcal{C}_\tau$.*
- *If $\forall s \in \mathcal{C}_\sigma.\ t\,\vec{s}\,s \in \mathcal{C}_\tau$ then $\mathsf{R}\, t\,\vec{s} \in \mathcal{C}_{\sigma\to\tau}$.*
- $0 \in \mathcal{C}_N$.
- *If $s \in \mathcal{C}_N$ then $\mathsf{s}s \in \mathcal{C}_N$.*
- *If $s \in \mathcal{C}_N$ and $t_0\,\vec{s} \in \mathcal{C}_\tau$ then $\mathsf{cond}\, t_0\, t_1\,\vec{s}0 \in \mathcal{C}_\tau$.*
- *If $s \in \mathcal{C}_N$ and $t_1\,\vec{s}\,s \in \mathcal{C}_\tau$ then $\mathsf{cond}\, t_0\, t_1\,\vec{s}\mathsf{s}s \in \mathcal{C}_\tau$.*

This is proved by an induction on the reduction trees of $\vec{s}, s, \vec{r}, r$ (which, again, are strongly normalising), in most cases appealing directly to the convertibility lemma above. For the L case we rely on closure of $\mathcal{C}$ under application, cf. Proposition 5.5, and for the R case we must employ a sub-induction on the reduction tree of an input $s \in \mathcal{C}_\sigma$.

As a consequence of our results in Sections 3 and 4, observe that any $s \in \mathcal{C}_N$ reduces to a unique numeral. This is because $\mathcal{C}_N$ contains *only* CT-coterms, by definition, which are weakly normalising and confluent. From here we may establish the main result of this subsection:

▶ **Theorem 5.8** (Convertibility for CT). *Any CT-coderivation $t : \vec{\sigma} \Rightarrow \tau$ is in $\mathcal{C}_{\vec{\sigma}\to\tau}$.*

The proof constructs a "non-converting" branch similarly to Definition 4.2 (or the proof of Proposition 2.11). There is one subtlety, however, in the treatment of the cond case, requiring the uniqueness of normal forms for elements of $\mathcal{C}_N$. We obtain the required inputs for the premiss occurrences of N by an induction on the reduction tree of an input of the conclusion occurrence.

[12] Note that König's lemma is *equivalent* to arithmetical comprehension, i.e. ACA_0, already over RCA_0 (cf., e.g., [34]).

[13] All rules have type as presented in Figure 1.

Since $\mathcal{C}$ is closed under application, Proposition 5.5, we inherit $\mathcal{C}$ membership for all CT-coterms. Since elements of $\mathcal{C}$ are strongly normalising, again Proposition 5.5, and since reduction is confluent, Theorem 3.4, we finally have:

▶ **Corollary 5.9** (Strong normalisation for CT). *Any closed CT coterm strongly normalises to a unique normal form.*

6 Conclusions

In this work we gave an interpretation of a theory of level n circular derivations (CT_n) into level $n+1$ T (T_{n+1}), by formalising models of CT within fragments of arithmetic and applying program extraction. This result is optimal by a converse result from parallel work [12]. In particular, CT_n and T_{n+1} are *equi-consistent*. We also showed confluence, strong normalisation, and continuity at type 2 for CT-coterms.

In future work it would be interesting to establish results on Curry-Howard aspects of our underlying type systems, establishing forms of cut-elimination and relationships with infinitary lambda-calculi. Ideas from [4, 15, 3] may prove useful to this effect.

References

1 Bahareh Afshari and Graham E. Leigh. Cut-free completeness for modal mu-calculus. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20–23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005088.

2 Jeremy Avigad and Solomon Feferman. Gödel's functional ("dialectica") interpretation. *Handbook of Proof Theory*, 137:337–405, 1998.

3 David Baelde, Amina Doumane, Denis Kuperberg, and Alexis Saurin. Bouncing threads for infinitary and circular proofs. *CoRR*, abs/2005.08257, 2020. arXiv:2005.08257.

4 David Baelde, Amina Doumane, and Alexis Saurin. Infinitary proof theory: the multiplicative additive case. In *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29–September 1, 2016, Marseille, France*, pages 42:1–42:17, 2016. doi:10.4230/LIPIcs.CSL.2016.42.

5 Stefano Berardi and Makoto Tatsuta. Equivalence of inductive definitions and cyclic proofs under arithmetic. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20–23, 2017*, pages 1–12, 2017. doi:10.1109/LICS.2017.8005114.

6 James Brotherston, Dino Distefano, and Rasmus Lerchedahl Petersen. Automated cyclic entailment proofs in separation logic. In *CADE-23 – 23rd International Conference on Automated Deduction, Wroclaw, Poland, July 31–August 5, 2011. Proceedings*, pages 131–146, 2011. doi:10.1007/978-3-642-22438-6_12.

7 James Brotherston, Nikos Gorogiannis, and Rasmus L. Petersen. A generic cyclic theorem prover. In *Programming Languages and Systems – 10th Asian Symposium, APLAS 2012, Kyoto, Japan, December 11–13, 2012. Proceedings*, pages 350–367, 2012. doi:10.1007/978-3-642-35182-2_25.

8 James Brotherston and Alex Simpson. Complete sequent calculi for induction and infinite descent. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10–12 July 2007, Wroclaw, Poland, Proceedings*, pages 51–62, 2007. doi:10.1109/LICS.2007.16.

9 James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *J. Log. Comput.*, 21(6):1177–1216, 2011. doi:10.1093/logcom/exq052.

10 Samuel R. Buss, editor. *Handbook of Proof Theory*. Studies in Logic and the Foundations of Mathematics 137. Elsevier, 1998.

11 Anupam Das. On the logical complexity of cyclic arithmetic. *Log. Methods Comput. Sci.*, 16(1), 2020. doi:10.23638/LMCS-16(1:1)2020.

12 Anupam Das. A circular version of Gödel's T and its abstraction complexity, 2021. arXiv:2012.14421.

13 Anupam Das, Amina Doumane, and Damien Pous. Left-handed completeness for kleene algebra, via cyclic proofs. In Gilles Barthe, Geoff Sutcliffe, and Margus Veanes, editors, *LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Awassa, Ethiopia, 16–21 November 2018*, volume 57 of *EPiC Series in Computing*, pages 271–289. EasyChair, 2018. URL: https://easychair.org/publications/paper/SDqf.

14 Anupam Das and Damien Pous. A cut-free cyclic proof system for Kleene algebra. In *Automated Reasoning with Analytic Tableaux and Related Methods – 26th International Conference, TABLEAUX 2017, Brasília, Brazil, September 25–28, 2017, Proceedings*, pages 261–277, 2017. doi:10.1007/978-3-319-66902-1_16.

15 Anupam Das and Damien Pous. Non-wellfounded proof theory for (kleene+action)(algebras+lattices). In Dan R. Ghica and Achim Jung, editors, *27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4–7, 2018, Birmingham, UK*, volume 119 of *LIPIcs*, pages 19:1–19:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CSL.2018.19.

16 Christian Dax, Martin Hofmann, and Martin Lange. A proof system for the linear time μ-calculus. In S. Arun-Kumar and Naveen Garg, editors, *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13–15, 2006, Proceedings*, volume 4337 of *Lecture Notes in Computer Science*, pages 273–284. Springer, 2006. doi:10.1007/11944836_26.

17 Christian Dax, Martin Hofmann, and Martin Lange. A proof system for the linear time μ-calculus. In S. Arun-Kumar and Naveen Garg, editors, *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13–15, 2006, Proceedings*, volume 4337 of *Lecture Notes in Computer Science*, pages 273–284. Springer, 2006. doi:10.1007/11944836_26.

18 Amina Doumane. Constructive completeness for the linear-time μ-calculus. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20–23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005075.

19 Jérôme Fortier and Luigi Santocanale. Cuts for circular proofs: semantics and cut-elimination. In *Computer Science Logic 2013 (CSL 2013), September 2–5, 2013, Torino, Italy*, pages 248–262, 2013. doi:10.4230/LIPIcs.CSL.2013.248.

20 Petr Hájek and Pavel Pudlák. *Metamathematics of First-Order Arithmetic*. Perspectives in mathematical logic. Springer, 1993. URL: http://www.springer.com/mathematics/book/978-3-540-63648-9.

21 J. Roger Hindley and Jonathan P. Seldin. *Introduction to Combinators and λ-Calculus*. Cambridge University Press, USA, 1986.

22 Denis R. Hirschfeldt. *Slicing the truth: On the computable and reverse mathematics of combinatorial principles*. World Scientific, 2014.

23 S.C. Kleene. *Introduction to Metamathematics*. Bibliotheca Mathematica. North Holland, 7 edition, 1980.

24 Ulrich Kohlenbach. *Applied Proof Theory – Proof Interpretations and their Use in Mathematics*. Springer Monographs in Mathematics. Springer, 2008. doi:10.1007/978-3-540-77533-1.

25 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.

26 Denis Kuperberg, Laureline Pinault, and Damien Pous. Cyclic Proofs, System T, and the Power of Contraction. *Proceedings of the ACM on Programming Languages*, 2021. doi:10.1145/3434282.

27 John Longley and Dag Normann. *Higher-Order Computability*. Theory and Applications of Computability. Springer, 2015. doi:10.1007/978-3-662-47992-6.

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

29 Charles Parsons. On n-quantifier induction. *The Journal of Symbolic Logic*, 37(3):466–482, 1972.

30 Frank Pfenning. A proof of the church-rosser theorem and its representation in a logical framework. Technical report, Carnegie-Mellon University, Pittsburgh. Department of Computer Science., 1992.

31 Luigi Santocanale. A calculus of circular proofs and its categorical semantics. In Mogens Nielsen and Uffe Engberg, editors, *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8–12, 2002, Proceedings*, volume 2303 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 2002. doi:10.1007/3-540-45931-6_25.

32 B. Scarpellini. A model for barrecursion of higher types. *Compositio Mathematica*, 23(1):123–153, 1971. URL: http://eudml.org/doc/89072.

33 Alex Simpson. Cyclic arithmetic is equivalent to Peano arithmetic. In *Foundations of Software Science and Computation Structures – 20th International Conference, FOSSACS 2017, Proceedings*, pages 283–300, 2017. doi:10.1007/978-3-662-54458-7_17.

34 Stephen G. Simpson. *Subsystems of second order arithmetic*, volume 1. Cambridge University Press, 2009.

35 Thomas Studer. On the proof theory of the modal mu-calculus. *Stud Logica*, 89(3):343–363, 2008. doi:10.1007/s11225-008-9133-6.

36 William W. Tait. Intensional interpretations of functionals of finite type I. *J. Symb. Log.*, 32(2):198–212, 1967. doi:10.2307/2271658.

37 Terese. *Term rewriting systems*, volume 55 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 2003.

38 Anne S. Troelstra. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*. Lecture Notes in Mathematics. Springer, 1 edition, 1973. URL: http://gen.lib.rus.ec/book/index.php?md5=5E0718442C0178C4B23065E54AC7889C.

39 Chuangjie Xu. A syntactic approach to continuity of T-definable functionals. *Logical Methods in Computer Science*, Volume 16, Issue 1, 2020. doi:10.23638/LMCS-16(1:22)2020.

A Further material for Section 4

Proof of Proposition 4.3. Let us write $\mathrm{Gen}(i,(t_0,\vec{s}_0),(t_i,\vec{s}_i))$ for "t_i and $\vec{s}_i$ are the i^{th} sequent and input tuple generated by t_0 and $\vec{s}_0$". Notice that the construction of t_i and $\vec{s}_i$ itself is recursive in HR_n, t_0 and $\vec{s}_0$, and so Gen is certainly recursion-theoretically $\Delta^0_{n+2}(t_0,\vec{s}_0)$, by appealing to Fact 3.14. To formally prove that Gen is Δ^0_{n+2} *inside* our theory, it suffices to show determinism:

$$\forall i.\forall(t_i,\vec{s}_i),(t_i',\vec{s}_i').\left(\begin{array}{ll} & \mathrm{Gen}(i,(t_0,\vec{s}_0),(t_i,\vec{s}_i))\wedge\mathrm{Gen}(i,(t_0,\vec{s}_0),(t_i',\vec{s}_i')) \\ \implies & t_i=t_i'\wedge\vec{s}_i=\vec{s}_i' \end{array}\right)$$

Writing Gen syntactically as a Σ^0_{n+2} formula, the above may be directly proved by Π^0_{n+2}-induction on i, appealing to the cases of Definition 4.2 above.

It remains to show that the construction is total, i.e. that each $(t_i,\vec{s}_i)$ actually exists. In fact we will simultaneously prove this and the inductive invariant of the construction, so the formula,

$$\exists(t_i,\vec{s}_i).(\mathrm{Gen}(i,(t_0,\vec{s}_0),(t_i,\vec{s}_i))\wedge t_i\,\vec{s}_i\ \notin \mathsf{HR}_{\tau_i}) \tag{7}$$

by induction on i. Note that, since $\mathsf{lev}(\tau_i) \leq n$ we have that HR_{τ_i} is Π^0_{n+1} by Fact 3.14, and so $t_i\vec{s}_i \notin \mathsf{HR}_{\tau_i}$ is Σ^0_{n+1}, whereas $\mathrm{Gen}(i, (t_0, \vec{s}_0), (t_i, \vec{s}_i))$ is Δ^0_{n+2} as already mentioned. Thus the inductive invariant in (7) is indeed Σ^0_{n+2}.

First, to justify (1), let us consider the possible initial sequents:

- For the 0 rule: we have $0 \in \mathsf{HR}_N$ by definition;
- For the s rule: if $t \in \mathsf{HR}_N$, then $t \approx \underline{n}$ for some $n \in \mathbb{N}$, by definition of HR_N, and so also $\mathsf{s}t \approx \mathsf{s}\underline{n}$, by closure of $\approx$ under contexts. Hence $\mathsf{s}t \in \mathsf{HR}_N$.
- For an id_σ rule: if $s \in \mathsf{HR}_\sigma$ then $\mathsf{id}\, s \approx s$ by id reduction. Hence $\mathsf{id}\, s \in \mathsf{HR}_\sigma$.

Now, the base case, for $i = 0$, follows by the assumption on t_0 and $\vec{s}_0$, so let us assume that $\mathrm{Gen}(i, (t_0, \vec{s}_0), (t_i, \vec{s}_i))$ and $t_i\, \vec{s}_i \notin \mathsf{HR}_{\tau_i}$. We will witness the existential of the inductive invariant with the coderivation t_{i+1} and inputs $\vec{s}_{i+1}$ as given in Definition 4.2 above (justifying their existence when necessary), showing $t_{i+1}\, \vec{s}_{i+1} \notin \mathsf{HR}_{\tau_{i+1}}$. We shall also adopt the same notation for inputs and types as in Definition 4.2.

For (2), the wk case, we have:

	$t_i\, \vec{s}_i \notin \mathsf{HR}_\tau$	by inductive hypothesis
$\therefore$	$\mathsf{wk}\, t\, \vec{s}\, s \notin \mathsf{HR}_{\tau_i}$	by definitions
$\therefore$	$t\, \vec{s} \notin \mathsf{HR}_\tau$	by $\rightsquigarrow_{\mathsf{wk}}$ and closure of HR_τ under $\approx$
$\therefore$	$t_{i+1}\, \vec{s}_{i+1} \notin \mathsf{HR}_{\tau_{i+1}}$	by definitions

For (3), the ex case, we have:

	$t_i\, \vec{s}_i \notin \mathsf{HR}_{\tau_i}$	by inductive hypothesis
$\therefore$	$\mathsf{ex}\, t\, \vec{r}\, r\, s\, \vec{s} \notin \mathsf{HR}_\tau$	by definitions
$\therefore$	$t\, \vec{r}\, s\, r\, \vec{s} \notin \mathsf{HR}_\tau$	by $\rightsquigarrow_{\mathsf{ex}}$ and $\because \mathsf{HR}_\tau$ closed under $\approx$
$\therefore$	$t_{i+1}\, \vec{s}_{i+1} \notin \mathsf{HR}_{\tau_{i+1}}$	by definitions

For (4), the cntr case, we have:

	$t_i\, \vec{s}_i \notin \mathsf{HR}_{\tau_i}$	by inductive hypothesis
$\therefore$	$\mathsf{cntr}\, t\, \vec{s}\, s \notin \mathsf{HR}_\tau$	by definitions
$\therefore$	$t\, \vec{s}\, s\, s \notin \mathsf{HR}_\tau$	by $\rightsquigarrow_{\mathsf{cntr}}$ and $\because \mathsf{HR}_\tau$ closed under $\approx$
$\therefore$	$t_{i+1}\, \vec{s}_{i+1} \notin \mathsf{HR}_{\tau_{i+1}}$	by definitions

For (5), the cut case, assume without loss of generality that $t\, \vec{s} \in \mathsf{HR}_\tau$. We have:

	$t_i\, \vec{s}_i \notin \mathsf{HR}_{\tau_i}$	by inductive hypothesis
$\therefore$	$\mathsf{cut}\, t\, t'\, \vec{s} \notin \mathsf{HR}_\tau$	by definitions
$\therefore$	$t'\, \vec{s}\, (t\, \vec{s}) \notin \mathsf{HR}_\tau$	by $\rightsquigarrow_{\mathsf{cut}}$ and $\because \mathsf{HR}_\tau$ closed under $\approx$
$\therefore$	$t_{i+1}\, \vec{s}_{i+1} \notin \mathsf{HR}_{\tau_{i+1}}$	by definitions

For (6), the L case, assume without loss of generality that $t\, \vec{s} \in \mathsf{HR}_\tau$, and so also $s\, (t\, \vec{s}) \in \mathsf{HR}_\sigma$ by Proposition 3.15. We have:

	$t_i\, \vec{s}_i \notin \mathsf{HR}_{\tau_i}$	by inductive hypothesis
$\therefore$	$\mathsf{L}\, t\, t'\, \vec{s}\, s \notin \mathsf{HR}_\tau$	by definitions
$\therefore$	$t'\, \vec{s}\, (s\, (t\, \vec{s})) \notin \mathsf{HR}_\tau$	by $\rightsquigarrow_{\mathsf{L}}$ and $\because \mathsf{HR}_\tau$ closed under $\approx$
$\therefore$	$t_{i+1}\, \vec{s}_{i+1} \notin \mathsf{HR}_{\tau_{i+1}}$	by definitions

For (7), the R case, we have:

	$t_i\,\vec{s}_i \notin \mathsf{HR}_{\tau_i}$	by inductive hypothesis
$\therefore$	$\mathsf{R}\,t\,\vec{s} \notin \mathsf{HR}_{\sigma\to\tau}$	by definitions
$\therefore$	$\exists s' \in \mathsf{HR}_\sigma.\ \mathsf{R}\,t\,\vec{s}\,s' \notin \mathsf{HR}_\tau$	by definition of $\mathsf{HR}_{\sigma\to\tau}$
$\therefore$	$\exists s' \in \mathsf{HR}_\sigma.\ t\,\vec{s}\,s' \notin \mathsf{HR}_\tau$	by $\rightsquigarrow_{\mathsf{R}}$ and $\because \mathsf{HR}_\tau$ closed under $\approx$
$\therefore$	$t\,\vec{s}\,s \notin \mathsf{HR}_\tau$	$\because s$ is well-defined by Σ^0_{n+1}-minimisation
$\therefore$	$t_{i+1}\,\vec{s}_{i+1} \notin \mathsf{HR}_{\tau_{i+1}}$	by definitions

In the penultimate step, note that we have from the inductive hypothesis $\exists s(s \in \mathsf{HR}_\sigma \wedge t\,\vec{s}\,s \notin \mathsf{HR}_\tau)$, where $\mathsf{lev}(\sigma) < n$ and $\mathsf{lev}(\tau) \leq n$. Thus $(s \in \mathsf{HR}_\sigma \wedge t\,\vec{s}\,s \notin \mathsf{HR}_\tau)$ is indeed Σ^0_{n+1}, by Fact 3.14, and so Σ^0_{n+1}-minimisation applies.

For (8), the cond case, note by the inductive hypothesis we have $r \in \mathsf{HR}_N$ so by definition of HR_N and confluence, we have that r converts to a unique numeral. Thus the two cases considered by the definition of t_{i+1} and $\vec{s}_{i+1}$ are exhaustive and exclusive, and we consider each separately.

If $r \approx 0$ then we have:

	$t_i\,\vec{s}_i \notin \mathsf{HR}_{\tau_i}$	by inductive hypothesis
$\therefore$	$\mathsf{cond}\,t\,t'\vec{s}\,r \notin \mathsf{HR}_\tau$	by definitions
$\therefore$	$\mathsf{cond}\,t\,t'\vec{s}\,0 \notin \mathsf{HR}_\tau$	by assumption and $\because \mathsf{HR}_\tau$ closed under $\approx$
$\therefore$	$t\,\vec{s} \notin \mathsf{HR}_\tau$	by $\rightsquigarrow_{\mathsf{cond}}$ and $\because \mathsf{HR}_\tau$ closed under $\approx$
$\therefore$	$t_{i+1}\,\vec{s}_{i+1} \notin \mathsf{HR}_{\tau_{i+1}}$	by definitions

If $r \approx \mathsf{s}\underline{n}$ then we have:

	$t_i\,\vec{s}_i \notin \mathsf{HR}_{\tau_i}$	by inductive hypothesis
$\therefore$	$\mathsf{cond}\,t\,t'\vec{s}\,r \notin \mathsf{HR}_\tau$	by definitions
$\therefore$	$\mathsf{cond}\,t\,t'\vec{s}\,\mathsf{s}\underline{n} \notin \mathsf{HR}_\tau$	by assumption and $\because \mathsf{HR}_\tau$ closed under $\approx$
$\therefore$	$t'\vec{s}\,\underline{n} \notin \mathsf{HR}_\tau$	by $\rightsquigarrow_{\mathsf{cond}}$ and $\because \mathsf{HR}_\tau$ closed under $\approx$
$\therefore$	$t_{i+1}\,\vec{s}_{i+1} \notin \mathsf{HR}_{\tau_{i+1}}$	by definitions

This concludes the proof. ◀

Proof of Proposition 4.5. We shall prove only the "moreover" clause, the former following a fortiori. First, suppose we have a (finite) N-thread $(N^i)_{i=k}^l$ beginning at t_k. Let $s_i \in \vec{s}_i$ be the corresponding input of N^i for $1 \leq i \leq l$, and let each $r_i \approx \underline{n}_i$, for unique $n_i \in \mathbb{N}$, by definition of HR_N and confluence. Letting m be the number of times that $(N^i)_{i=1}^l$ progresses, we may show by induction on l that $n_l \leq n_k - m$, using Lemma 4.4 for the inductive steps.

Now, to prove the "moreover" statement, fix some k and let $\vec{N}^k \subseteq \vec{\sigma}_k$ exhaust the N occurrences in $\vec{\sigma}_k$. Let $\vec{r}_k \subseteq \vec{s}_k$ be the corresponding inputs, and write $\vec{n}_k$ for the unique natural numbers such that each $r_{ki} \approx \underline{n}_{ki}$, by definition of HR_N and confluence. We may now simply set $m := \max \vec{n}_k$, whence no thread from t_k may progress more than m times by the preceding paragraph. ◀

Proof of Theorem 4.6. First, by Proposition 2.15 (from [11]), we have that RCA_0 proves that t is progressing. Consequently RCA_0 proves that, for any branch $(t_i)_i$, there is some k s.t. there are arbitrarily often progressing finite threads beginning from t_k:[14]

$$\exists k.\forall m.\text{ there is a (finite) } N\text{-thread from } t_k \text{ progressing} > m \text{ times} \tag{8}$$

[14] The argument for this is similar to that of Proposition 6.2 from [11].

Note that this statement is purely arithmetical in $(t_i)_i$ and so, if $(t_i)_i$ is Δ^0_{n+2}-well-defined, then in fact $\mathsf{RCA}_0 + I\Sigma^0_{n+2}$ proves (8), by conservativity over $I\Sigma_{n+2}((t_i)_i)$ and then substitution of the Δ_{n+2}-definition of $(t_i)_i$.

Now, working inside $\mathsf{RCA}_0 + I\Sigma^0_{n+2}$, suppose for contradiction that $\vec{s} \in \mathsf{HR}_{\vec{\sigma}}$ s.t. $t\,\vec{s} \notin \mathsf{HR}_\tau$. By Proposition 4.3, we can Δ^0_{n+2}-well-define the branch $(t_i)_i$ generated by t and $\vec{s}$. Thus we indeed have (8), contradicting Proposition 4.5. ◀

Proof sketch of Theorem 4.8. Let us work inside $\mathsf{RCA}_0 + I\Sigma^0_{n+2}$. By Theorem 4.6 we have that $s, t \in \mathsf{HR}_\sigma$, so suppose that $CT_n \vdash s = t$ (which is a Σ^0_1 relation). Now, invoking Lemma 3.17 and by verifying the other axioms for FARs in general, we indeed have that $s \approx t$, by Σ^0_1-induction on the CT_n proof of $s = t$.

Now, invoking the extraction theorem, Proposition 2.14, for the above paragraph, we can extract a T_{n+1}-term $d(\cdot)$ witnessing the following "reflection" principle:

$$T_{n+1} \vdash \text{“}P \text{ is a } CT_n \text{ proof of } s = t\text{”} \;\supset\; d(P) : s \approx t$$

We may duly substitute a concrete CT_n proof P of $s = t$ into the above principle to conclude that $T_{n+1} \vdash s \approx t$, as required. ◀

B Further material for Section 5

Proof sketch of Theorem 5.1. The argument is essentially the same as that for Theorem 4.6. Assuming otherwise, for contradiction, we may generate a non-hereditarily-total branch just as in Definition 4.2, and its well-definedness is shown just as in Proposition 4.3. Note that all induction/minimisation used is in fact arithmetical in $\rightsquigarrow$ and $\mathsf{HR}^{\vec{f}}_\sigma$, so the branch is indeed $\Delta^0_{n+2}(\vec{f})$-well-defined (for n the maximal type level in t).

Since we no longer concern ourselves with the refinement of type levels, the remainder of the argument is actually simpler than that of Section 4. Instead of dealing with the arithmetical approximation of progressiveness, we may immediately access the generated non-total branch *as a set*, thanks to the availability of arithmetical comprehension in ACA_0. We also have a suitable version of Lemma 4.4 for $\mathsf{HR}^{\vec{f}}_N$, this time using UNF_N instead of confluence, and so the appropriate contradiction of the well-ordering property of $\mathbb{N}$ is readily obtained. ◀

▶ **Observation B.1.** *If $s \in \mathcal{C}_N$ then s reduces to a unique numeral.*

Proof. Since $\mathcal{C}_N$ contains *only* CT-coterms, we have as a special case of Theorem 4.6 that $s \approx \underline{n}$ for some $n \in \mathbb{N}$. By confluence, we have that n is unique and furthermore $s \rightsquigarrow^* \underline{n}$. ◀

Proof of Theorem 5.8. Suppose for contradiction we have $\vec{s} \in \mathcal{C}_{\vec{\sigma}}$ such that $t\,\vec{s} \notin \mathcal{C}_\tau$. We define a branch $(t_i : \vec{\sigma}_i \Rightarrow \tau_i)_i$ of t and inputs $\vec{s}_i \in \mathcal{C}_{\vec{\sigma}_i}$ s.t. $t_i\,\vec{s}_i \notin \mathcal{C}_{\tau_i}$ by induction on i just like in Definition 4.2 (or the proof of Proposition 2.11). The only difference is that we use Proposition 5.7 above for preservation in $\mathcal{C}$ rather than the analogous closure properties for HR (or $\mathfrak{N}$).

There is one subtlety, which is the treatment of the cond case. Suppose we have a regular progressing coderivation,

$$\mathsf{cond}\;\dfrac{\overset{t}{\triangledown}\quad \vec{\sigma} \Rightarrow \tau \qquad \overset{t'}{\triangledown}\quad \vec{\sigma}, N \Rightarrow \tau}{\vec{\sigma}, N \Rightarrow \tau}$$

and $\vec{s}_i = (\vec{s}, s)$ with $\vec{s} \in \mathcal{C}_{\vec{\sigma}}$, $s \in \mathcal{C}_N$ and $\mathsf{cond}\,t\,t'\vec{s}\,s \notin \mathcal{C}_\tau$. Since $s \in \mathcal{C}_N$ we have from Observation B.1 that s reduces to a unique numeral $\underline{n}$. We will show that,

- if $n = 0$ then $t\,\vec{s} \notin \mathcal{C}_\tau$; and,
- if $n = m + 1$ then there is some $r \in \mathcal{C}_N$ reducing to $\underline{m}$ with $t'\,\vec{s}\,r \notin \mathcal{C}_\tau$;

by induction on $\mathrm{RedTree}(\vec{s}) + \mathrm{RedTree}(s)$. By the conversion lemma, Lemma 5.6, there must be a reduction from $\mathsf{cond}\,t\,t'\,\vec{s}\,s$ not reaching $\mathcal{C}_\tau$. Let us consider the possible cases:

- If $s = 0$ and $\mathsf{cond}\,t\,t'\,\vec{s}\,s \rightsquigarrow t\,\vec{s} \notin \mathcal{C}_\tau$ then we are done.
- If $s = \mathsf{s}r$ and $\mathsf{cond}\,t\,t'\,\vec{s}\,s \rightsquigarrow t'\,\vec{s}\,r \notin \mathcal{C}_\tau$ then we are done. (Note that such r must strongly normalise to $\underline{m}$, and so in particular $r \in \mathcal{C}_N$).
- If $\mathsf{cond}\,t\,t'\,\vec{s}\,s \rightsquigarrow \mathsf{cond}t\,t'\vec{s}'s' \notin \mathcal{C}_\tau$, then by the inductive hypothesis either,
  - $n = 0$ and $t\,\vec{s}' \notin \mathcal{C}_\tau$, so $t\,\vec{s} \notin \mathcal{C}_\tau$ by Proposition 5.5.(2); or,
  - $n = m + 1$ and there is some $r \in \mathcal{C}_N$ reducing to $\underline{m}$ s.t. $t'\,\vec{s}'\,r \notin \mathcal{C}_\tau$, so $t'\,\vec{s}\,r \notin \mathcal{C}_\tau$ by Proposition 5.5.(2).

From here, any progressing thread $(N^i)_{i\geq k}$ along $(t_i)_i$ yields a sequence of coterms $(r_i \in \mathcal{C}_N)_{i\geq k}$ that, under normalisation, induces an infinitely often descending sequence of natural numbers, yielding the required contradiction. ◀