UNIVERSITY OF BIRMINGHAM
University of Birmingham Research at Birmingham

A circular version of Gödel's T and its abstraction complexity

Das, Anupam

Citation for published version (Harvard):
Das, A 2020 'A circular version of Gödel's T and its abstraction complexity'.

Link to publication on Research at Birmingham portal

General rights
Unless a licence is specified above, all rights (including copyright and moral rights) in this document are retained by the authors and/or the copyright holders. The express permission of the copyright holder must be obtained for any use of this material other than for purposes permitted by law.

•Users may freely distribute the URL that is used to identify this publication.
•Users may download and/or print one copy of the publication from the University of Birmingham research portal for the purpose of private study or non-commercial research.
•User may use extracts from the document in line with the concept of 'fair dealing' under the Copyright, Designs and Patents Act 1988 (?)
•Users may not further distribute the material nor use it for the purposes of commercial gain.

Where a licence is displayed above, please note the terms and conditions of the licence govern your use of this document.

When citing, please reference the published version.

Take down policy
While the University of Birmingham exercises care and attention in making items available there are rare occasions when an item has been uploaded in error or has been deemed to be commercially or otherwise sensitive.

If you believe that this is the case for this document, please contact UBIRA@lists.bham.ac.uk providing details and we will remove access to the work immediately and investigate.

Download date: 18. Apr. 2024

arXiv:2012.14421v2 [cs.LO] 16 Jan 2021

A CIRCULAR VERSION OF GÖDEL'S T AND ITS ABSTRACTION COMPLEXITY

ANUPAM DAS*

Abstract. *Circular* and *non-wellfounded* proofs have become an increasingly popular tool for metalogical treatments of systems with forms of induction and/or recursion. In this work we investigate the expressivity of a variant CT of Gödel's system T where programs are circularly typed, rather than including an explicit recursion combinator. In particular, we examine the abstraction complexity (i.e. type level) of CT, and show that the Gödel primitive recursive functionals may be typed more *succinctly* with circular derivations, using types precisely one level lower than in T. In fact we give a logical correspondence between the two settings, interpreting the quantifier-free type 1 *theory* of level $n+1$ T into that of level n CT and vice-versa.

We also obtain some further results and perspectives on circular 'derivations', namely strong normalisation and confluence, models based on hereditary computable functionals, continuity at type 2, and a translation to terms of T computing the same functional, at all types.

1. Introduction

In recent years *non-wellfounded* proofs have attracted increasing attention. The modern inception of the area arguably lies with the celebrated work [NW96] of Niwinski and Walukiewicz, where a circular analytic tableau system was proved sound and complete for Kozen's modal μ-calculus [Koz83]. Since then several distinct lines of research have emerged, in particular pivoting towards *proof theoretical* aspects of non-wellfounded reasoning:

- *Modal logic.* Niwinski and Walukiewicz's system has been recast as more traditional sequent based systems in, e.g., the works [DHL06a] for the linear time fragment, and [Stu08] for the general fragment, both offering alternative cut-free completeness proofs. Recently more 'constructive' proofs have emerged of both these results, namely [Dou17] and [AL17]. Similar results may be readily recovered for many other fixed point modal logics.
- *Predicate logic.* Brotherston and Simpson initiated a program for non-wellfounded proof theory based on predicate logic [BS07, BS11]. Here, syntactic correctness criteria from modal logic, themselves inspired by automaton theory, were adapted to provide a sound circular proof theory for forms of *inductive definitions*. This approach has found applications in automated theorem proving [BGP12, BDP11, RB17], and more recently variations of these systems in the setting of first-order arithmetic have been investigated [Sim17, BT17b, Das20, CR20].

*University of Birmingham
E-mail address: a.das@bham.ac.uk.

- *Type systems.* Recently, formulations of cyclic proofs through the lens of the *Curry-Howard* correspondence have garnered attention in the French school of proof theory. Fortier and Santocanale, also inspired by [NW96], seem to have been the first of the modern era to give a *cut-elimination* result for circular proofs with the aforementioned correctness conditions,[1] namely for *additive linear logic* with least and greatest fixed points [FS13]. This was generalised in later work to the logic including multiplicatives too (for closed formulas) in [BDS16], and with more expressive correctness conditions recently in [BDKS20]. Deepening this Curry-Howard viewpoint, presentations of 'proof nets' have recently appeared [DS19], yielding a form of Natural Deduction for circular proofs.
- *Algebras.* Inspired by the aforementioned work on type systems, there have been many recent applications to classes of algebras based on fixed points, in particular the *Kleene star*. [DP17] has presented a cut-free complete system for Kleene algebra, and established a cut-elimination result for it and the extension by *residuals* and *meets* (i.e. Lambek calculus + Kleene star) in [DP18]. These have been used directly to obtain alternative completeness results, e.g. [DDP18], and have inspired recent undecidability results, e.g. for the logic of *action lattices* [Kuz19], which solved a longstanding open problem.

A key motivation in all these areas is the so-called 'Brotherston-Simpson conjecture': are cyclic proofs and inductive ones equally powerful? Naturally, the answer depends on how one interprets 'equally powerful', e.g. as provability, proof complexity, logical complexity etc., as well as on the logic at hand. One example of this nuance is readily found in the setting of first-order logic:

- Berardi and Tatsuta have shown that first-order proofs with induction, over particular inductive definitions, do not prove the same theorems as cyclic ones [BT17a].
- Simpson in [Sim17] and Berardi and Tatsuta in [BT17b] have independently shown that inductive and cyclic proofs are equally powerful in the presence of Peano Arithmetic.
- Both these results were refined in [Das20] where it was shown that provability by cyclic proofs containing only Σ_n formulas ($C\Sigma_n$) coincides with provability by Σ_{n+1}-induction ($I\Sigma_{n+1}$), over Π_{n+1} theorems.

The current work is somewhat inspired by the line of work just mentioned, in the sense that it attempts to understand the Brotherston-Simpson question for type systems and their corresponding equational theories at the level of *abstraction complexity* (or *type level*). In particular, motivated by the results for arithmetic mentioned above, our natural starting point is Gödel's 'system T' [Gö58].

T is a multi-sorted classical quantifier-free theory over a simply typed programming language based on primitive recursion at all finite types. Gödel's celebrated *Dialectica* functional interpretation [Gö58] allows all of first-order arithmetic to be interpreted by this simple theory,[2] essentially trading off logical complexity of

[1] Of course, in this regard, one must also mention Mints' famous 'continuous cut-elimination' [Min78]; while his approach indeed seems to apply, the main difficulty herein seems to be the preservation of syntactic correctness at the limit.

[2] In fact, Gödel's interpretation was only for the intuitionistic theory Heyting Arithmetic. Peano Arithmetic may be duly interpreted upon composition with a suitable double negation translation.

quantifiers for abstraction complexity in functional programs. This tradeoff was made precise by Parsons in [Par72]: the fragment $I\Sigma_{n+1}$ of Peano Arithmetic is ND-interpreted into the fragment of T admitting only type n recursion (T_n). Naturally, a converse result holds too, in the sense that T_n may be interpreted into appropriate fragments of arithmetic where the interpretation of types is relativised to classes of hereditarily computable functionals.

In this work we present a *circular* (or *cyclic*) version CT of Gödel's T, where typing derivations may be non-wellfounded, but remain finitely presentable. At the level of the type system, we simply adapt the correctness conditions of many of the previously mentioned works to the language of simple types. At the level of the theory we admit the same axioms as T, in particular quantifier-free induction, and include a form of extensional equality. Our main result is that, similar to the arithmetic setting, CT and T are mutually interpretable. In fact, we obtain a similar refinement: the type n restriction of CT (CT_n) is interpretable in T_{n+1} and vice versa, over at least the type 1 quantifier-free theory. Intuitively this means that cyclic typing derivations, and their induced theory, are more succinct than ones in T, by precisely 1 type level.

Our arguments, however, are more subtle and technical than the analogous ones from [Das20], since we are working simultaneously with systems for typing and systems for reasoning within a theory. We take a *proof mining* approach to interpreting CT in T, formalising a totality argument in suitable fragments of 'second-order'[3] arithmetic. Extraction of witnessing functionals and corresponding specifications in fragments of T follow by the aforementioned results of Parsons, under well-known conservation results over fragments of first-order arithmetic. Notably, since we are unable to formalise the standard set theoretic model of higher order functionals, we take a detour through the model theory of T, in particular presenting 'coterm' models that play the roles of the *hereditarily recursive* (and *hereditarily effective*) operations. We give a rewriting theoretic implementation of program execution, and establish a *confluence* result within RCA_0, yielding determinism of normalising programs. The interpretation of CT in T is then a consequence of the fact that these type structures indeed constitute models of CT. For the refinement at each type level, we take advantage of recent results on the reverse mathematics of cyclic proof checking from [Das20], inspired by [KMPS19b, KMPS19a].

Our ultimate motivation is to establish a correspondence between two of the proof theoretic worlds mentioned at the start: predicate logic and type systems. In future work we would like to establish a 'circular Dialectica' functional interpretation, thereby completing the picture and formally associating the two settings.

1.1. **Related work.** Kuperberg, Pinault and Pous have notably also studied non-wellfounded typing derivations inspired by T in [KPP21], in particular investigating *affinity*. Their types are closed under a 'Kleene star' operation for list formation, inspired by previous works such as [DP18], and are equivalent to the usual notion of simple (or *finite*) types. They show that the affine fragment of this type system, where *contraction* is omitted, computes precisely the primitive recursive functions in the standard set-theoretic model, generalising a similar result by Dal Lago for affine T [Lag09]. They also show that, in the presence of contraction, their type 1

[3]As for simple type theory, the allusion to 'second' or 'higher' order is only suggestive: formally speaking these are multi-sorted first-order settings and not bona fide second or higher order.

fragment computes just the type 1 primitive recursive function*al*s (à la Gödel), i.e. those computed by type 1 terms of T in the standard model.

This latter result is subsumed by and, in particular, refined in the current work in terms of type level (we discuss this further in Section 8.5). Moreover, the point of this work is to establish a *logical* correspondence between fragments of CT and T, i.e. at the level of their equational/quantifier-free theories, not only at the level of interpretation of their terms in the standard model.

1.2. **Outline and prerequisites.** The remainder of this paper is structured as follows. In Section 2 we give some preliminaries on Church's simple type theory and we recall system T in Section 3, in particular giving a sequent-style presentation of typing derivations.

In Section 4 we present non-wellfounded typing derivations ('coderivations'), in particular giving semantic results with respect to the standard model such as extensional completeness at type 1, Turing completeness for the regular fragment and, of course, well-definedness of the induced functionals. This section concludes with the definition of CT and its type-level-restricted fragments CT_n. In Section 5 we give a simulation of T_{n+1} within CT_n, over the type $n+1$ theory, in the presence of extensionality. The techniques of this section are entirely proof-theoretic.

In Section 6 we turn to the model theory of CT. We recast traditional type structures of hereditarily recursive and hereditarily effective operations into 'coterm' models, in light of the aforementioned Turing-completeness result. At the base level, program execution is implemented as a rewrite system induced from the equational axioms in the usual way; in particular we prove a confluence result, yielding determinism of computation. In Section 7 we formalise the aforementioned type structures within fragments of second-order arithmetic, in particular proving (within these fragments) that they constitute models of CT (with extensionality). By applying standard proof mining results, we obtain an interpretation of CT_n into T_{n+1}, over the type 1 equational theory, a converse result to that of Section 5.

Finally, in Section 8 we give some further results and perspectives on CT coterms, in particular obtaining type 2 continuity, weak and strong normalisation, and a translation to T terms computing the same functional. We also discuss proof theoretic strength, cut-elimination and the incorporation of fixed point operators.

It would not be pertinent to give a purely self-contained presentation of the content herein, since we rely on a number of established disciplines. That said, we aim for a level of exposition that highlights the significance and subtleties of our results and techniques for the general proof theorist.

Naturally, it is helpful to have some background with Gödel's system T and the Dialectica functional interpretation (though we shall not explicitly work with it), for which [AF98] is an excellent survey and [Tro73, Koh08] are more comprehensive. There are also excellent references for the technical disciplines underlining this work, namely rewriting theory (e.g. [Ter03]), reverse mathematics (e.g. [Sim09, Hir14]) and higher-order computability theory (e.g. [LN15]). Finally, we give metamathematical accounts of many of our results,[4] and so assume some familiarity with metamathematics of first- and second-order arithmetic (e.g. [HP93], also [Kle80] for a recursion-theoretic viewpoint and [Tro73] for a constructive viewpoint).

[4]This is for two reasons: (a) for self-contained interest; while some such results are probably folklore, they have not appeared elsewhere, as far as we know; and, more importantly, (b) since we exploit these metamathematical resuls in Section 7 in order to interpret CT within T.

2. Preliminaries on Church's simple type theory

At the heart of Gödel's T is a rudimentary version of Church's Simple Type Theory [Chu40]. Since we later consider a rather non-standard 'circular' calculus CT, we will here give a presentation of simple type theory that underlies both T and CT.

2.1. **Simple types.** Throughout this work we will deal with *terms* that are *simply typed*. (Simple) **types**, written σ, τ etc., are defined as follows:

- N is a type.
- If σ and τ are types, then so is $(\sigma \to \tau)$.

We typically omit parentheses on types when they are associated to the right. E.g., we may write $\rho \to \sigma \to \tau$ instead of $\rho \to (\sigma \to \tau)$ and so on.

We define the **level** of a type σ, written $\mathsf{lev}(\sigma)$, inductively as follows:

- $\mathsf{lev}(N) = 0$
- $\mathsf{lev}(\sigma \to \tau) = \max(1 + \mathsf{lev}(\sigma), \mathsf{lev}(\tau))$.

Every type σ can be uniquely written as $\sigma_1 \to \cdots \to \sigma_n \to N$, for some $n \in \mathbb{N}$. In this case we sometimes write $\vec{\sigma}$ for $(\sigma_1, \ldots, \sigma_n)$ and, as an abuse of notation, we also sometimes write $\vec{\sigma} \to N$ for σ. We call n here the **arity** of σ (in reference to the type isomorphism $\rho \to \sigma \to \tau \equiv (\rho \times \sigma) \to \tau$). Note that, in this case, $\mathsf{lev}(\sigma) = 1 + \max_{i=1}^{n}(\mathsf{lev}(\sigma_i))$.

2.2. **Simply typed theories (STTs).** A simply typed **language** is a multi-sorted first-order language, whose sorts are just the simple types. The simply typed languages considered in this work will consist of some basic set of constants of simple type, called *combinators*, as well as infinitely many *variables*, written x, y, z etc., of each simple type. We may sometimes indicate the type of a variable (or term) as superscript to aid parsing, e.g. writing x^σ for a variable x of type σ.

Terms are formed from constants (and variables) by typed *application*:

- Any constant or variable of type σ is a term of type σ.
- if s and t are terms of types σ and $\sigma \to \tau$, respectively, then $(t \circ s)$ is a term of type τ.

We usually just omit the application symbol $\circ$, e.g. writing $t\,s$ instead of $t \circ s$, and omit parentheses for long applications when they are associated to the left, e.g. writing $r\,s\,t$ for $(r\,s)\,t$ and so on. We do not include a λ-abstraction operation as primitive, instead requiring that it is coded by constants and composition for combinatory completeness (see, e.g., Fact 3).

Remark 1 (Application)**.** Formally speaking, being in a multi-sorted first-order framework, 'application' itself comprises a family of operations, one for each pair of types $(\sigma \to \tau, \sigma)$. We shall gloss over this formality in what follows, unless we need to distinguish application operations of differing types.

Simply typed languages, for us, always include a binary relation symbol $=_\sigma$ '**equality** at type σ', on each type σ. **Atomic formulas** have the form $s =_\sigma t$, where s and t are terms of type σ, although we shall suppress the subscript σ when it does not cause confusion. **Formulas** are built from atomic ones in the usual way, using $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), $\supset$ (implication), $\equiv$ (if and only if), $\exists$ (existential quantifier), and $\forall$ (universal quantifier).

Simply typed **theories** (or *STTs*) are (first-order, classical) theories over a (simply typed) language, typically specified by a set of quantifier-free axioms and rules. We always assume that STTs include the axioms for equality from Figure 1.

(1) $t =_\sigma t$, for any term t of type σ. (Reflexivity)
(2) $(s =_\sigma t \wedge \varphi(s)) \supset \varphi(t)$, for terms s, t of type σ. (Leibniz)

FIGURE 1. Axiom schemata for (intensional) equality.

We will also include a form of extensionality for equality given by the extensionality rule (ER) in Figure 2.

(ER) If $\vdash s\,\vec{x} = t\,\vec{x}$ then $\vdash s = t$.

FIGURE 2. Extensionality rule.

Remark 2 (On equality)**.** Our inclusion of an equality symbol in all finite types coupled with a rule for extensionality is non-standard, but it eases some of the technical development. Let us note, however, that our version has been previously considered in the literature, e.g. in [Par72] where ER is called 'SI' and facilitates the translation from $I\Sigma_{n+1}$ into T_n (cf. Section 3.5 later).

Our axiomatisation of equality thus sits somewhere between *intensional* or *weakly extensional* variants and the *fully extensional* variant. Since our principal concern in this work is in comparing STTs by their type 1 theories, the precise variant of equality is not so important, vis a vis known extensionality-elimination techniques at lower types, cf. [Luc73]. Weaker formulations, in particular weak extensionality, seem to suffice for certain results, but the appropriate proof adaptations seem to introduce technicalities that detract from our main purpose. Thus such considerations are beyond the scope of this work, but they will naturally be of greater importance for related directions, in particular for *proof interpretations*.

2.3. **Structures and the standard model.** We consider usual Henkin structures for simply typed languages, called *type structures*. Note that we do not, a priori, require $=_\sigma$ to be interpreted as true equality on the interpretation of σ.

One particular structure, the 'standard' or 'full set-theoretic' model $\mathfrak{N}$, is given by the following interpretation of types:

- $N^{\mathfrak{N}}$ is $\mathbb{N}$.
- $(\sigma \to \tau)^{\mathfrak{N}}$ is the set of functions $\sigma^{\mathfrak{N}} \to \tau^{\mathfrak{N}}$.
- $\circ^{\mathfrak{N}}$ is just function application, i.e. 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 $\forall x \in \sigma^{\mathfrak{N}}. f(x) =^{\mathfrak{N}}_\tau g(x)$.

As a notational convention, for a function $f : X \to Z^Y$ and $x \in X$, $y \in Y$, we may write $f(x, y)$ for $f(x)(y)$, and so on, in reference to the usual 'Currying' isomorphism $(Z^Y)^X \cong Z^{X\times Y}$.

This structure also has standard extensions to the STTs considered later in this work, which will be presented at the appropriate moments.

2.4. **Example: Combinatory Algebra.** The language of *combinatory algebra* consists of the following constants:

- $\mathrm{K}_{\sigma\tau}$, for each pair (σ, τ), of type $\sigma \to \tau \to \sigma$.
- $\mathrm{S}_{\rho\sigma\tau}$, for each triple (ρ, σ, τ), of type $(\rho \to \sigma \to \tau) \to (\rho \to \sigma) \to \rho \to \tau$.

We will typically omit the type subscripts of the combinators in what follows when it is unambiguous. **Combinatory Algebra** is a theory over this language that includes all axioms of the form:

$$\begin{array}{rcl} \mathrm{K}\, x\, y & = & x \\ \mathrm{S}\, x\, y\, z & = & x\, z\, (y\, z) \end{array} \tag{1}$$

The standard model $\mathfrak{N}$ may be extended to a model of Combinatory Algebra by taking the equations above as definitions (oriented from left to right). It is well-known that Combinatory Algebra is *complete*:

Fact 3. *For each term t of type τ and variable x of type σ, there is a term $\lambda x t$ of type $\sigma \to \tau$ s.t. (1) $\vdash$ $(\lambda x t)\, y \;=\; t[y/x]$.*

2.5. **Sequent style type system.** *Sequent calculi* give us a way to write typed terms that are more succinct with respect to type level, and also enjoy nice proof theoretic properties, e.g. cut-elimination. From the point of view of the *Curry-Howard correspondence*, they associate sequent proofs of minimal logic to simply typed terms. Importantly, the sequent presentation and its induced relations between type occurrences makes it easier to define our notion of progressing non-wellfounded derivation later.

Definition 4 (Sequent calculus)**.** Sequents are expressions $\vec{\sigma} \Rightarrow \tau$, where $\vec{\sigma}$ is a list of types and τ is a type. The rules for minimal logic are given in Figure 3.

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

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

FIGURE 3. Sequent style typing rules from minimal logic.

Here, and throughout this section, colours of each type occurrence in typing rules may be ignored for now and will become relevant later in Section 4. That said, it may be illustrative for the reader to imagine that, once we give interpretations of these rules, type occurrences of the same colour will correspond to identical inputs for the corresponding functionals. In this way, the colour assigned to a type on the LHS of a sequent is a sort of variable annotation for that occurrence.

Recall that, for a list of types $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$, we sometimes write $\vec{\sigma} \to \sigma$ for the type $\sigma_1 \to \cdots \to \sigma_n \to \sigma$. Each rule instance (or **step**) determines a constant of the appropriate type:

- A step $\dfrac{}{\vec{\tau} \Rightarrow \tau}$ is a constant of type $\vec{\tau} \to \tau$.
- A step $\dfrac{\vec{\sigma} \Rightarrow \sigma}{\vec{\tau} \Rightarrow \tau}$ is a constant of type $(\vec{\sigma} \to \sigma) \to \vec{\tau} \to \tau$.

- A step $\dfrac{\vec{\rho} \Rightarrow \rho \quad \vec{\sigma} \Rightarrow \sigma}{\vec{\tau} \Rightarrow \tau}$ is a constant of type $(\vec{\rho} \to \rho) \to (\vec{\sigma} \to \sigma) \to \vec{\tau} \to \tau$.

We will usually refer to steps only by their labels, e.g. ex, cntr, R etc., rather than explicitly indicating their types in the premisses and conclusions; unless otherwise clear from context, the associated typing should be assumed to be as given in the original specification of the rule, e.g. in Figure 3.

Definition 5 (Derivations and terms)**.** A sequent calculus **derivation** of $\vec{\sigma} \Rightarrow \tau$ determines a term of type $\vec{\sigma} \to \tau$ in the expected way, by applying all the inference steps according to its structure. Formally, derivations are construed as terms by inductively setting:

- $\mathsf{r}\dfrac{\overset{s}{\vec{\sigma} \Rightarrow \sigma}}{\vec{\tau} \Rightarrow \tau}$ is the term $\mathsf{r}\, s$; and,
- $\mathsf{r}\dfrac{\overset{r}{\vec{\rho} \Rightarrow \rho} \quad \overset{s}{\vec{\sigma} \Rightarrow \sigma}}{\vec{\tau} \Rightarrow \tau}$ is the term $\mathsf{r}\ r\, s$.

We write $t : \vec{\sigma} \Rightarrow \tau$ if t is a derivation of the sequent $\vec{\sigma} \Rightarrow \tau$ (and so also a term of type $\vec{\sigma} \to \tau$).

Note that, strictly speaking, derivations form a strict subset of all closed terms, since they are not formally closed under application. I.e., for $s : \Rightarrow \sigma$ and $t : \sigma \Rightarrow \tau$, we have that $t\, s$ is a term of type τ, but $t\, s$ is not, in general, a derivation.[5] In what follows, we will see that we can *interpret* the term $t\, s$ as the derivation $\mathsf{cut}\, t\, s$ (with appropriate types), but it will nonetheless be convenient to distinguish these two terms. In particular this interpretation does not admit the same notion of 'thread' for non-wellfounded derivations we consider later.

Remark 6 (Why rules as combinators?)**.** While it may seem strange to adopt a sequent style type system but construe inference steps as combinators rather than meta-level operations on, say, λ-terms, we adopt this approach to facilitate our later notions of non-wellfounded derivations and coterms. This combinatory approach ensures that the 'term associated to a derivation' is actually a *continuous* construction, so that when we later consider non-wellfounded derivations, the corresponding notion of 'coterm' is well-defined.

On the other hand, why do we use the sequent calculus at all? This is due to the particular termination criterion we will adopt for non-wellfounded derivations later, exploiting well-known notions of formula *ancestry* available in the sequent calculus. This is why the sequent calculus is the standard formalism in circular proof theory.

Definition 7 (Axiomatisation)**.** We define the axiomatisation in Figure 4, where the types corresponding to each rule label are as indicated in the corresponding rule instance in Figure 3.

[5] E.g., if t concludes with a binary step, then the only way to write $t\, s$ in the form $\mathsf{r}\, r_1 \cdots r_n$, with r an inference step, requires $n > 2$, which is not possible since all rules are at most binary.

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

FIGURE 4. Equational axiomatisation of sequent calculus rules.

Note that, here and elsewhere, there is no formal reason why we distinguish the arguments corresponding to subderivations by term meta-variables, s, t etc., and other arguments by variables x, y etc. It is purely in order to facilitate the identification of the corresponding arguments. All axioms are closed under substitution of terms for variables.

Remark 8 (Combinatory completeness)**.** The sequent calculus in Figure 3, under the axiomatisation in Figure 4, is equivalent to Combinatory Algebra (from Section 2.4). In particular there are derivations for $\mathrm{K}_{\sigma\tau}$ and $\mathrm{S}_{\rho\sigma\tau}$ that satisfy the corresponding equational axioms. As a consequence, our sequent calculus is also combinatory complete.

Remark 9 (Standard model)**.** The structure $\mathfrak{N}$ from Section 2.3 may be extended to one for Figures 3 and 4 by taking the equations of Figure 4 as definitions, oriented left-to-right.

Convention 10 (Rules modulo exchange)**.** In the rest of this work, it will often be convenient to omit instances of exchange, ex, in typing derivations and their corresponding notation as terms. For example, we may freely write a 'rule instance',

$$\mathsf{cut}\ \frac{\vec{\rho}, \vec{\sigma} \Rightarrow \sigma \qquad \vec{\rho}, \sigma, \vec{\sigma} \Rightarrow \tau}{\vec{\rho}, \vec{\sigma} \Rightarrow \tau} \tag{2}$$

instead of the corresponding derivation with exchanges, and an 'axiom',

$$\mathsf{cut}\, s\, t\, \vec{x}\, \vec{y} \;=\; t\, \vec{x}\, (s\, \vec{x}\, \vec{y})\, \vec{y}$$

instead of the corresponding equation derived using the ex and cut axioms. Again, we may omit typing of rule labels when it is unambiguous.

3. Preliminaries on Gödel's system T

So far we have not imposed any restrictions on our base type N, despite the fact that the standard model $\mathfrak{N}$ interprets N as $\mathbb{N}$. System T is a simple type theory that extends Combinatory Algebra by including new constants, axioms and rules that constrain the interpretation of N to this effect. Its definition is borne out over the following subsections.

3.1. Constants for natural numbers and recursion.

The language of T extends the sequent system from Figure 3 by the typing rules in Figure 5. Again, we may omit the subscript (and other typing information) of an instance of rec_τ when it is unambiguous.

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

FIGURE 5. Typing rules for 0, s and rec_τ combinators.

When writing terms, we assume s binds stronger than application. This is usually visually signified since the symbol s will appear in closer proximity to the term it is bound to. E.g. We write $s\ \mathsf{s}t$ for $s \circ (\mathsf{s} \circ t)$.

A **numeral** is a term of the form $\underbrace{\mathsf{s} \cdots \mathsf{s}}_{n} 0$, which we more succinctly write as $\underline{n}$.

3.2. **Recursion axioms.** T includes as axioms the equations from Figure 6. Be-

$$\begin{array}{rcl} \mathsf{rec}\ s\ t\ \vec{x}\ 0 & = & s\,\vec{x} \\ \mathsf{rec}\ s\ t\ \vec{x}\ \mathsf{s}y & = & t\ \vec{x}\ y\ (\mathsf{rec}\ s\ t\ \vec{x}\ y) \end{array}$$

FIGURE 6. Equational axioms for recursion combinators, where y is a variable of type N.

fore concluding our definition of T, let us note that the equational axioms thus far presented are enough to expose well-behaved computational content:[6]

Fact 11 ([Tai67])**.** *Orienting the axioms of Figures 4 and 6 left-to-right yields a terminating and confluent rewriting system on closed terms of T.*

We will not elaborate now on the rewriting theoretic aspects of T since we will revisit it in more detail later in Section 6. However, let us note that Tait's result above induces a well-behaved *term model* of T, with equality simply comparing (unique) normal forms. The point of Section 6 is to establish similar models for the non-wellfounded type system we will introduce in Section 4, formalised in the setting of second-order arithmetic.

3.3. **Number-theoretic axioms.** Finally, T includes the axioms from Figure 7, indicating that $(0, \mathsf{s})$ generates a free and inductive structure.

(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 7. Number-theoretic axioms for T, where x and y are variables of type N, and t is a term of type N.

This concludes the definition of T, i.e.:

Definition 12 (System T)**.** T is the simple type theory over the language given by Figures 3 and Figures 5, axiomatised by the formulas and rules from Figures 1, 2, 4, 6 and 7.

[6]Note that this result also applies to alternative combinatorial bases such as ours, e.g. as noted in [CH72], Chapter B.

Going back to Fact 11 and the succeeding discussion, one crucial property of T is that the only (closed) normal forms of type N are numerals. For the aforementioned term model induced by unique normal forms of closed terms, this property allows the verification of the induction schema (2) in Figure 7 to be reduced to induction at the meta-level. Interestingly, this property *fails* for the non-wellfounded calculus we will present in Section 4 (see Remark 44), necessitating a somewhat specialised construction of corresponding models in Section 6.

3.4. **The standard model and primitive recursive functionals.** The standard model $\mathfrak{N}$ from Section 2.3 may be extended to one of T by setting,

- $0^{\mathfrak{N}} := 0 \in \mathbb{N}$.
- $\mathsf{s}^{\mathfrak{N}}(n) := n + 1$.

and taking the axioms for rec from Figure 6 as definitions, oriented left-to-right. Note that the interpretation of rec is indeed well-defined by these axioms, provable by induction on $\mathbb{N}$.

The interpretations of terms in this model, i.e. the functionals $t^{\mathfrak{N}}$, form a higher-order function algebra known as the (Gödel) *primitive recursive functionals* of finite type, written PRF. It is well-known by a result of Kreisel that its type 1 functions coincide with those definable by effective transfinite recursion up to ε_0 [Kre51, Kre52], and moreover that ordinal complexity (height of an ω-tower) can be effectively traded off for abstraction complexity (type level) and vice-versa (cf., e.g., [Tai68, Sch75]).

3.5. **Restricting the level of recursors.** The main subject of study in this work will be fragments of T induced by restricting the type level of recursors.

Definition 13 (Fragments of T)**.** T_n is the restriction of T to the language containing only recursors rec_σ where $\mathsf{lev}(\sigma) \leq n$.

The significance of these fragments was investigated in the seminal work of Parsons [Par72]. In particular we have:

Proposition 14 ([Par72])**.** *If $I\Sigma_{n+1} \vdash \forall \vec{x} \exists y A(\vec{x}, y)$, where A is Δ_0, then there is a T_n term t with $T_n \vdash A(\vec{x}, t\,\vec{x})$.*[7]

In fact, this result is a direct consequence of Gödel's famous 'Dialectica' functional interpretation [Gö58], composed with a suitable negative translation. The converse, that $I\Sigma_{n+1}$ proves the totality of all type 1 terms of T_n, is obtained by formalising models of hereditarily computable functionals similar to those in Section 6. Both directions may be alternatively obtained via the aforementioned transfinite recursion theoretical characterisations of T, using purely structural proof theoretic methods, cf. [Bus95].

Both results naturally extend to the conservative extension $\mathsf{RCA}_0 + I\Sigma^0_{n+1}$.

Corollary 15. *If $\mathsf{RCA}_0 + I\Sigma_{n+1} \vdash \forall \vec{x} \exists y A(\vec{x}, y)$, where A is Δ^0_0, then there is a T_n term t with $T_n \vdash A(\vec{x}, t\,\vec{x})$.*

For the results of Section 5, it will be useful to have the following normal form of typing derivations:

[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.

Proposition 16 (Partial normalisation)**.** *Let $t : \vec{\sigma} \Rightarrow \tau$ be a T_n derivation, where τ and each σ_i have level $\leq n$. Then there is a T_n-derivation $t' : \vec{\sigma} \Rightarrow \tau$ such that $t^{\mathfrak{N}} = t'^{\mathfrak{N}}$. Moreover, $T_n \vdash t = t'$.*

Since we could not easily find an explicit statement of this in the literature, a self-contained proof is given in Appendix A.

3.6. **Example: typing the Ackermann-Péter function.** Let us take a moment to see an example of typing and reasoning within T. The Ackermann-Péter function $A : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is defined by the following equations:

$$\begin{array}{rcl} A(0,y) & := & y+1 \\ A(x+1,0) & := & A(x,1) \\ A(x+1,y+1) & := & A(x,A(x+1,y)) \end{array} \tag{3}$$

Formally, we may see A as being defined by induction on a lexicographical product order on $\mathbb{N} \times \mathbb{N}$. This function may duly be computed by a term of T by appealing to primitive recursion at type level 1. We first define a functional I by primitive recursion (at type N) satisfying:

$$\begin{array}{rcl} \mathrm{I}\, f\, 0 & = & f\, \underline{1} \\ \mathrm{I}\, f\, \mathsf{s} y & = & f\, (\mathrm{I}\, f\, y) \end{array} \tag{4}$$

Formally, I may be typed by the following derivation,

$$\dfrac{\dfrac{\dfrac{\overset{1}{\triangledown}}{\Rightarrow N} \quad \dfrac{\mathsf{id}}{N \Rightarrow N}}{\mathsf{L}\ \dfrac{\underline{N \to N} \Rightarrow N}{}} \qquad \dfrac{\dfrac{\mathsf{id}}{N \Rightarrow N} \quad \dfrac{\dfrac{\mathsf{id}}{N \Rightarrow N}}{\mathsf{wk}\ \dfrac{N, \underline{N} \Rightarrow N}{}}}{\mathsf{L}\ \dfrac{\underline{N \to N}, N \Rightarrow N}{}}}{\mathsf{rec}_N\ \dfrac{}{N \to N, \underline{N} \Rightarrow N}}$$

where principal types are underlined and red occurrences of N correspond to the same input (morally y in (4)). T proves the defining equations from (4) for I:

$$\begin{array}{rcll} \mathrm{I}\, f\, 0 & = & \mathsf{L}\, \underline{1}\, \mathsf{id}\, f & \text{by rec axioms} \\ & = & \mathsf{id}\, (f\, \underline{1}) & \text{by L axiom} \\ & = & f\, \underline{1} & \text{by id axiom} \\ \mathrm{I}\, f\, \mathsf{s} y & = & \mathsf{L}\, \mathsf{id}\, (\mathsf{wk}\, \mathsf{id})\, f\, (\mathrm{I}\, f\, y) & \text{by rec axioms} \\ & = & \mathsf{wk}\, \mathsf{id}\, (f\, (\mathsf{id}\, (\mathrm{I}\, f\, y)))\, (\mathrm{I}\, f\, y) & \text{by L axiom} \\ & = & \mathsf{id}\, (f\, (\mathsf{id}\, (\mathrm{I}\, f\, y))) & \text{by wk axiom} \\ & = & f\, (\mathsf{id}\, (\mathrm{I}\, f\, y)) & \text{by id axiom} \\ & = & f\, (\mathrm{I}\, f\, y) & \text{by id axiom} \end{array}$$

From here A is obtained by primitive recursion at type $N \to N$, satisfying:

$$\begin{array}{rcl} \mathrm{A}\, 0 & = & \mathsf{s} \\ \mathrm{A}\, \mathsf{s} x & = & \mathrm{I}\, (\mathrm{A}\, x) \end{array} \tag{5}$$

Formally, such A may be typed by the following derivation,

$$\mathsf{rec}_{N\to N}\,\dfrac{\mathsf{R}\,\dfrac{\mathsf{s}\,\dfrac{}{N \Rightarrow N}}{\Rightarrow \underline{N \to N}} \qquad \mathsf{wk}\,\dfrac{\mathsf{R}\,\dfrac{\overset{\mathrm{I}}{\triangledown}\; N \to N, N \Rightarrow N}{N \to N \Rightarrow \underline{N \to N}}}{\underline{N}, N \to N \Rightarrow N \to N}}{\underline{N} \Rightarrow N \to N}$$

where principal types are underlined.

Proposition 17. *T_1 proves the following equations:*

$$\begin{array}{rcl} \mathrm{A}\,0\,y & = & \mathsf{s}y \\ \mathrm{A}\,\mathsf{s}x\,0 & = & \mathrm{A}\,x\,\underline{1} \\ \mathrm{A}\,\mathsf{s}x\,\mathsf{s}y & = & \mathrm{A}\,x\,(\mathrm{A}\,\mathsf{s}x\,y) \end{array}$$

Proof. We have,

$$\begin{array}{rcll} \mathrm{A}\,0\,y & = & \mathsf{R}\,\mathsf{s}\,y & \text{by } \mathsf{rec} \text{ axioms} \\ & = & \mathsf{s}y & \text{by } \mathsf{R} \text{ axiom} \\ \mathrm{A}\,\mathsf{s}x\,y & = & \mathsf{wk}\,(\mathsf{R}\,\mathrm{I})\,x\,(\mathrm{A}\,x)\,y & \text{by } \mathsf{rec} \text{ axioms} \\ & = & \mathsf{R}\,\mathrm{I}\,(\mathrm{A}\,x)\,y & \text{by } \mathsf{wk} \text{ axiom} \\ & = & \mathrm{I}\,(\mathrm{A}\,x)\,y & \text{by } \mathsf{R} \text{ axiom} \quad (*) \end{array}$$

whence we have immediately $\mathrm{A}\,\mathsf{s}x\,0 \;=\; \mathrm{A}\,x\,\underline{1}$ by (4). For y non-zero, we have:

$$\begin{array}{rcll} \mathrm{A}\,\mathsf{s}x\,\mathsf{s}y & = & \mathrm{I}\,(\mathrm{A}\,x)\,\mathsf{s}y & \text{by } (*) \text{ above} \\ & = & \mathrm{A}\,x\,(\mathrm{I}\,(\mathrm{A}\,x)\,y) & \text{by } (4) \\ & = & \mathrm{A}\,x\,(\mathrm{A}\,\mathsf{s}x\,y) & \text{by } (*) \text{ above} \quad \square \end{array}$$

The recursors used to type A have level 1. This is not a coincidence, since primitive recursion at level 0 (i.e. on only natural numbers) computes just the primitive recursive functions:

Fact 18. *If $t : N^k \to N$ is a term of T_0 then $t^{\mathfrak{N}} : \mathbb{N}^k \to \mathbb{N}$ is primitive recursive.*

Note that, together with Proposition 14, this constitutes a proof that $I\Sigma_1$ (or RCA_0) well-defines just the primitive recursive functions.

4. A circular version of T

We will now move on to the main subject of study in this work: typing 'derivations' that are non-wellfounded and their corresponding notion of term. Let us henceforth write T^- for the restriction of T to the language without recursion combinators rec_τ.

Definition 19 (Conditional combinator)**.** We introduce a new typing rule cond for derivations, as well as corresponding axioms, in Figure 8. As before, the colouring of type occurrences above will become apparent soon.

Again, the interpretation of cond in the standard model $\mathfrak{N}$ is uniquely determined by the defining axioms of Figure 8. Throughout this section, we will work in the language of $T^- + \mathsf{cond}$, unless otherwise specified.

$$\mathsf{cond}\ \frac{\vec{\sigma} \Rightarrow \tau \quad \vec{\sigma}, N \Rightarrow \tau}{\vec{\sigma}, N \Rightarrow \tau} \qquad \begin{array}{rcl} \mathsf{cond}\ s\ t\ \vec{x}\ 0 & = & s\ \vec{x} \\ \mathsf{cond}\ s\ t\ \vec{x}\ \mathsf{s}y & = & t\ \vec{x}\ y \end{array}$$

FIGURE 8. Typing rule and axioms for cond combinators.

4.1. **Non-wellfounded 'terms' and 'derivations'. 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.

At the risk of confusion, we expand the range of the metavariables s, t, etc. to include coterms as well as terms, clarifying further only in the case of ambiguity. We adopt the same writing and bracketing conventions as for terms, where it is meaningful, e.g. writing $r\,s\,t$ for $(r \circ s) \circ t$.

We will not dwell much on arbitrary coterms, since we will only deal with those induced by our sequent style type system.

Definition 20 (Coderivations)**.** A **coderivation** is some possibly non-wellfounded 'derivation' built from the typing rules of $T^- + \mathsf{cond}$, in a locally correct manner. Formally, a coderivation is a possibly infinite labelled binary tree (of height $\leq \omega$) whose nodes are labelled by rule instances s.t. the premisses of a node (if any) match the conclusions of the node's respective children (if any).

We construe coderivations as coterms in the same way as we construed derivations as terms. Namely, we coinductively set:

- $\mathsf{r}\,\dfrac{\overset{s}{\vec{\sigma} \Rightarrow \sigma}}{\vec{\tau} \Rightarrow \tau}$ is the coterm $\mathsf{r}\,s$; and,
- $\mathsf{r}\,\dfrac{\overset{r}{\vec{\rho} \Rightarrow \rho} \quad \overset{s}{\vec{\sigma} \Rightarrow \sigma}}{\vec{\tau} \Rightarrow \tau}$ is the coterm $\mathsf{r}\ r\ s$.

Note that this association is continuous at the level of the underlying trees, so it is indeed well-defined.

Again overloading notation, we will write $t : \vec{\sigma} \Rightarrow \tau$ if t is a coderivation of the sequent $\vec{\sigma} \Rightarrow \tau$, and $t : \sigma$ if t is a coterm of type τ. If we need to distinguish t as a term then we will say so explicitly.

Note that the equational theory induced by Figures 4, 6 and 8 form a Kleene-Herbrand-Gödel style equational specification for coterms (cf., e.g., [Kle80]), now understanding the metavariables s, t etc. there to range over coterms. We may thus view coterms as *partial recursive functionals* in the standard model $\mathfrak{N}$ of the appropriate type. More formally:

Definition 21 (Interpretation of coterms as partial functionals)**.** We define a type structure $\mathfrak{N}_\perp$ by interpretations $\cdot^{\mathfrak{N}}_\perp$ and corresponding 'totally undefined functionals' $\perp_\sigma$ as follows:

- $\perp_N$ is just some fresh element $\perp$.
- $N^{\mathfrak{N}}_\perp$ is $\mathbb{N} \cup \{\perp\}$.
- $\perp_{\sigma\to\tau} : \sigma^{\mathfrak{N}}_\perp \to \tau^{\mathfrak{N}}_\perp$ by $a \mapsto \perp_\tau$, for any $a \in \sigma^{\mathfrak{N}}_\perp$.
- $(\sigma \to \tau)^{\mathfrak{N}}_\perp$ is the set of functions $f : \sigma^{\mathfrak{N}}_\perp \to \tau^{\mathfrak{N}}_\perp$ s.t. $f(\perp_\sigma) = \perp_\tau$.
- $=^{\mathfrak{N}}_\perp$ is just extensional equality (for each type).

A *partial functional of type* σ is just a function in $\sigma^{\mathfrak{N}}_\perp$. A *(total) functional of type* σ is just a partial functional f of type σ with $f(x) = \perp$ if and only if $x = \perp$.

We now define the interpretation of coterms in $\mathfrak{N}_\perp$ as follows:

- If $t : N$ then $t^{\mathfrak{N}}_\perp = n \in \mathbb{N}$ just if n is the unique interpretation of t (under the equations of Figures 4 and 8) in $\mathfrak{N}_\perp$. Otherwise $t^{\mathfrak{N}}_\perp$ is $\perp$.
- If $t : \sigma \to \tau$ and $a \in \sigma^{\mathfrak{N}}_\perp$ then $t^{\mathfrak{N}}_\perp(a) := (t\,a)^{\mathfrak{N}}_\perp$.[8]

Note that total functionals of type σ are just elements of $\sigma^{\mathfrak{N}}$, when restricted to non-$\perp$ arguments. Moreover, for any (finite) term t we have immediately that $t^{\mathfrak{N}}_\perp = t^{\mathfrak{N}}$, when restricted to non-$\perp$ arguments. In light of this, we shall henceforth unambiguously write $t^{\mathfrak{N}}$ rather than $t^{\mathfrak{N}}_\perp$, and simply write $\mathfrak{N}$ instead of $\mathfrak{N}_\perp$.

We shall omit here the finer details of this interpretation of coterms as partial functionals, since we will give a more formal (and, indeed, formal*ised*) treatment of 'regular' coterms and their computational interpretations in Section 6. We point the reader to the excellent book [LN15] for further details on models of (partial) (recursive) function(al)s.

Let us now consider some relevant examples of coderivations and coterms, 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 22 (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*.*[9] *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:*

(6)
$$\dfrac{f_0 \quad \vec{N} \Rightarrow N}{\mathsf{cond}\dfrac{\dfrac{f_1 \quad \vec{N} \Rightarrow N}{\mathsf{cond}\dfrac{\dfrac{f_2 \quad \vec{N} \Rightarrow N}{\mathsf{cond}\dfrac{\vdots}{N, \vec{N} \Rightarrow N}}}{N, \vec{N} \Rightarrow N}}}{N, \vec{N} \Rightarrow N}}$$

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*.*

[8] Here we are implicitly using parameters from the model.

[9] 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).

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 23 (Naïve simulation of primitive recursion)**.** *Terms of T may be interpreted as coterms in a straightforward manner. The only difficulty is the simulation of the* rec *combinators, which may be interpreted by coderivations as follows:*

$$(7)\qquad \mathsf{rec}\,\frac{\vec{\sigma}\Rightarrow\sigma \quad \vec{\sigma}, N, \sigma\Rightarrow\sigma}{\vec{\sigma}, N\Rightarrow\sigma} \quad\rightsquigarrow\quad \mathsf{cond}\,\frac{\vec{\sigma}\Rightarrow\sigma \qquad \mathsf{cut}\,\dfrac{\mathsf{cond}\,\dfrac{\vdots}{\vec{\sigma}, N\Rightarrow\sigma}\bullet \qquad \vec{\sigma}, N, \sigma\Rightarrow\sigma}{\vec{\sigma}, N\Rightarrow\sigma}}{\vec{\sigma}, N\Rightarrow\sigma}\bullet$$

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

Denoting the RHS of (7) *above as* rec'*, we can check that the two sides of* (7) *are indeed equivalent. Formally, we show* $\mathsf{rec}\,s\,t\,\vec{x}\,y \;=\; \mathsf{rec}\,s\,t\,\vec{x}\,y$ *by induction on* y*:*

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

Note that our reasoning here was completely syntactic, indeed only using axioms and rules from Figures 4, 7 and 8, (understanding metavariables s, t etc. in those figures to now range over coterms as well as terms). This is no coincidence, and the argument above will actually turn out to be a formal proof in our theory defined later, thus inducing equivalence of rec *and* rec' *in all models.*

4.2. **Regularity.** Until now, our coderivations and coterms were potentially non-uniform in structure and, as exemplified in Example 22, comprise a computational model of extreme expressivity. Naturally, within formal theories, we would prefer to manipulate only finitely presentable objects. To this end we will study a natural fragment in non-wellfounded proof theory:

Definition 24 (Regular coderivations and coterms)**.** A coderivation t is **regular** (or **circular**) if it has only finitely many distinct sub-coderivations. A regular coterm is similarly just one with finitely many distinct sub-coterms.

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'. When dealing with recursion-theoretic matters, we will implicitly assume such a finitary representation. Further details on such a formalised representation are given in Section 6.3.

Once again we have that regular coderivations are regular coterms, and conversely that closed regular coterms may be interpreted as regular coderivations (under cut-as-composition).

One example we have already seen of a regular coderivation is the RHS of (7). In fact, it turns out that the regular coterms constitute a sufficiently expressive programming language:

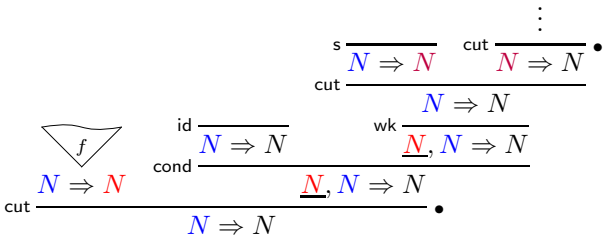
Proposition 25 (Turing completeness)**.** *The set of regular coderivations of type level 1 is Turing-complete,*[10] *i.e.* $\{t^{\mathfrak{N}} \mid t : N^k \Rightarrow N \text{ regular}\}$ *includes all partial recursive functions on* $\mathbb{N}$*.*

Proof. We have already seen in Example 23 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. For this, we observe that $\mu x(fx = 0)$ is equivalent to $H\,0$ where

$$H\,x \;=\; \mathsf{cond}\;(f\,x)\;x\;(H\,\mathsf{s}x) \tag{8}$$

Note that H may be interpreted by the following coderivation:

(9)



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 $\mathsf{s}x$.

Working in the standard model $\mathfrak{N}$, may show that H indeed satisfies (8) as follows. We have that:

$$\begin{array}{rcll} H\,x & = & \mathsf{cut}\,f\,(\mathsf{cond}\,\mathsf{id}\,(\mathsf{wk}\,(\mathsf{cut}\,\mathsf{s}\,H)))\,x & \text{by definition of } H \\ & = & \mathsf{cond}\,\mathsf{id}\,(\mathsf{wk}\,(\mathsf{cut}\,\mathsf{s}\,H))\,(f\,x)\,x & \text{by } \mathsf{cut} \text{ axiom} \end{array}$$

Now, we conduct a case analysis on the value of $f(x)$:

- If $f(x) = 0$, then we have $H\,x \;=\; \mathsf{id}\,x \;=\; x$, by the cond and id axioms, thus satisfying (8).
- If $f(x) = \mathsf{s}y$, for some y, then,

$$\begin{array}{rcll} H\,x & = & \mathsf{wk}\,(\mathsf{cut}\,\mathsf{s}\,H)\,y\,x & \text{by } \mathsf{cond} \text{ axioms} \\ & = & \mathsf{cut}\,\mathsf{s}\,H\,x & \text{by } \mathsf{wk} \text{ axiom} \\ & = & H\,\mathsf{s}x & \text{by } \mathsf{cut} \text{ axiom} \end{array}$$

  again satisfying (8). □

Remark 26 (Reasoning over partial functionals)**.** Note, again, that the reasoning above was entirely syntactic, using only axioms thus far presented in Figures 4, 6, 7 and 8. While the coderivation in (10) will not formally be a symbol of our eventual theory CT, the reasoning above hints at well-behaved extensions accommodating partially defined coterms.

[10]For a model of program execution, we may simply take the aforementioned Kleene-Herbrand-Gödel model with *equational derivability*, cf. [Kle80]. Note that this coincides with derivability by the axioms thus far presented.

Remark 27 (Turing completeness at level 0)**.** Note that the argument above required coderivations including only occurrences of N. This means that the set of type 0 regular coderivations are already a Turing-complete programming language, under derivability via Figures 4, 6, 7 and 8.

4.3. **The progressing criterion.** Despite regular coderivations being finitely presentable, they do not necessarily denote totally defined functionals in the standard model $\mathfrak{N}$, cf. Proposition 25, contrary to the norm for terms in formal theories. In this work we will consider coderivations satisfying a standard 'termination criterion' in non-wellfounded proof theory. First, let us recall some standard structural proof theoretic concepts about (co)derivations.

Definition 28 (Immediate ancestry)**.** Let t be a (co)derivation. A type occurrence σ^1 is an **immediate ancestor**[11] 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 3, 5 or 8. 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.

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 29 (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 8). 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 8.)

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 30 (Extensional completeness at type 1, revisited)**.** *Recalling Example 22, note that the infinite branch marked $\cdots$ in* (6) *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 arbitraryily far from the root of* (6)*.*

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 31 (Primitive recursion, revisited)**.** *Recalling Example 23, notice that the RHS of* (7) *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.*

Example 32 (Turing completeness, revisited)**.** *Recalling the proof of Proposition 25, notice that the coderivation given for H in* (9) *is* not *progressing: the*

[11]This terminology is standard in proof theory, e.g. as in [Bus98].

only infinite branch loops on $\bullet$ and immediate ancestry, as indicated by the colouring, admits no thread along the $\bullet$-loop. This is no coincidence, as it turns out that the progressing criterion suffices for coderivations to denote total *functionals in the standard model $\mathfrak{N}$, as we will show in the next subsection.*

One of the most appealing features of the progressing criterion is that, while being rather expressive and admitting many natural programs, e.g. as we will see in Section 4.6, it remains effective (for regular coderivations) thanks to well known arguments in automaton theory:

Fact 33 (Folklore)**.** *It is decidable whether a regular coderivation is progressing.*

This well-known result (see, e.g., [DHL06b] for an exposition for a similar circular system) follows from the fact that the progressing criterion is equivalent to the universality of a Büchi automaton of size determined by the (finite) representation of the input coderivation. This problem is decidable in polynomial space, though the correctness of this algorithm requires nontrivial infinitary combinatorics, as formally demonstrated in [KMPS19b]. Nonetheless, a non-uniform version of this problem is formalisable in the weakest of the big-five theories of reverse mathematics:

Proposition 34 ([Das20])**.** *For any regular progressing coderivation t, RCA_0 proves that t is progressing.*

As noted in that work, the above result cannot be strengthened to a uniform one unless RCA_0 (and so PRA) is inconsistent, by a reduction to Gödel-incompletness.

4.4. **Progressing coterms denote total functionals.** As outlined in Definition 21, coderivations denote partial functionals in the standard model $\mathfrak{N}$. In fact, the partial functionals induced by progressing coderivations are indeed totally defined, by adapting well-known infinite descent arguments in non-wellfounded proof theory:

Proposition 35. *If $t : \vec{\sigma} \Rightarrow \tau$ is a progressing coderivation, then $t^{\mathfrak{N}}$ is a well-defined total functional in $(\vec{\sigma} \to \tau)^{\mathfrak{N}}$.*

The idea behind this result is to, by contradiction, assume a non-terminating 'run' of a progressing coderivation, and thence extract an infintely decreasing sequence of natural numbers from a progressing thread, contradicting the well-ordering property. We stop short of giving an explicit 'operational semantics' here, being beyond the scope of this work. Rather, let us simply note that the totally defined functionals are closed under composition by typing rules (since typing rules are constants interpreted as totally defined functionals themselves). Contrapositively this means that if a coderivation is interpreted by a non-total functional, then so is one of its immediate sub-coderivations.

Proof of Proposition 35. Suppose otherwise and let $\vec{a} \in \vec{\sigma}^{\mathfrak{N}}$ be inputs on which $t^{\mathfrak{N}}$ is not well-defined, i.e. $t^{\mathfrak{N}}(\vec{a}) = \bot$. We may thus inductively construct an infinite branch $(t_i : \vec{\sigma}_i \Rightarrow \tau_i)_{i\in\omega}$ and associated inputs $(\vec{a}_i \in \vec{\sigma}_i^{\mathfrak{N}})$ s.t. $t_i^{\mathfrak{N}}(\vec{a}_i) = \bot$ as follows:

- $t_0 = t$ and $\vec{a}_0 = \vec{a}$.
- If t_i concludes with a wk, ex or cntr step then t_{i+1} is the only immediate sub-coderivation and $\vec{a}_{i+1}$ is just $\vec{a}_i$ with the appropriate deletion, switch or duplication of arguments.

- (t_i cannot conclude with a nullary step id, 0 or s, by assumption that t_i is non-total.)
- If t_i concludes with a cut step, as typeset in Figure 3, then t_{i+1} is the left sub-coderivation if it is not totally defined on inputs $\vec{a}_i$; otherwise t_{i+1} is the right sub-coderivation and $\vec{a}_{i+1} = (\vec{a}, a)$, for some $a \in \sigma^{\mathfrak{N}}$ s.t. $t^{\mathfrak{N}}_{i+1}(\vec{a}_i, a) = \bot$.
- If t_i concludes with a L step, as typeset in Figure 3, then t_{i+1} is the left sub-coderivation if it is not totally defined on inputs $\vec{a}_i$; otherwise t_{i+1} is the right sub-coderivation and $\vec{a}_{i+1} = (\vec{a}_i, a)$, for some $a \in \sigma^{\mathfrak{N}}$ s.t. $t^{\mathfrak{N}}_{i+1}(\vec{a}_i, a) = \bot$.
- If t_i concludes with a R step, as typeset in Figure 3, then t_{i+1} is the only immediate sub-coderivation and $\vec{a}_{i+1} = (\vec{a}_i, a)$, for some $a \in \sigma^{\mathfrak{N}}$ s.t. $t^{\mathfrak{N}}_{i+1}(\vec{a}_i, a) = \bot$.
- If t_i concludes with a cond step and $\vec{a}_i = (\vec{a}'_i, n)$, then t_{i+1} is the left sub-coderivation if $n = 0$ and $\vec{a}_{i+1} = \vec{a}'_i$; otherwise, if $n = m + 1$, t_{i+1} is the right sub-coderivation and $\vec{a}_{i+1} = (\vec{a}'_i, m)$.

Now, notice that, since t is progressing, we must have some progressing thread $(N^i)_{i\geq k}$ along some tail $(t_i)_{i\geq k}$. Writing n_i for the input in $\vec{a}_i$ corresponding to N^i, notice that $(n_i)_{i\geq k}$ is a non-increasing sequence of natural numbers, by construction of t_i and $\vec{a}_i$. Moreover, we have that $n_{i+1} < n_i$ whenever N^i is principal for a cond step, so for infinitely many $i \geq k$ by definition of a progressing thread. Thus $(n_i)_{i\geq k}$ has no least element, contradicting the well-ordering property. □

4.5. **The simply typed theory CT and its fragments.** We are finally ready to give the definition of our circular version of System T.

Definition 36 (Circular version of T)**.** The language of CT extends contains every regular progressing coderivation of $T^- + \mathsf{cond}$ as a symbol. We identify 'terms' of this language with coterms in the obvious way, and call them (regular) **progressing coterms**. CT itself is a STT axiomatised by the schemata from Figures 4, 7 and 8, now interpreting the metavariables s, t etc. there as ranging over coterms.

The aim of this work is to compare fragments of CT and fragments of T delineated by type level. In light of Proposition 16, the following definition gives natural circular counterparts of the fragments T_n of T:

Definition 37 (Type level restricted fragments of CT)**.** CT_n is the fragment of CT restricted to the language containing only coderivations where all types occurring have level $\leq n$. CT_n still has constant symbols for each individual constant of $T^- + \mathsf{cond}$.

Notice that, despite the fact that coderivations of CT_n may type only level $n+1$ functionals, progressing coterms are closed under application and so, by definition of a STT, CT_n admits 'terms' of arbitrary type by composing with the constants and variables of $T^- + \mathsf{cond}$. In particular we still have combinatory completeness, as usual. In what follows, however, it will usually suffice to only consider the coderivations when proving properties of CT_n, the generalisation to progressing coterms following by closure under application.

4.6. **Example: Ackermann-Péter, revisited.** Let us revisit the example of the Ackermann-Péter function from Section 3.6. Despite the fact that type level 1

recursion is required to type it in T (cf. Fact 18), A may be circularly typed in CT by a coderivation A using only the base type N:[12]

(10)
$$\dfrac{\dfrac{\dfrac{\dfrac{}{N \Rightarrow N}\,\mathsf{s}}{\underline{N}, N \Rightarrow N}\,\mathsf{wk}}{N, \underline{N}, N \Rightarrow N}\,\mathsf{cond} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\Rightarrow N}\,1 \quad \vdots(1)}{N, N \Rightarrow N}\,\bullet}{N \Rightarrow N}\,\mathsf{cut}}{N, \underline{N}, N \Rightarrow N}\,\mathsf{wk} \qquad \dfrac{\dfrac{\vdots(2)}{N, N \Rightarrow N}\,\bullet \quad \dfrac{\vdots(3)}{N, N \Rightarrow N}\,\bullet}{N, N, N \Rightarrow N}\,\mathsf{cut}}{N, N, \underline{N} \Rightarrow N}\,\mathsf{cond}}{\dfrac{\underline{N}, N, N \Rightarrow N}{\underline{N}, N \Rightarrow N}\,\bullet}\,\mathsf{cntr}$$

As usual, the occurrences of $\bullet$ above indicate roots of identical coderivations, and we have indicated three distinct threads, coloured red, blue and orange. Note that the purple N prefixes both the red and the blue thread. Finally note that the red N thread progresses on every visit to (1) and (3), while the orange N thread progresses on every visit to (2).

Proposition 38. A *is progressing and regular, and so is a symbol of* CT_0.

Proof. To show that A is progressing, we conduct a case analysis on an infinite branch B, based on which of the simple loops (1), (2) and (3) are traversed infinitely often:

- B hits only (1) infinitely often. Then there is a progressing thread along the red N.
- B hits only (2) infinitely often. Then eventually there is a progressing thread along the orange N.
- B hits only (3) infinitely often. Then eventually there is a progressing thread along the red N.
- B hits only (1) and (2) infinitely often. Then eventually there is a progressing thread along the red N on iterations of (1) (on which it progresses) and along the blue N on iterations of (2) (on which it is constant).
- B hits only (1) and (3) infinitely often. Then eventually there is a progressing thread along the red N, which progresses on any iteration of (1) or (3).
- B hits only (2) and (3) infinitely often. Then eventually there is a progressing thread along the blue N on iterations of (2) (on which it is constant) and along the red N on iterations of (3) (on which it progresses).
- B hits all of (1), (2) and (3) infinitely often. Then there is a progressing thread along the red N on iterations of (1) and (3) (on which it progresses) and the blue N on iterations of (2) (on which it is constant).

Clearly A is regular and contains only occurrences of N, so A is indeed a symbol of CT_0. □

In fact, we may also show that CT_0 proves the defining equations of A from (3):

[12]For convenience we have implemented some branching rules as context-splitting, namely the cut steps. Formally, there are implicit wk steps that are not indicated, a convention that we will henceforth adopt for the sake of easing legibility.

Proposition 39. *CT_0 proves the following equations:*

$$\begin{array}{rcl} \mathrm{A}\,0\,y & = & \mathsf{s}y \\ \mathrm{A}\,\mathsf{s}x\,0 & = & \mathrm{A}\,x\,\underline{1} \\ \mathrm{A}\,\mathsf{s}x\,\mathsf{s}y & = & \mathrm{A}\,x\,(\mathrm{A}\,\mathsf{s}x\,y) \end{array}$$

Proof. Writing A_0 and A_1 for the left and right coderivations, respectively, composed by the lowermost cond step in (10), we have:

$$\begin{array}{rcll} \mathrm{A}\,x\,y & = & \mathsf{cntr}\,(\mathsf{cond}\,\mathrm{A}_0\,\mathrm{A}_1)\,x\,y & \text{by definition of A} \\ & = & \mathsf{cond}\,\mathrm{A}_0\,\mathrm{A}_1\,x\,x\,y & \text{by } \mathsf{cond} \text{ axioms} \end{array} \tag{11}$$

From here we obtain the first equation of (3), in CT_0:

$$\begin{array}{rcll} \mathrm{A}\,0\,y & = & \mathrm{A}_0\,x\,y & \text{by (11) above and } \mathsf{cond} \text{ axioms} \\ & = & \mathsf{wk}\,\mathsf{s}\,x\,y & \text{by definition of } \mathrm{A}_0 \\ & = & \mathsf{s}y & \text{by } \mathsf{wk} \text{ axiom} \end{array}$$

Now writing A_{10} and A_{11} for the left and right coderivations, respectively, composed by the uppermost cond step typeset in (10), we have:

$$\begin{array}{rcll} \mathrm{A}\,\mathsf{s}x\,y & = & \mathrm{A}_1\,x\,\mathsf{s}x\,y & \text{by (11) above and } \mathsf{cond} \text{ axioms} \\ & = & \mathsf{cond}\,\mathrm{A}_{10}\,\mathrm{A}_{11}\,x\,\mathsf{s}x\,y & \text{by definition of } \mathrm{A}_1 \end{array} \tag{12}$$

From here we obtain the second and third equations of (3), in CT_0:

$$\begin{array}{rcll} \mathrm{A}\,\mathsf{s}x\,0 & = & \mathrm{A}_{10}\,x\,\mathsf{s}x & \text{by (12) above and } \mathsf{cond} \text{ axioms} \\ & = & \mathsf{wk}\,(\mathsf{cut}\,(1\,\mathrm{A}))\,x\,\mathsf{s}x & \text{by definition of } \mathrm{A}_{10} \\ & = & \mathsf{cut}\,(1\,\mathrm{A})\,x & \text{by } \mathsf{wk} \text{ axiom} \\ & = & \mathrm{A}\,x\,\underline{1} & \text{by } \mathsf{cut} \text{ axiom} \\ \mathrm{A}\,\mathsf{s}x\,\mathsf{s}y & = & \mathrm{A}_{11}\,x\,\mathsf{s}x\,y & \text{by (12) above and } \mathsf{cond} \text{ axioms} \\ & = & \mathsf{cut}\,(\mathrm{A}\,\mathrm{A})\,x\,\mathsf{s}x\,y & \text{by definition of } \mathrm{A}_{11} \\ & = & \mathrm{A}\,x\,(\mathrm{A}\,\mathsf{s}x\,y) & \text{by } \mathsf{cut} \text{ axiom} \quad \square \end{array}$$

Now, let us revisit Section 3.6, where we gave a T term for the Ackermann-Péter function based on type 1 recursion. Calling that term A', temporarily, we now have that A' and A satisfy the same defining equations from (3). Working in a combined theory, we may thus deduce their equivalence via a nested induction.

Proposition 40. $CT_0, \mathsf{rec}_1 \vdash \mathrm{A}\,x\,y = \mathrm{A}'x\,y$.

Proof. Working inside $CT_0 + \mathsf{rec}_1$, we show $\mathrm{A}\,x = \mathrm{A}'x$ by induction on x. We have,

$$\begin{array}{rcll} \mathrm{A}\,0\,y & = & \mathsf{s}y & \text{by Proposition 39} \\ & = & \mathrm{A}'0\,y & \text{by Proposition 17} \end{array}$$

so $\mathrm{A}\,0 = \mathrm{A}'0$ by ER. For the inductive step, we show that $\mathrm{A}\,\mathsf{s}x\,y = \mathrm{A}'\mathsf{s}x\,y$ by a sub-induction on y:

$$\begin{array}{rcll} \mathrm{A}\,\mathsf{s}x\,0 & = & \mathrm{A}\,x\,\underline{1} & \text{by Proposition 39} \\ & = & \mathrm{A}'x\,\underline{1} & \text{by main inductive hypothesis} \\ & = & \mathrm{A}'\mathsf{s}x\,0 & \text{by Proposition 17} \\ \mathrm{A}\,\mathsf{s}x\,\mathsf{s}y & = & \mathrm{A}\,x\,(\mathrm{A}\,\mathsf{s}x\,y) & \text{by Proposition 39} \\ & = & \mathrm{A}'x\,(\mathrm{A}\,\mathsf{s}x\,y) & \text{by main induction hypothesis} \\ & = & \mathrm{A}'x\,(\mathrm{A}'\mathsf{s}x\,y) & \text{by sub-induction hypothesis} \\ & = & \mathrm{A}'\mathsf{s}x\,\mathsf{s}y & \text{by Proposition 17} \end{array}$$

Thus we have $\mathrm{A}\,\mathsf{s}x = \mathrm{A}'\mathsf{s}x$ by ER, proving the main inductive step as required. $\square$

As we mentioned before, general extensionality is not strictly necessary for many of our results, and this is in particular the case for the result above. We could have also proceeded under *weak extensionality* by an instance of Π_1-induction $(\forall y(\mathrm{A}\,x\,y = \mathrm{A}'x\,y))$. Standard witnessing theorems then reduce this to an instance of quantifier-free induction, but such a development is beyond the scope of this work.

5. CT simulates T, more succinctly

In this section we give a simulation of T_{n+1} terms of level $\leq n+1$ into CT_n, provably satisfying the same equational theory, thus improving on the naive simulation of primitive recursion from Examples 23 and 31. This matches similar results from the setting of arithmetic, [Das20], and the exposition is entirely proof theoretic.

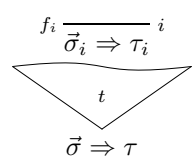
The main goal is to show the following result:

Theorem 41. *If $t : \sigma$ is a term of T_{n+1} with $\mathsf{lev}(\sigma) \leq n+1$ then there is a CT_n coderivation $t' : \sigma$ s.t.:*

$$CT_n, \mathsf{rec}_{n+1} \vdash t = t' \tag{13}$$

The idea is to rely on the partial normalisation result, Proposition 16, to work with a normal form of T derivations, and then translate into CT coderivations under a constructive realisation of the deduction theorem (for typing derivations).

5.1. **Derivations with 'oracles'.** We consider (co)derivations with fresh intial sequents of the form $f\,\dfrac{}{\sigma_1, \ldots, \sigma_n \Rightarrow \tau}$, denoting a functional f of type $\sigma_1 \to \cdots \to \sigma_n \to \tau$, as expected. We write,

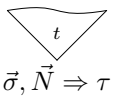


for a (co)derivation t of $\vec{\sigma} \Rightarrow \tau$ with initial sequents among $f_i\,\dfrac{}{\vec{\sigma}_i \Rightarrow \tau_i}\,i$, with i varying over some fixed range. We distinguish f_i from variable symbols, since coderivations until now are closed coterms; instead it is more pertinent to think of them as some fresh constant 'oracle' symbols. We thus use metavariables f, g, etc. for these new initial sequents. The interpretation of such (co)derivations into the standard model $\mathfrak{N}$ is as expected, with (co)derivations now computing (partial) functionals with respect to oracles for each f_i.

5.2. **Constructive realisation of the deduction theorem.** We describe how to 'realise' a version of the *deduction theorem* for typing derivations in order to lower type level. The key feature of this translation is that recursion is simulated by a cyclic derivation in a succinct way, in terms of abstration complexity.

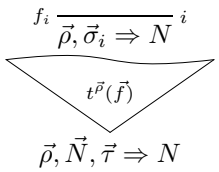
As notation throughout this section, if $\vec{f} = f_1, \ldots, f_n$ then we may simply write $(\vec{f}\,\vec{x})$ for $(f_1\,\vec{x})\,(f_2\,\vec{x})\,\ldots\,(f_n\,\vec{x})$, to lighten the syntax. Our key intermediate result is the following lemma:

Lemma 42. *Let $t : \vec{\sigma}, \vec{N} \Rightarrow \tau$ be a T_{n+1} derivation, i.e.,*



with all Ns indicated, s.t. all types occurring in t have level $\leq n+1$.

Write $\tau = \vec{\tau} \to N$ and $\sigma_i = \vec{\sigma}_i \to N$. For each $\vec{\rho}$ of levels $\leq n$ there is a CT_n coderivation $t^{\vec{\rho}}(\vec{f}) : \vec{\rho}, \vec{N}, \vec{\tau} \Rightarrow N$ using initial sequents $f_i \dfrac{}{\vec{\rho}, \vec{\sigma}_i \Rightarrow N}\, i$, i.e.,



such that,

$$CT_n, \mathsf{rec}_{n+1} \vdash t^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,\vec{z} \;=\; t\,(\vec{f}\,\vec{x})\,\vec{y}\,\vec{z} \tag{14}$$

Moreover, for any j, there is a ρ_j-thread from the ρ_j in the conclusion of $t^{\vec{\rho}}(\vec{f})$ to the ρ_j in any occurrence of the initial sequent f_i.

Before giving the proof let us set up some further notation to lighten the exposition as much as possible.

- We shall sometimes suppress the initial sequent arguments of a coderivation when it is unambiguous, e.g. writing $t^{\vec{\rho}}$ instead of $t^{\vec{\rho}}(\vec{f})$ etc. We will only do this when the initial sequents, or approrpriate substituted (co)derivations, are explicitly typeset.
- We shall write r^* for the reflexive transitive closure of a rule r.
- As in the statement of the lemma, we shall typically assume that a type, say, τ has the form $\vec{\tau} \to N$ and so on.
- Variables $\vec{x}, x$ will typically correspond to types $\vec{\rho}, \rho$, variables $\vec{y}, y$ to $\vec{N}, N$, and $\vec{z}\,z$ to $\vec{\tau}, \tau$.

Proof of Lemma 42. We proceed by induction on the structure of t, and refer to an inductive hypothesis for a smaller derivation s by $IH(s)$.

If t is the initial sequent $\mathsf{id}\dfrac{}{\sigma \Rightarrow \sigma}$ then:

- If $\sigma = N$, then $t^{\vec{\rho}}$ is:

$$\mathsf{wk}\dfrac{\mathsf{id}\dfrac{}{N \Rightarrow N}}{\vec{\rho}, N \Rightarrow N}$$

  There are no new initial sequents so also no thread condition to check. To verify (14), we have:

$$\begin{array}{rcll} t^{\vec{\rho}}\,\vec{x}\,y & = & \mathsf{wk}^*\,\mathsf{id}\,\vec{x}\,y & \text{by definition of } t^{\vec{\rho}} \\ & = & \mathsf{id}\,y & \text{by } \mathsf{wk} \text{ axioms} \\ & = & t\,y & \text{by definition of } t \end{array}$$

- Otherwise $t^{\vec{\rho}}$ is just:

$$f\,\dfrac{}{\vec{\rho}, \vec{\sigma} \Rightarrow N}$$

The required threading property is immediate, and to verify (14) we have:

$$\begin{array}{rcll} t^{\vec{\rho}}(f)\,\vec{x}\,\vec{y} & = & f\,\vec{x}\,\vec{y} & \text{by definition of } t^{\vec{\rho}}. \\ & = & \mathsf{id}\,(f\,\vec{x})\,\vec{y} & \text{by } \mathsf{id} \text{ axiom} \\ & = & t\,(f\,\vec{x})\,\vec{y} & \text{by definition of } t \end{array}$$

If t is the initial sequent $0\,\dfrac{}{\Rightarrow N}$, then $t^{\vec{\rho}}$ is:

$$\mathsf{wk}\,\dfrac{0\,\dfrac{}{\Rightarrow N}}{\vec{\rho} \Rightarrow N}$$

There are no new initial sequents, so no threading property to check. To verify (14) we have:

$$\begin{array}{rcll} t^{\vec{\rho}}\,\vec{x} & = & \mathsf{wk}^*\,0\,\vec{x} & \text{by definition of } t^{\vec{\rho}} \\ & = & 0 & \text{by } \mathsf{wk} \text{ axioms} \\ & = & t & \text{by definition of } t \end{array}$$

If t is the initial sequent $\mathsf{s}\,\dfrac{}{N \Rightarrow N}$ then $t^{\vec{\rho}}$ is:

$$\mathsf{wk}\,\dfrac{\mathsf{s}\,\dfrac{}{N \Rightarrow N}}{\vec{\rho}, N \Rightarrow N}$$

There are no new initial sequents, so no threading property to check. To verify (14) we have:

$$\begin{array}{rcll} t^{\vec{\rho}}\,\vec{x}\,y & = & \mathsf{wk}^*\,\mathsf{s}\,\vec{x}\,y & \text{by definition of } t^{\vec{\rho}} \\ & = & \mathsf{s}\,y & \text{by } \mathsf{wk} \text{ axioms} \\ & = & t\,y & \text{by definition of } t \end{array}$$

If t concludes with a weakening step,

$$\mathsf{wk}\,\dfrac{\overset{s}{\triangledown}}{\vec{\sigma}, \vec{N} \Rightarrow \tau}{\vec{\sigma}, \pi, \vec{N} \Rightarrow \tau}$$

then:

- If $\pi = N$ then we define $t^{\vec{\rho}}(\vec{f})$ by just commuting with the weakening step:

$$\mathsf{wk}\,\dfrac{\overset{s^{\vec{\rho}}(\vec{f})}{\triangledown}}{\vec{\rho}, \vec{\sigma}, \vec{N}, \vec{\tau} \Rightarrow N}{\vec{\rho}, \vec{\sigma}, N, \vec{N}, \vec{\tau} \Rightarrow N}$$

The threading property is readily obtained from the inductive hypothesis, and we verify (14) as follows:

$$\begin{array}{rcll} t^{\vec{\rho}}(\vec{f})\,\vec{x}\,y\,\vec{y}\,\vec{z} & = & \mathsf{wk}\,s^{\vec{\rho}}(\vec{f})\,\vec{x}\,y\,\vec{y}\,\vec{z} & \text{by definition of } t^{\vec{\rho}} \\ & = & s^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,\vec{z} & \text{by } \mathsf{wk} \text{ axiom} \\ & = & s\,(\vec{f}\,\vec{x})\,\vec{y}\,\vec{z} & \text{by } IH(s) \\ & = & \mathsf{wk}\,s\,(\vec{f}\,\vec{x})\,y\,\vec{y}\,\vec{z} & \text{by } \mathsf{wk} \text{ axiom} \\ & = & t\,(\vec{f}\,\vec{x})\,y\,\vec{y}\,\vec{z} & \text{by definition of } t \end{array}$$

- Otherwise, $t^{\vec{\rho}}(\vec{f},g)$ is simply just $s^{\vec{\rho}}(\vec{f})$ (the initial sequents $f:\vec{\pi}\Rightarrow N$ are never used), and the required properties are inherited directly from the inductive hypothesis.

If t concludes with a contraction step,

$$\mathsf{cntr}\,\dfrac{\overset{s}{\vec{\sigma},\pi,\pi,\vec{N}\Rightarrow\tau}}{\vec{\sigma},\pi,\vec{N}\Rightarrow\tau}$$

then:

- If $\pi = N$ then we define $t^{\vec{\rho}}(\vec{f})$ by just commuting with the contraction step:

$$\mathsf{cntr}\,\dfrac{\overset{s^{\vec{\rho}}(\vec{f})}{\vec{\rho},\vec{\sigma},N,N,\vec{N},\vec{\tau}\Rightarrow N}}{\vec{\rho},\vec{\sigma},N,\vec{N},\vec{\tau}\Rightarrow N}$$

The threading property is readily obtained from the inductive hypothesis, and we verify (14) as follows:

$$\begin{array}{rcll} t^{\vec{\rho}}(\vec{f})\,\vec{x}\,y\,\vec{y}\,\vec{z} & = & \mathsf{cntr}\,s^{\vec{\rho}}(\vec{f})\,\vec{x}\,y\,\vec{y}\,\vec{z} & \text{by definition of } t^{\vec{\rho}} \\ & = & s^{\vec{\rho}}(\vec{f})\,\vec{x}\,y\,y\,\vec{y}\,\vec{z} & \text{by } \mathsf{cntr} \text{ axiom} \\ & = & s\,(\vec{f}\,\vec{x})\,y\,y\,\vec{y}\,\vec{z} & \text{by } IH(s) \\ & = & \mathsf{cntr}\,s\,(\vec{f}\,\vec{x})\,y\,\vec{y}\,\vec{z} & \text{by } \mathsf{cntr} \text{ axiom} \\ & = & t\,(\vec{f}\,\vec{x})\,y\,\vec{y}\,\vec{z} & \text{by definition of } t \end{array}$$

- Otherwise, $t^{\vec{\rho}}(\vec{f},g)$ is just $s^{\vec{\rho}}(\vec{f},g,g)$ and the threading property is immediate from the inductive hypothesis. (14) is also easily verified:

$$\begin{array}{rcll} t^{\vec{\rho}}(\vec{f},g)\,\vec{x}\,\vec{y}\,\vec{z} & = & s^{\vec{\rho}}(\vec{f},g,g)\,\vec{x}\,\vec{y}\,\vec{z} & \text{by definition of } t \\ & = & s\,(\vec{f}\,\vec{x})\,(g\,\vec{x})\,(g\,\vec{x})\,\vec{y}\,\vec{z} & \text{by } IH(s) \\ & = & \mathsf{cntr}\,s\,(\vec{f}\,\vec{x})\,(g\,\vec{x})\,\vec{y}\,\vec{z} & \text{by } \mathsf{cntr} \text{ axiom} \\ & = & t\,(\vec{f}\,\vec{x})\,(g\,\vec{x})\,\vec{y}\,\vec{z} & \text{by definition of } t \end{array}$$

If t concludes with a cut step,

$$\mathsf{cut}\,\dfrac{\overset{r}{\vec{\sigma},\vec{N}\Rightarrow\pi}\quad\overset{s}{\vec{\sigma},\pi,\vec{N}\Rightarrow\tau}}{\vec{\sigma},\vec{N}\Rightarrow\tau}$$

then:

- If $\pi = N$ then we define $t^{\vec{\rho}}(\vec{f})$ by just commuting with the cut step:

$$\mathsf{cut}_N \dfrac{\overset{r^{\vec{\rho}}(\vec{f})}{\vec{\rho}, \vec{N} \Rightarrow N} \quad \overset{s^{\vec{\rho}}(\vec{f})}{\vec{\rho}, N, \vec{N}, \vec{\tau} \Rightarrow N}}{\vec{\rho}, \vec{N}, \vec{\tau} \Rightarrow N}$$

  The threading property is readily obtained from the inductive hypotheses and we verify (14) as follows:

$$\begin{array}{rcll} t^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,\vec{z} & = & s^{\vec{\rho}}(\vec{f})\,\vec{x}\,(r^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y})\,\vec{y}\,\vec{z} & \text{by } \mathsf{cut} \text{ axiom} \\ & = & s\,(\vec{f}\,\vec{x})\,(r\,(\vec{f}\,\vec{x})\,\vec{y})\,\vec{y}\,\vec{z} & \text{by } IH(r) \text{ and } IH(s) \\ & = & t\,(\vec{f}\,\vec{x})\,\vec{y}\,\vec{z} & \text{by } \mathsf{cut} \text{ axiom} \end{array}$$

- Otherwise, we define $t^{\vec{\rho}}(\vec{f})$ as:

$$\mathsf{cntr} \dfrac{\mathsf{wk} \dfrac{f_i \dfrac{}{\vec{\rho}, \vec{\sigma}_i \Rightarrow N}\, i}{\vec{\rho}, \vec{N}, \vec{\sigma}_i \Rightarrow N} \quad \overset{r^{\vec{\rho}}(\vec{f})}{\vec{\rho}, \vec{N}, \vec{\pi} \Rightarrow N}}{\vec{\rho}, \vec{N}, \vec{N}, \vec{\tau} \Rightarrow N} \; s^{\vec{\rho}, \vec{N}}$$

  The threading property is readily obtained from the inductive hypotheses, and we verify (14) as follows:

$$\begin{array}{rcll} t^{\vec{\rho}}\,\vec{x}\,\vec{y}\,\vec{z} & = & s^{\vec{\rho}, \vec{N}}(\mathsf{wk}^*\,\vec{f}, r^{\vec{\rho}}(\vec{f}))\,\vec{x}\,\vec{y}\,\vec{y}\,\vec{z} & \text{by } \mathsf{cntr} \text{ axioms} \\ & = & s\,(\mathsf{wk}^*\,\vec{f}\,\vec{x}\,\vec{y})\,(r^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y})\,\vec{y}\,\vec{z} & \text{by } IH(s) \\ & = & s\,(\mathsf{wk}^*\,\vec{f}\,\vec{x}\,\vec{y})\,(r\,(\vec{f}\,\vec{x})\,\vec{y})\,\vec{y}\,\vec{z} & \text{by } IH(r) \text{ and ER} \\ & = & s\,(\vec{f}\,\vec{x})\,(r\,(\vec{f}\,\vec{x})\,\vec{y})\,\vec{y}\,\vec{z} & \text{by } \mathsf{wk} \text{ axioms and ER} \\ & = & t\,(\vec{f}\,\vec{x})\,\vec{y}\,\vec{z} & \text{by definition of } t \end{array}$$

If t concludes with a right-implication step,

$$\mathsf{R} \dfrac{\overset{s}{\vec{\sigma}, \pi, \vec{N} \Rightarrow \tau}}{\vec{\sigma}, \vec{N} \Rightarrow \pi \to \tau}$$

then:

- If $\pi = N$ then $t^{\vec{\rho}}(\vec{f})$ is just:

$$\mathsf{ex} \dfrac{\overset{s^{\vec{\rho}}(\vec{f})}{\vec{\rho}, N, \vec{N}, \vec{\tau} \Rightarrow N}}{\vec{\rho}, \vec{N}, N, \vec{\tau} \Rightarrow N}$$

The threading property is readily obtained from the inductive hypothesis and (14) is easily verified:

$$\begin{array}{rcll} t^{\vec\rho}(\vec f)\,\vec x\,\vec y\,y\,\vec z & = & s^{\vec\rho}(\vec f)\,\vec x\,y\,\vec y\,\vec z & \text{by } \mathsf{ex} \text{ axioms} \\ & = & s\,(\vec f\,\vec x)\,y\,\vec y\,\vec z & \text{by } IH(s) \\ & = & t\,(\vec f\,\vec x)\,\vec y\,y\,\vec z & \text{by } \mathsf{R} \text{ axiom} \end{array}$$

- Otherwise $\mathsf{lev}(\pi) \le n$ and we define $t^{\vec\rho}(\vec f)$ as:

$$\dfrac{\dfrac{\dfrac{}{\vec\rho, \vec\sigma_i \Rightarrow N}\,{\scriptstyle f_i,\,i}}{\vec\rho, \pi, \vec\sigma_i \Rightarrow N}\,\mathsf{wk} \qquad \dfrac{\dfrac{}{\pi, \vec\pi \Rightarrow N}\,\mathsf{app}}{\vec\rho, \pi, \vec\pi \Rightarrow N}\,\mathsf{wk}}{\vec\rho, \pi, \vec N, \vec\tau \Rightarrow N}\,s^{\vec\rho,\pi} \qquad \dfrac{\vec\rho, \pi, \vec N, \vec\tau \Rightarrow N}{\vec\rho, \vec N, \pi, \vec\tau \Rightarrow N}\,\mathsf{ex}$$

where app is a simple (finite) derivation satisfying:

$$\mathsf{app}\,z\,\vec w \;=\; z\,\vec w \tag{15}$$

The threading property is readily obtained from the inductive hypothesis, and we verify (14) as follows:

$$\begin{array}{rcll} t^{\vec\rho}\,\vec x\,\vec y\,z\,\vec z & = & s^{\vec\rho,\pi}(\mathsf{wk}\,\vec f, \mathsf{wk}^*\mathsf{app})\,\vec x\,z\,\vec y\,\vec z & \text{by } \mathsf{ex} \text{ axioms} \\ & = & s\,(\mathsf{wk}\,\vec f\,\vec x\,z)\,(\mathsf{wk}^*\mathsf{app}\,\vec x\,z)\,\vec y\,\vec z & \text{by } IH(s) \\ & = & s\,(\vec f\,\vec x)\,(\mathsf{app}\,z)\,\vec y\,\vec z & \text{by } \mathsf{wk} \text{ axioms and ER} \\ & = & s\,(\vec f\,\vec x)\,z\,\vec y\,\vec z & \text{by (15) and ER} \\ & = & t\,(\vec f\,\vec x)\,\vec y\,z\,\vec z & \text{by } \mathsf{R} \text{ axiom} \end{array}$$

If t concludes with a left-implication step,

$$\mathsf{L}\,\dfrac{\overset{r}{\triangledown}\quad \vec\sigma, \vec N \Rightarrow \tau' \qquad \vec\sigma, \pi, \vec N \Rightarrow \tau \quad \overset{s}{\triangledown}}{\vec\sigma, \tau' \to \pi, \vec N \Rightarrow \tau}$$

then:

- If $\pi = N$ then we define $t^{\vec\rho}(\vec f, g)$ as:

$$\mathsf{cut}\,\dfrac{\mathsf{R}\,\dfrac{\overset{r^{\vec\rho}(\vec f)}{\triangledown}\quad \vec\rho, \vec N, \vec\tau' \Rightarrow N}{\vec\rho, \vec N \Rightarrow \tau'} \qquad \mathsf{cut}\,\dfrac{{\scriptstyle g}\;\dfrac{}{\vec\rho, \tau' \Rightarrow N} \qquad \overset{s^{\vec\rho}(\vec f)}{\triangledown}\quad \vec\rho, N, \vec N, \vec\tau \Rightarrow N}{\vec\rho, \tau', \vec N, \vec\tau \Rightarrow N}}{\vec\rho, \vec N, \vec\tau \Rightarrow N}$$

The threading property is readily obtained from the inductive hypotheses and we verify (14) as follows:

$$\begin{array}{rcll}
t^{\vec{\rho}}(\vec{f},g)\,\vec{x}\,\vec{y}\,\vec{z} & = & \mathsf{cut}\,g\,s^{\vec{\rho}}(\vec{f})\,\vec{x}\,(\mathsf{R}^*\,r^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y})\,\vec{y}\,\vec{z} & \text{by } \mathsf{cut} \text{ axiom} \\
& = & s^{\vec{\rho}}(\vec{f})\,\vec{x}\,(g\,\vec{x}\,(\mathsf{R}^*\,r^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}))\,\vec{y}\,\vec{z} & \text{by } \mathsf{cut} \text{ axiom} \\
& = & s^{\vec{\rho}}(\vec{f})\,\vec{x}\,(g\,\vec{x}\,(r^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}))\,\vec{y}\,\vec{z} & \text{by } \mathsf{R} \text{ axioms and ER} \\
& = & s\,(\vec{f}\,\vec{x})\,(g\,\vec{x}\,(r^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}))\,\vec{y}\,\vec{z} & \text{by } IH(s) \\
& = & s\,(\vec{f}\,\vec{x})\,(g\,\vec{x}\,(r\,(\vec{f}\,\vec{x})\,\vec{y}))\,\vec{y}\,\vec{z} & \text{by } IH(r) \text{ and ER} \\
& = & t\,(\vec{f}\,\vec{x})\,(\vec{g}\,\vec{x})\,\vec{y}\,\vec{z} & \text{by } \mathsf{L} \text{ axiom}
\end{array}$$

- Otherwise we define $t^{\vec{\rho}}(\vec{f},g)$ as:

$$\dfrac{\dfrac{\dfrac{}{\vec{\rho},\vec{\sigma}_i \Rightarrow N}\,f_i}{\vec{\rho},\vec{N},\vec{\sigma}_i \Rightarrow N}\,\mathsf{wk}\,i \qquad \dfrac{\dfrac{\overset{r^{\vec{\rho}}(\vec{f})}{\vec{\rho},\vec{N},\vec{\tau}' \Rightarrow N}}{\vec{\rho},\vec{N} \Rightarrow \tau'}\,\mathsf{R} \qquad \dfrac{\dfrac{}{\vec{\rho},\tau',\vec{\pi} \Rightarrow N}\,g}{\vec{\rho},\vec{N},\tau',\vec{\pi} \Rightarrow N}\,\mathsf{wk}}{\vec{\rho},\vec{N},\vec{\pi} \Rightarrow N}\,\mathsf{cut}}{\vec{\rho},\vec{N},\vec{N},\vec{\tau} \Rightarrow N}\,\mathsf{cntr}$$

(with $s^{\vec{\rho},\vec{N}}$ applied to the premise $\vec{\rho},\vec{N},\vec{N},\vec{\tau} \Rightarrow N$)

The threading property is readily obtained from the inductive hypotheses and we verify (14) as follows:

$$\begin{array}{rcll}
t^{\vec{\rho}}(\vec{f},g)\,\vec{x}\,\vec{y}\,\vec{z} & = & s^{\vec{\rho},\vec{N}}(\mathsf{wk}^*\vec{f},\mathsf{cut}(\mathsf{R}^*r^{\vec{\rho}}(\vec{f}))(\mathsf{wk}^*g))\,\vec{x}\,\vec{y}\,\vec{y}\,\vec{z} & \text{by } \mathsf{cntr} \text{ axioms} \\
& = & s\,(\mathsf{wk}^*\vec{f}\,\vec{x}\,\vec{y})\,(\mathsf{cut}(\mathsf{R}^*r^{\vec{\rho}}(\vec{f}))(\mathsf{wk}^*g)\vec{x}\,\vec{y})\,\vec{y}\,\vec{z} & \text{by } IH(s) \\
& = & s\,(\vec{f}\,\vec{x})\,(\mathsf{cut}(\mathsf{R}^*r^{\vec{\rho}}(\vec{f}))(\mathsf{wk}^*g)\vec{x}\,\vec{y})\,\vec{y}\,\vec{z} & \text{by } \mathsf{wk} \text{ axioms and ER} \\
& = & s\,(\vec{f}\,\vec{x})(\mathsf{wk}^*g\,\vec{x}\,\vec{y}\,(\mathsf{R}^*r^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}))\,\vec{y}\,\vec{z} & \text{by } \mathsf{cut} \text{ axiom} \\
& = & s\,(\vec{f}\,\vec{x})(g\,\vec{x}\,(\mathsf{R}^*r^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}))\,\vec{y}\,\vec{z} & \text{by } \mathsf{wk} \text{ axioms and ER} \\
& = & s\,(\vec{f}\,\vec{x})(g\,\vec{x}\,(r^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}))\,\vec{y}\,\vec{z} & \text{by } \mathsf{R} \text{ axioms and ER} \\
& = & s\,(\vec{f}\,\vec{x})\,(g\,\vec{x}\,(r\,(\vec{f}\,\vec{x})\,\vec{y}))\,\vec{y}\,\vec{z} & \text{by } IH(r) \text{ and ER} \\
& = & t\,(\vec{f}\,\vec{x})\,(g\,\vec{x})\,\vec{y}\,\vec{z} & \text{by } \mathsf{L} \text{ axiom}
\end{array}$$

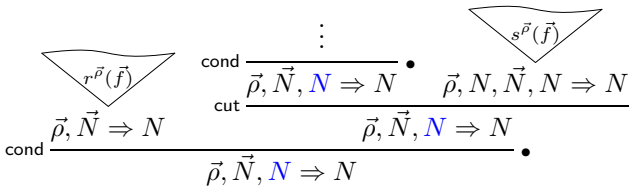
Finally, if t concludes with a recursion step,

$$\dfrac{\overset{r}{\vec{\sigma},\vec{N} \Rightarrow \tau} \qquad \overset{s}{\vec{\sigma},\tau,\vec{N},N \Rightarrow \tau}}{\vec{\sigma},\vec{N},N \Rightarrow \tau}\,\mathsf{rec}$$

then:

- If $\tau = N$ (so $\vec{\tau}$ is empty) then we define $t^{\vec{\rho}}$ similarly to before in Example 23,

(16)



where • marks roots of identical sub-coderivations. The threading property is readily obtained from the inductive hypotheses. For progressiveness, any infinite branch either just loops on • indefinitely, in which case there is a progressing thread along the blue N, or is eventually in just $r^{\vec{\rho}}(\vec{f})$ or $s^{\vec{\rho}}(\vec{f})$, in which case there is a progressing thread by the inductive hypotheses. To verify (14) we shall show,

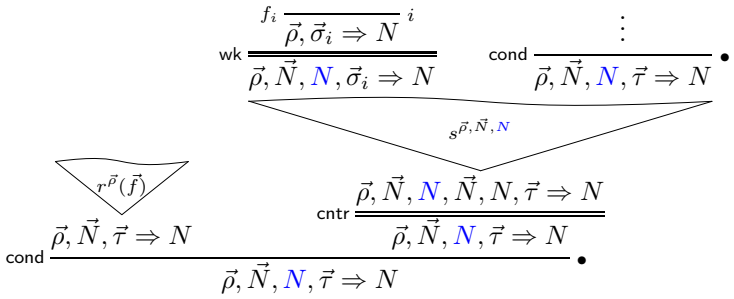
(17) $$t^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,y \;=\; t\,(\vec{f}\,\vec{x})\,\vec{y}\,y$$

by (object-level) induction on y:

$$\begin{array}{rcll}
t^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,0 & = & r^{\vec{f}}(\vec{f})\,\vec{x}\,\vec{y} & \text{by } \mathsf{cond} \text{ axiom} \\
& = & r\,(\vec{f}\,\vec{x})\,\vec{y} & \text{by } IH(r) \\
& = & t\,(\vec{f}\,\vec{x})\,\vec{y}\,0 & \text{by } \mathsf{rec} \text{ axioms} \\
t^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,\mathsf{s}y & = & \mathsf{cut}\,t^{\vec{\rho}}(\vec{f})\,s^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,y & \text{by } \mathsf{cond} \text{ axioms} \\
& = & s^{\vec{\rho}}(\vec{f})\,\vec{x}\,(t^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,y)\,\vec{y}\,y & \text{by } \mathsf{cut} \text{ axiom} \\
& = & s\,(\vec{f}\,\vec{x})\,(t^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,y)\,\vec{y}\,y & \text{by } IH(s) \\
& = & s\,(\vec{f}\,\vec{x})\,(t\,(\vec{f}\,\vec{x})\,\vec{y}\,y)\,\vec{y}\,y & \text{by inductive hypothesis (17)} \\
& = & t\,(\vec{f}\,\vec{x})\,\vec{y}\,\mathsf{s}y & \text{by } \mathsf{rec} \text{ axioms}
\end{array}$$

- Otherwise we define $t^{\vec{\rho}}(\vec{f})$ to be,

(18)



where • marks roots of identical sub-coderivations. The threading property is readily obtained from the inductive hypotheses. For progressiveness, notice that any infinite branch that hits • infinitely often will have a progressing thread along the blue N, thanks to the threading property from the inductive hypothesis for s. Any other infinite branch is eventually in $r^{\vec{\rho}}(\vec{f})$ or $s^{\vec{\rho},\vec{N},N}$, so progressiveness follows from the inductive hypotheses.

To verify (14) we show that,

(19) $$t^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,y\,\vec{z} \;=\; t\,(\vec{f}\,\vec{x})\,\vec{y}\,y\,\vec{z}$$

by (object-level) induction on y:

$$\begin{array}{rcll}
t^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,0\,\vec{z} & = & r^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,\vec{z} & \text{by } \mathsf{cond} \text{ axioms} \\
& = & r\,(\vec{f}\,\vec{x})\,\vec{y}\,\vec{z} & \text{by } IH(r) \\
& = & t\,(\vec{f}\,\vec{x})\,\vec{y}\,0\,\vec{z} & \text{by } \mathsf{rec} \text{ axioms} \\
t^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,\mathsf{s}y\,\vec{z} & = & \mathsf{cntr}^*\, s^{\vec{\rho},\vec{N},N}(\mathsf{wk}^*\vec{f}, t^{\vec{\rho}}(\vec{f}))\,\vec{x}\,\vec{y}\,y\,\vec{z} & \text{by } \mathsf{cond} \text{ axioms} \\
& = & s^{\vec{\rho},\vec{N},N}(\mathsf{wk}^*\vec{f}, t^{\vec{\rho}}(\vec{f}))\,\vec{x}\,\vec{y}\,y\,\vec{y}\,y\,\vec{z} & \text{by } \mathsf{cntr} \text{ axioms} \\
& = & s\,(\mathsf{wk}^*\vec{f}\,\vec{x}\,\vec{y}\,y)\,(t^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,y)\,\vec{y}\,y\,\vec{z} & \text{by } IH(s) \\
& = & s\,(\vec{f}\,\vec{x})\,(t^{\vec{\rho}}(\vec{f})\,\vec{x}\,\vec{y}\,y)\,\vec{y}\,y\,\vec{z} & \text{by } \mathsf{wk} \text{ axioms and ER} \\
& = & s\,(\vec{f}\,\vec{x})\,(t\,(\vec{f}\,\vec{x})\,\vec{y}\,y)\,\vec{y}\,y\,\vec{z} & \text{by inductive hypothesis (19)} \\
& = & t\,(\vec{f}\,\vec{x})\,\vec{y}\,\mathsf{s}y\,\vec{z} & \text{by } \mathsf{rec} \text{ axioms} \quad \square
\end{array}$$

5.3. CT_n **simulates** T_{n+1}**.** We are now ready to prove the main result of this section, which essentially boils down to an instance of the main Lemma in the previous subsection.

Proof of Theorem 41. Without loss of generality assume t is a derivation of $\Rightarrow \sigma$ and apply Lemma 42 with $\vec{\rho}$ empty, to obtain a coderivation $t^{\varnothing} : \vec{\sigma} \Rightarrow N$ s.t.:

$$CT_n, \mathsf{rec}_{n+1} \vdash\ t\,\vec{x} = t^{\varnothing}\,\vec{x} \tag{20}$$

Note that $t^{\varnothing}$ uses no new initial sequents since the antecedent of the conclusion of t is empty. Now we define t' as:

$$\mathsf{R}\,\dfrac{\overset{t^{\varnothing}}{\vec{\sigma} \Rightarrow N}}{\Rightarrow \sigma}$$

We verify (13) as follows, working inside the theory:

$$\begin{array}{rll}
 & t\,\vec{x} = t^{\varnothing}\,\vec{x} & \text{by (20)} \\
\therefore & t\,\vec{x} = \mathsf{R}^* t^{\varnothing}\,\vec{x} & \text{by } \mathsf{R} \text{ axioms} \\
\therefore & t\,\vec{x} = t'\vec{x} & \text{by definition of } t' \\
\therefore & t = t' & \text{by ER} \quad \square
\end{array}$$

6. Coterm-based models of T and CT

Our ultimate goal is to establish a converse to the main result of the previous section, which we shall demonstrate in the next two sections. Before that we need to introduce some type structures arising from our formulation of CT. Ultimately, we will reduce the simulation of CT_n in T_{n+1} to the provability in an appropriate arithmetic theory that certain structures are models of CT .

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* and the *hereditarily effective* operations of finite type (see, e.g., [LN15]).

Since regular coterms form a Turing-complete programming language (cf. Proposition 25), the domains of our structures will simply be classes of regular coterms. Viewed as Kleene-Herbrand-Gödel equational specifications (see, e.g., [Kle80]), note that the corresponding notion of *computation* is subsumed by provable equality of

coterms by rules and axioms of CT. In fact, we will distill from CT a suitable fragment of provable equality, namely that induced by the respective rewriting system. This will eventually give rise to an intensional model, whence an extensional one can be obtained by, as usual, taking the extensional collapse.

6.1. **Reduction and conversion of coterms.** The reduction relation $\rightsquigarrow$ on coterms is defined by orienting all the equations in Figures 4, 6 and 8 left-to-right and taking closure under substitution and contexts. Formally:

Definition 43 (Reduction and conversion)**.** $\rightsquigarrow$ is the least relation on (co)terms satisfying the reductions in Figure 9, where the types of each rule label are as indicated in Figures 3, 5 and 8,[13] and closed 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. We sometimes write $\rightsquigarrow_\sigma$ for the restriction of $\rightsquigarrow$ to coterms of type σ, and $\approx_\sigma$ for the restriction of $\approx$ to coterms of type σ.

$$
\begin{array}{rcl}
\mathsf{id}\ x & \rightsquigarrow & x \\
\mathsf{ex}\ t\ \vec{x}\ x\ y\ \vec{y} & \rightsquigarrow & t\ \vec{x}\ y\ x\ \vec{y} \\
\mathsf{wk}\ t\ \vec{x}\ x & \rightsquigarrow & t\ \vec{x} \\
\mathsf{cntr}\ t\ \vec{x}\ x & \rightsquigarrow & t\ \vec{x}\ x\ x \\
\mathsf{cut}\ s\ t\ \vec{x} & \rightsquigarrow & t\ \vec{x}\ (s\ \vec{x}) \\
\mathsf{L}\ s\ t\ \vec{x}\ y & \rightsquigarrow & t\ \vec{x}\ (y\ (r\ \vec{x})) \\
\mathsf{R}\ t\ \vec{x}\ x & \rightsquigarrow & t\ \vec{x}\ x \\
\mathsf{rec}\ s\ t\ \vec{x}\ 0 & \rightsquigarrow & s\ \vec{x} \\
\mathsf{rec}\ s\ t\ \vec{x}\ \mathsf{s}y & \rightsquigarrow & t\ \vec{x}\ (\mathsf{rec}\ s\ t\ \vec{x}\ y) \\
\mathsf{cond}\ s\ t\ \vec{x}\ 0 & \rightsquigarrow & s\ \vec{x} \\
\mathsf{cond}\ s\ t\ \vec{x}\ \mathsf{s}y & \rightsquigarrow & t\ \vec{x}\ y
\end{array}
$$

FIGURE 9. Reduction rules for (co)terms

We shall address metamathematical matters w.r.t. formalising reduction and reduction sequences shortly, but first let us examine some basic mathematical properties of reduction.

6.2. **On normality and numerality.** The term model for T, due to Tait [Tai67], may be obtained by proving normalisation and confluence of the rewrite system in Figure 9 for *terms*, and taking the (unique) normal forms of closed terms as the domain of the type structure. Crucial to this approach is the property that the only closed normal forms of terms of type N are the numerals, which allows induction in T to be reduced to induction at the meta-level.

Since regular coterms are Turing-complete, cf. Proposition 25,[14] one cannot hope for such a normalisation result, and so our models will be obtained by restricting to normalising coterms. Normalisation for progressing coterms at type N follows by showing that they are indeed elements of the type structure. We will address

[13] Again there is no reason, other than for ease of legibility, that we write terms variables s, t etc. for the premiss inputs and variables x, y etc. for the antecedent inputs. Under substitution, it makes no difference.

[14] Notice from that proof that Turing-completeness is retained under $\rightsquigarrow$ as a model of execution.

confluence shortly, adapting known proofs to the non-wellfounded setting. However, we point out that the final point, that closed normal forms of type N are just the numerals, fails for coterms:

Remark 44 (Non-numeral normal coterms of type N)**.** There are closed $\rightsquigarrow$-normal coterms of type N that are not numerals, e.g. the coterm H with,

$$H = \mathsf{cond}\,0\,1\,H$$

that returns 0 or 1, depending on whether it itself is 0 or non-zero. Clearly H is undefined in the standard model, though may be consistently interpreted by 0 or 1 in extensions of it. Note that H is even a *regular* counterexample to numerality of closed $\rightsquigarrow$-normal coterms, and so the $\rightsquigarrow$-normal regular coterms will be too large a domain for the type structures we later define. For this reason, our type structures will restrict the interpretation of N to (classes of) coterms that normalise to a numeral.

6.3. **Formalising reduction sequences of regular coterms.** Before continuing, let us make some comments about our arithmetisation of reduction sequences. First and foremost, to avoid unnecessary technicalities, all the coterms we consider later will be regular, and so can be coded by natural numbers. Thus all quantification over them is strictly first-order. In fact, many of our results go through in a more general setting since finite reduction sequences may be coded by finite data, but such a treatment is beyond the scope of this work. In what follows we shall be rather brief, outlining only the main ideas and proof methods behind the results we need.

While equality for arbitrary coterms is Π^0_1, we should justify that it is recursive for regular coterms. Namely, we should argue that we may decide if two *presentations* of regular coterms (as finite labelled graphs) represent the same coterm (as an infinite labelled tree), i.e. are *bisimilar*.

For $v \in \{0,1\}^*$ and a coterm t, construed as a labelled binary tree, let us temporarily write t_v for the sub-coterm of t rooted at position v, and $t[v]$ for the rule instance at position v. For a regular presentation G of a coterm, a finite rooted labelled graph, write $G(v)$ for the node of G reached by following the word v along its edge relation. Note that all of these notions are provably computable in t, G and v in RCA_0.

Proposition 45. *Bisimilarity of rooted finite labelled directed graphs is provably recursive in* RCA_0.

Proof. For recursivity, just blindly search for a bisimulation relation between the two vertex sets (in exponential time). Now we need to show that two finite graphs G and H are bisimilar if and only if there is a bisimulation between them:

- Suppose R is a bisimulation between G and H. We prove by induction on node position $v \in \{0,1\}^*$ that $(G(v), H(v)) \in R$. From here, by definition of a bisimulation, any two nodes related by R have the same label, and so the unfoldings of G and H are equal.
- Suppose G and H have the same unfolding. We inductively construct a bisimulation R from the root by continually adding $(G(v), H(v))$ to R. We

terminate when we hit a pair that is already in R, which will happen in at most $|G||H|+1$ steps, by the *finite* pigeonhole principle.[15] □

Corollary 46. *Equality for regular coterms is provably* Δ^0_1 *in* RCA_0.

Now, we better show that reduction preserves certain properties of coterms, not least regularity. More generally, since we will work with a certain class of regular coterms, we should make sure that this class is closed under reduction.

Proposition 47 (RCA_0)**.** *If* $s \rightsquigarrow t$ *then* t *is finitely composed of sub-coterms of* s*:*

$$\exists \text{ a finite term } r(x_1,\dots,x_n).\, \exists \langle v_1,\dots,v_n\rangle.\, t = r(s_{v_1},\dots,s_{v_n}) \tag{21}$$

This result holds for arbitrary coterms and, indeed, the existential quantifiers may be explicitly witnessed by primitive recursive functions in terms of s and t. We can take $s_{v_1},\dots,s_{v_n}$ to include the coderivations indicated in the contractum of a reduction in Figure 9, 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.[16]

As an immediate consequence of the above proposition we have, for arbitrary coterms:

Corollary 48 (RCA_0)**.** *Suppose* $s \rightsquigarrow t$.

(1) *If* s *has only finitely many variable occurrences then so does* t.
(2) *If* s *has only finitely many redexes then so does* t.
(3) *If* s *is regular then so is* t.
(4) *If* s *is progressing then so is* t.

Restricting now to regular coterms, we obtain the analogue of Proposition 47 for reduction *sequences* by Σ^0_1-induction:

Proposition 49 (RCA_0)**.** *If* s *is regular and* $s \rightsquigarrow^* t$*, then* (21).

Again, we could also deduce a similar result for arbitrary coterms, specifying reduction sequences as finite lists of redex positions, but formally developing this is beyond the scope of this work. As expected, we obtain the same properties of Corollary 48 for regular reduction sequences:

Corollary 50 (RCA_0)**.** *Suppose* s *is regular and* $s \rightsquigarrow^* t$*. Then the conditions* (1)*,* (2)*,* (3) *and* (4) *from Corollary 48 hold.*

6.4. **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.

We will need to formalise our argument within RCA_0 for later results, comprising an additional contribution to the literature, bounding the logical strength of confluence for finitary and certain infinitary rewrite systems operating with regular

[15] Note that, while the usual *infinite* pigeonhole principle is not provable in RCA_0, it is easy to see that the finite one is provable in RCA_0 (and even weaker theories), by a straightforward Σ^0_1 induction.

[16] Note that there is no need for weak König's lemma here, since combs may be explicitly proved to induce finite trees in RCA_0. In any case, WKL_0 is arithmetically conservative over RCA_0.

coterms. To facilitate this formalisation, all the coterms we consider in this section will be finite applications of regular coderivations to variables and constants ('farcs' for short) which, by Proposition 49, are closed under reduction sequences, provably in RCA_0. farcs are not necessarily progressing but, since coderivations are closed and have no redexes or variables, farcs have at most finitely many variables or redexes, provably in RCA_0 (each application operation can add at most one variable and redex).

The main goal of this subsection is to prove the following:

Theorem 51 (Church-Rosser, RCA_0)**.** *Let $t : \sigma$ be a farc. 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. In particular we aim to find a relation $\rhd$ such that:

(1) $s \rightsquigarrow t \implies s \rhd t \implies s \rightsquigarrow^* t$; and,
(2) $\rhd$ satisfies the 'diamond' property: if $t_0 \lhd t \rhd t_1$ then there is t' such that $t_0 \rhd t' \lhd t_1$.

The first property ensures that the reflexive transitive closure of $\rightsquigarrow$ and $\rhd$ coincide, i.e. $\rightsquigarrow^* = \rhd^*$. The second property then ensures confluence of $\rightsquigarrow^*$

Since coterms are infinite (and, moreover, non-wellfounded), we must carry out our argument without appeal to induction on *term structure*, ruling out standard arguments due to Tait and Martin-Löf (cf., e.g., [HS86]). Approaches of Takahashi in [Tak95] and others that rely on *complete developments* could potentially be adapted, since we are working with farcs which have only finitely many redexes. However, instead, we perform an argument by induction on reduction *length*, as in, e.g., [Pfe92], which also seems to require less machinery from rewriting theory.

Definition 52 (Parallel reduction)**.** We define the relation $\rhd$ on farcs as follows:

(1) $t \rhd t$ for any farc t.
(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 $s\,t \rhd s'\,t'$.

Note that we really do seem to require clause (1), $t \rhd t$, for arbitrary farcs t, not just variables and constants, since we cannot finitely derive the former from the latter.

Proposition 53 (RCA_0)**.** $s \rightsquigarrow t \implies s \rhd t \implies s \rightsquigarrow^* t$.

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 54 (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 55 (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'.*

In what follows, we may write $d : s \rightsquigarrow^* t$ or $d : s \rhd t$ or even $d : s \rhd^* t$ to indicate that d is a (finite) derivation witnessing the respective relation.

Proof. We show $d : s \rhd s' \implies s[t/x] \rhd s'[t'/x]$ by Σ^0_1-induction on the structure of the derivation $d : s \rhd s'$.

If d is obtained by just (1), i.e. $s' = s$, then we show $s[t/x] \rhd s[t'/x]$ by a subinduction on the maximum depth of an x-occurrence in s. (Recall that farcs have only finitely many variable occurrences.)

- if s is just the variable x then $s[t/x] \rhd s'[t'/x]$ by assumption of $t \rhd t'$.
- if s is just a variable $y \neq x$ then $s[t/x] = y \rhd y = s'[t'/x]$ by (1).
- if s is just a constant symbol r then $s[t/x] = \mathsf{r} \rhd \mathsf{r} = s'[t'/x]$ by (1).
- if $s = s_0 s_1$ then all the (finitely many) occurrences of x in s_0 and s_1 have lower depth, so we have $s_0[t/x] \rhd s_0[t'/x]$ and $s_1[t/x] \rhd s_1[t'/x]$ by the inductive hypothesis, whence $s[t/x] \rhd s[t'/x]$ by (4).

If d ends by (2) we can write $s = \mathsf{r}\,\vec{s}$ and $s' = r(\vec{s}')$ s.t. $\mathsf{r}\,\vec{s} \rightsquigarrow r(\vec{s})$ is a r-reduction and each $s_i \rhd s'_i$. By the inductive hypothesis we have that $s_i[t/x] \rhd s'_i[t'/x]$. We also have that $\mathsf{r}\,\vec{s}[t/x] \rightsquigarrow r(\vec{s}[t/x])$ is an r-reduction, where r is variable-free (by inspection of the reduction rules). Thus we have $s[t/x] = \mathsf{r}\,\vec{s}[t/x] \rhd r(\vec{s}'[t/x]) = s'[t/x]$ by (2).

(The argument when d ends by (3) is similar to that of (2).)

If d ends by (4) we can write $s = s_0 s_1$ and $s' = s'_0 s'_1$ s.t. $s_0 \rhd s'_0$ and $s_1 \rhd s'_1$. By the inductive hypothesis we have $s_0[t/x] \rhd s'_0[t/x]$ and $s_1[t/x] \rhd s'_1[t/x]$, whence $s[t/x] \rhd s'[t/x]$ by (4). □

Notice that Proposition 53 now follows immediately, by simply instantiating the Lemma above with $s = s'$ to deduce context-closure of $\rhd$.

We can now turn to proving the required 'diamond property' for $\rhd$:

Lemma 56 (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:

Proposition 57 (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 $d_i < d$.*

Proof. We proceed by Σ^0_1-induction on the length of $\vec{s}$. If $\vec{s}$ is empty, then only clause (1) applies to $\mathsf{r}\,\vec{s}$, so we may set $\vec{t}$ empty too.

Suppose $\mathsf{r}\,\vec{s}\,s \rhd t$, and there is no redex involving r. Then only clause (4) applies to $\mathsf{r}\,\vec{s}\,s$, so we have $t = t_0 t_1$ and some $d_0 : \mathsf{r}\,\vec{s} \rhd t_0$ and $d_1 : s \rhd t_1$ with $d_0, d_1 < d$. By inductive hypothesis there are $\vec{t_0}$ s.t. $t_0 = \mathsf{r}\,\vec{t_0}$ and some $d_{0i} : s_i \rhd t_{0i}$ with $d_{0i} < d_0$. Thus we have $t = \mathsf{r}\,\vec{t_0}\,t_1$ and $d_{0i} : s_i \rhd t_{0i}$ and $d_1 : s \rhd t_1$ with $d_{0i}, d_1 < d$, as required. □

We are now ready to prove the diamond property for $\rhd$:

Proof of Lemma 56. We proceed by simultaneous induction on the structure of the reductions $t \rhd t_0$ and $t \rhd t_1$. Formally we show,

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

by Σ^0_1-induction on $\min(|d_0|, |d_1|)$.

- If $t_0 = s$ (i.e. d_0 is just (1)) then we have $t_0 \rhd t_1 \lhd t_1$, by assumption and reflexivity, and we are done.
- Similarly if $t_1 = s$ (i.e. d_1 is just (1)), so we may henceforth assume that $t_0 \neq t \neq t_1$, and d_0 and d_1 conclude with one of the clauses (2),(3) or (4).

- If d_0 and d_1 both end by clause (2), then we have $s = \mathsf{r}\,\vec{s}$ and $t_0 = r(\vec{s}_0)$ and $t_1 = r(\vec{s}_1)$, for some reduction step $\mathsf{r}\,\vec{s} \rightsquigarrow r(\vec{s})$ and some $\vec{s}_0, \vec{s}_1$ s.t. $s_i \rhd s_{0i}$ and $s_i \rhd s_{1i}$. By the inductive hypothesis we have $\vec{s}'$ s.t. $s_{0i} \rhd s'_i$ and $s_{1i} \rhd s'_i$. Thus by repeatedly applying Lemma 55 we have $t_0 = r(\vec{s}_0) \rhd r(\vec{s}')$ and $t_1 = r(\vec{s}_1) \rhd r(\vec{s}')$, so we may set $s' = r(\vec{s}')$.
- (similarly if d_0 and d_1 end with clause (3))
- (it is not possible for one to end with clause (2) and the other to end by (3), since in all cases precisely one reduction step applies at the head.)
- If d_0 ends by clause (2) and d_1 ends by clause (4), then from d_0 we have r, r s.t. $s = \mathsf{r}\,\vec{s}\,u \rightsquigarrow r(\vec{s}, u)$ and some $\vec{s}_0, u_0$ with $s_i \rhd s_{0i}$ and $u \rhd u_0$ with $t_0 = r(\vec{s}_0, u_0)$. From d_1 we further have some t', u_1 s.t. $t_1 = t'u_1$ and $\mathsf{r}\,\vec{s} \rhd t'$ and $u \rhd u_1$. By Proposition 57 we have some $\vec{s}_1$ s.t. $t' = \mathsf{r}\,\vec{s}_1$ and smaller derivations of $s_i \rhd s_{1i}$. By the inductive hypothesis we have $\vec{s}', u'$ s.t. $s_{0i} \rhd s'_i \lhd s_{1i}$ and $u_0 \rhd u' \lhd u_1$. Thus, by repeatedly applying Lemma 55 we have that $t_0 = r(\vec{s}_0, u_0) \rhd r(\vec{s}', u')$, and by (2) we have $t_1 = \mathsf{r}\,\vec{s}_1\,u_1 \rhd r(\vec{s}', u')$, so it suffices to set $t' = r(\vec{s}', u')$.
- (the case when d_0 ends by clause (2) and d_1 ends by clause (4) is symmetric to the one above)
- If both d_0 and d_1 end by clause (4), then we have $s = s_0 s_1$, $t_0 = r_0 r_1$, $t_1 = u_0 u_1$ s.t. $r_0 \lhd s_0 \rhd u_0$ and $r_1 \lhd s_1 \rhd u_1$. By inductive hypothesis we have s'_0 and s'_1 s.t. $r_0 \rhd s'_0 \lhd u_0$ and $r_1 \rhd s'_1 \lhd u_1$. Thus we have that $t_0 = r_0 r_1 \rhd s'_0 s'_1 \lhd u_0 u_1$ by (4), so we may set $s' = s'_0 s'_1$.

□

Proposition 58 (Weighted CR for $\rhd$, RCA_0)**.** *If $t_0 \lhd^m t \rhd^n t_1$ then there is some t' with $t_0 \rhd^n t' \lhd^m t_1$.*

Proof. We show,

$$(d_0 : t \rhd^m t_0 \text{ and } d_1 : t \rhd^n t_1) \implies \exists t'(t_0 \rhd^n t' \text{ and } d'_1 : t_1 \rhd^m t')$$

by Σ^0_1-induction on $m = |d_0|$.

- If $m = 0$ then $t_0 = t$ and we may simply set $t' = t_1$, whence we have that $t_0 = t \rhd^n t'$ by assumption and $t' = t_1 \lhd^0 t_1$.
- Now, suppose $t_0 \lhd t'_0 \lhd^m t \rhd^n t_1$. By the inductive hypothesis we have that
$$t'_0 \rhd^n t'' \lhd^m t_1$$
for some t'_1. This means in particular that $t_0 \lhd t'_0 \rhd^n t'_1$, so we have again from the inductive hypothesis a t' with
$$t_0 \rhd^n t' \lhd t'_1$$
Putting these together we indeed have $t_0 \rhd^n t' \lhd^{m+1} t_1$, as required. □

The following corollary is immediate:

Corollary 59 (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 CR for $\rightsquigarrow$:

Proof of Theorem 51. Suppose $t_0 \leftsquigarrow^* s \rightsquigarrow^* t_1$. Then, by Corollary 54 we have $t_0 \lhd^* s \rhd^* t_1$. By Corollary 59 above, we have some s' with $t_0 \rhd^* s' \lhd^* t_1$, whence $t_0 \rightsquigarrow^* s' \leftsquigarrow t_1$ by Corollary 54 again. □

From here we derive other properties:

Corollary 60 (RCA_0)**.** *We have the following:*

(1) *(Peaks and valleys)* $s \approx t$ *if and only if* $\exists r.\, s \rightsquigarrow^* r \leftsquigarrow^* t$.
(2) *(UN for* $\rightsquigarrow$*) If* s_0 *and* s_1 *are normal with* $s_0 \leftsquigarrow^* t \rightsquigarrow^* s_1$*, then* $s_0 = s_1$.
(3) *(UN for* $\approx$*) If* s_0 *and* s_1 *are normal with* $s_0 \approx t \approx s_1$*, then* $s_0 = s_1$.

Proof sketch. For (1), the right-to-left implication is obvious, so we prove the left-to-right implication by induction on the length of a derivation $s \approx t$. The critical case is when we have $s \approx t' \rightsquigarrow t$ (or equivalently $s \leftsquigarrow s' \approx t$). By the inductive hypothesis we have some r' s.t. $s \rightsquigarrow^* r' \leftsquigarrow^* t'$. This means that we have $r' \leftsquigarrow^* t' \rightsquigarrow t$ so by confluence, Theorem 51, we have some r s.t. $r' \rightsquigarrow^* r \leftsquigarrow^* t$.

(2) is immediate from Theorem 51 and the definition of normal form.

(3) is immediate from (1), the definition of normal form and (2). □

6.5. **Structure of finitely applied coderivations, constants and variables.** Before presenting the main type structures of this section, let us take a moment to note that we have now proven enough to show that the set of farcs, under conversion, forms a model of T without induction and extensionality, even provably within RCA_0.

Definition 61 (Farc structure)**.** We write FARC for the type structure of farcs, defined as follows:

- $\sigma^{\mathsf{FARC}} := \{t : \sigma \mid t \text{ is a farc}\}$.
- $\mathsf{r}^{\mathsf{FARC}}$ is just r, for each constant r.
- $t \circ^{\mathsf{FARC}} s$ is just ts.
- $=_\sigma^{\mathsf{FARC}}$ is just $\approx_\sigma$.

All the structures we consider in this section will be substructures of FARC that are moreover closed under conversion. Thus they will inherit the (quantifier-free) theory of FARC, to which end the following result is naturally indispensable:

Theorem 62 (RCA_0)**.** FARC *is a model of* $T - \{\mathrm{ER}, \mathrm{Ind}\}$*. I.e. each* T *term* t *of type* σ *is in* σ^{FARC}*, and moreover* FARC *satisfies the axioms of Figures 1, 4, 6, 8 and the axioms* (1) *and* (2) *of Figure 7.*

Proof. For each T term t of type σ we have $t \in \sigma^{\mathsf{FARC}}$ simply by definition of a farc, so we continue to verify the axioms.

The axioms governing the constants, i.e. those from Figures 4, 6 and 8, follow immediately from the definitions of $\rightsquigarrow$ and $\approx$, cf. Figure 9.

For the equality axioms from Figure 1:

- Reflexivity of $\approx$ follows by definition.
- For the Leibniz property, suppose that $t \approx t'$ and $r(t) \approx s(t)$. Then we have $r(t) \approx r(t')$ and $s(t) \approx s(t')$ by closure of $\approx$ under contexts, and the fact that farcs have only finitely many redexes, so there are only finitely many occurrences of t in $r(t)$ and $s(t)$ (cf. Corollary 48). Thus we have $r(t') \approx s(t')$ by transitivity of $\approx$. By symmetry, we also have that if $r(t') \approx s(t')$ then $r(t) \approx s(t)$, and so in general $\varphi(t) \equiv \varphi(t')$ for any formula φ.

For the first two number-theoretic axioms from Figure 7:

- $\neg \mathsf{s}0 \approx 0$ follows from confluence, in particular unique normal forms cf. Corollary 60: both $\mathsf{s}0$ and 0 are normal but are not identical.

- Suppose $\mathsf{s}s \approx \mathsf{s}t$, so $\mathsf{s}s \rightsquigarrow^* u \leftsquigarrow^* \mathsf{s}t$, by Corollary 60. Notice that no reduction rule has s at the head, so by Σ^0_1 induction we can extract reduction sequences $s \rightsquigarrow^* u' \leftsquigarrow^* t$ for some u' with $u = \mathsf{s}u'$. Thus indeed $s \approx t$. □

6.6. **Hereditarily total coterms under conversion.** We are now ready to present the type structures that will allow us to obtain a simulation 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 farcs under conversion as the underlying model of computation, cf. Proposition 25.

Definition 63 (Hereditarily total farcs)**.** We define the following sets of farcs:

- $\mathsf{HR}_N := \{t : N \mid \exists n \in \mathbb{N}.\ t \approx \underline{n}\}$
- $\mathsf{HR}_{\sigma\to\tau} := \{t : \sigma \to \tau \mid \forall s \in \mathsf{HR}_\sigma.\ t\,s \in \mathsf{HR}_\tau\}$

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 farcs of type σ.

Notice that, by the confluence result of the previous subsection, namely by Corollary 60, 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 64. *HR_N is Σ^0_1, and if $\mathsf{lev}(\sigma) = n > 0$ then HR_σ is Π^0_{n+1}.*

Proof. HR_N is Σ^0_1 just since $\approx$ is Σ^0_1. Furthermore, expanding out the definition:

$$\mathsf{HR}_{\sigma\to\tau}(t) \iff \forall s.(s \in \mathsf{HR}_\sigma \implies ts \in \mathsf{HR}_\tau)$$

We proceed by the induction on the structure of σ.

In the base case, when $\sigma = \tau = N$, we have that HR_σ and HR_τ are Σ^0_1, and so $\mathsf{HR}_{\sigma\to\tau}$ is indeed Π^0_2, as required.

For the inductive hypothesis, we have that HR_σ and HR_τ are Π_n and Π_{n+1} respectively, and so again $\mathsf{HR}_{\sigma\to\tau}$ is indeed Π_{n+1}. □

Proposition 65 (Closure properties of HR)**.** *Suppose $\mathsf{lev}(\sigma) < n$ and $\mathsf{lev}(\tau) \leq n$. Then we have the following, provably in RCA_0:*

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.

Proof. (1) is immediate from the definition of the sets HR_σ. (2) follows by (meta-level) induction on the structure of τ:

- The base case, when $\tau = N$, follows from symmetry and transitivity of $\approx$.[17]
- Suppose $\tau = \sigma \to \tau'$ and let $s \in \mathsf{HR}_\sigma$. Since $t \approx t'$ we also have $t\,s \approx t's$, by closure of $\approx$ under contexts, and so $t's \in \mathsf{HR}_{\tau'}$, by the inductive hypothesis. Thus $t' \in \mathsf{HR}_\tau$, as required. □

These properties justify defining the following type structure:

[17]Note here that it was important to take conversion, $\approx$, which is symmetric, rather than reduction, $\rightsquigarrow^*$, for the definition of HR_N.

Definition 66 (HR structure)**.** We simply 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.
- $=_\sigma^{\mathsf{HR}}$ is $\approx_\sigma$.

Ultimately we will show that this structure constitutes a model of $CT-\mathrm{ER}$ (i.e. without extensionality). In fact, in Section 7 we will prove this within an appropriate theory of arithmetic, so that we may also extract terms of T that are equivalent under conversion.

Note that HR is a substructure of FARC, in the model-theoretic sense, and so inherits its quantifier-free theory (over the respective domains). The key feature of HR over FARC is that it satisfies induction, provably in RCA_0 in the following sense:

Lemma 67 (Induction for HR, RCA_0)**.** *Suppose $r(x)$ and $s(x)$ are farcs. 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 $\forall t \in \mathsf{HR}_N.r(t) \approx s(t)$.*

Proof. This is essentially 'forced' by the definition of HR_N, reducing the statement to induction in RCA_0. Assuming the premisses, in particular we have $r(\underline{n}) \approx s(\underline{n})$ implies $r(\mathsf{s}\underline{n}) \approx s(\mathsf{s}\underline{n})$ for any $n \in \mathbb{N}$, and so by Σ^0_1-induction on n we have $\forall n \in \mathbb{N}.\, r(\underline{n}) \approx s(\underline{n})$. Now, suppose $t \in \mathsf{HR}_N$. Then by definition we have $t \approx \underline{n}$ for some $n \in \mathbb{N}$, and so indeed $r(t) \approx s(t)$ by the Leibniz property (inherited from FARC, cf. Theorem 62) □

Notice that induction for arbitrary quantifier-free formulas may be duly reduced to the case of equational formulas in the usual way, interpreting Boolean connectives as operations on (co)terms. To conclude that HR actually constitutes a model of T (and later of CT), without extensionality, we will further have to show that it can interpret each term of T (and later coterm of CT). For T, this follows from well-known standard results:

Proposition 68. HR *is a model of* $T-\mathrm{ER}$.

Proof sketch. Given Theorem 62 and Lemma 67 above, it remains to show that for each closed term t of type τ in T, we indeed have that $t \in \mathsf{HR}_\tau$. This part of the argument is in fact what requires significant logical complexity, since it implies the consistency of T (and so also PA and ACA_0), and is what cannot be directly transferred to the coterm setting. Given a term of type $\tau = \vec{\tau} \to N$ and $\vec{t} \in \mathsf{HR}_{\vec{\tau}}$, we have that $t\,\vec{t}$ converts to a unique numeral, thanks to Tait's result Fact 11, and since the only normal terms of type N are numerals. Thus $t\,\vec{t} \in \mathsf{HR}_N$, and so $t \in \mathsf{HR}_\tau$ by repeatedly unfolding its definition. □

In fact this proof 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 coterms in HR_n later, but with the quantifier complexity of induction increased by 1.

6.7. **Modelling extensionality via collapse of conversion.** One issue with the HR structure is that equality is not extensional, and so it is not closed under ER. In particular two distinct coderivations may compute the same function, but are

already in normal form if we have not fed any inputs. A very simple example already at type 1 are the coderivations $\mathsf{cond}\,0\,0 : N \to N \to N$ vs $\mathsf{cond}'\,0\,0 : N \to N \to N$, where cond examines the first input and cond' examines the second input. Both return 0, but the two coderivations are not convertible. Naturally more complex and nontrivial examples abound, e.g. distinct programs for sorting, etc.

However, we may recover an extensional equality relation in a standard way thanks to the notion of extensional collapse from higher-order computability theory, cf. [LN15].

Definition 69 (Extensional equality and HE-structure)**.** We define the following relations $\simeq_\sigma$ on coterms.

- $\simeq_N$ is just $\approx_N$.
- $t \simeq_{\sigma\to\tau} t'$ if $\forall s \in \mathsf{HR}_\sigma.\ t\,s \simeq t's$

We define the type structure HE just as HR, but with $=_\sigma$ interpreted by $\simeq_\sigma$.

The price to pay for extensionality, however, is high: the equality relation is no longer semi-recursive, and its logical complexity grows with type level.

Fact 70. *$\simeq_N$ is Σ^0_1 and, if $\mathsf{lev}(\tau) = n > 0$, then $\simeq_\tau$ is Π^0_{n+1}.*

Proof. The case of N follows immediately from the definition, and for $\mathsf{lev}(\tau) > 0$ we proceed by induction on the structure of $\tau = \rho \to \sigma$ with $\mathsf{lev}(\rho) < n$ and $\mathsf{lev}(\sigma) \leq n$.

- If $\rho = \sigma = N$, so $n = 1$, then $t \simeq_\tau t'$ iff $\forall s(s \in \mathsf{HR}_N \supset ts \approx t's)$. We have $s \in \mathsf{HR}_N$ is Σ^0_1 by Fact 64 and $\approx$ is also Σ^0_1, so indeed $t \simeq_\tau t'$ is Π^0_2.
- For the inductive step, $t \simeq_\tau t'$ iff $\forall r(r \in \mathsf{HR}_\rho \supset tr \simeq_\sigma t'r)$. We have $r \in \mathsf{HR}_\rho$ is Π^0_n by Fact 64 and $\simeq_\sigma$ is Π^0_n by inductive hypothesis, so indeed $t \simeq_\tau t'$ is Π^0_{n+1}. □

Note that we have the following closure properties in HE:

Proposition 71. *Suppose $t \in \mathsf{HR}_\tau$. Then:*

(1) *if $t \approx_\tau t'$ then $t \simeq_\tau t'$. ($\simeq$ coarser than $\approx$)*
(2) *if $t \simeq_\tau t'$ then $t' \in \mathsf{HR}_\tau$. (HR closed under $\simeq$)*

Proof. We proceed by induction on the structure of τ.

- If $\tau = N$ then the statements are immediate from the equivalence of $\approx_N$ and $\simeq_N$ and symmetry/transitivity.
- Suppose $\tau = \rho \to \sigma$ and let $r \in \mathsf{HR}_\rho$.
  (1) If $t \approx_\tau t'$ then also $t\,r \approx t'r$, by closure of $\approx$ under contexts. Therefore $t\,r \simeq_\sigma t'r$, by the inductive hypothesis, and so $t \simeq_\tau t'$.
  (2) If $t \simeq_\tau t'$ then $t\,r \simeq_\sigma t'r$, by definition. Therefore $t'r \in \mathsf{HR}_\sigma$ by the inductive hypothesis, and so $t' \in \mathsf{HR}_\tau$. □

Consequently, HE is in turn a substructure of HR and inherits its quantifier-free theory. Notice that this does not a priori cover satisfaction of induction, but nonetheless, as promised, we have the following:

Proposition 72. *HE is a model of T.*

Proof sketch. Since HE is a substructure of FARC, it remains to verify ER and Ind.

Extensionality is 'forced' by the definition of $\simeq$. Suppose $t, t' \in \mathsf{HR}_{\sigma\to\tau}$ and for all $s \in \mathsf{HR}_\sigma$ we have $ts \simeq t's$. Then by definition we have $t \simeq t'$.

We may show that HE satisfies induction in a similar way as we did for HR, Lemma 67, though note that this is no longer possible in RCA_0, due to the increasing complexity of $\simeq_\tau$ with the level of τ. Supose $r(0) \simeq s(0)$ and for all $t \in \mathsf{HR}_N$ we have $r(t) \simeq s(t) \implies r(\mathsf{s}t) \simeq s(\mathsf{s}t)$. In particular we have that $r(\underline{n}) \simeq s(\underline{n}) \implies r(\mathsf{s}\underline{n}) \simeq s(\mathsf{s}\underline{n})$, for any $n \in \mathbb{N}$. We thus have for all $n \in \mathbb{N}$ that $r(\underline{n}) \simeq s(\underline{n})$ by Π^0_{k+1}-induction on n (where $\mathsf{lev}(r), \mathsf{lev}(s) \leq k$). Now, suppose $t \in \mathsf{HR}_N$. Then by definition we have $t \approx \underline{n}$, and so $t \simeq \underline{n}$, for some $n \in \mathbb{N}$, so indeed $r(t) \simeq s(t)$ by the Leibniz property (inherited from FARC, Theorem 62). □

7. From CT_n to T_{n+1}, via arithmetisation of models

In this section we will present a converse result to that of Section 5, i.e. that terms of T_{n+1} may simulate terms of CT_n, satisfying the same type 1 quantifier-free theory. The methodology of Section 5 was entirely proof theoretic, thanks in part to the well-foundedness of the T-terms that were simulated. A priori, we do not admit a similar methodology for the converse direction, due to the non-wellfoundedness of coterms, and so we employ a model-theoretic approach. Namely, we will show that the type structures HR and HE introduced in the previous section indeed constitute models of $CT-\mathrm{ER}$ and CT, respectively. In fact, we will formalise the membership of CT_n coterms 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 proof mining.

Throughout this section we continue to work only with coterms that are finite applications of coderivations, variables and constants (farcs). We will work mainly within $\mathsf{RCA}_0 + I\Sigma^0_{n+1}$ or $\mathsf{RCA}_0 + I\Sigma^0_{n+2}$, to be clearly indicated at the appropriate points.

7.1. Canonical branches of non-total coterms.

As for T, the main difficulty in showing that HR or HE is a model of CT is in showing that each CT coterm is, indeed, interpreted in the structure. For us, this will amount to a formalised proof of the totality of (regular) progressing coterms. Our approach will be to import a suitable version of the proof of Proposition 35 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 73. *Consider a rule instance as follows, with $k \leq 2$:*

$$\mathsf{r}\,\frac{\vec{\sigma}_1 \Rightarrow \tau_1 \quad \cdots \quad \vec{\sigma}_k \Rightarrow \tau_k}{\vec{\sigma} \Rightarrow \tau}$$

If $t_i \in \mathsf{HR}_{\vec{\sigma}_i \to \tau_i}$ then $\mathsf{r}\ t_1\ \cdots\ t_n \in \mathsf{HR}_{\vec{\sigma} \to \tau}$.

This follows by simple inspection of the rules of CT, and we shall indeed give a proof of a more refined statement shortly. As a consequence, by contraposition, any coderivation $\notin \mathsf{HR}$ must induce an infinite branch of coderivations $\notin \mathsf{HR}$, similarly to the proof of Propostion 35. The next definition formalises a canonical such branch, as induced by an input on which the 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 T_n coderivation, in $\mathsf{RCA}_0 + I\Sigma^0_{n+2}$.

Definition 74 (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. $t\,\vec{s} \notin \mathsf{HR}_\tau$. 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 $\vec{s}_0$ as follows, always satisfying the invariant $t_i\,\vec{s}_i \notin \mathsf{HR}_{\tau_i}$:

(1) (t_i cannot be an initial sequent).

(2) Suppose $t_i = \mathsf{wk}\dfrac{\overset{t}{\vec{\sigma} \Rightarrow \tau}}{\vec{\sigma}, \sigma \Rightarrow \tau}$ and $\vec{s}_i = (\vec{s}, s)$. Then $t_{i+1} := t$ and $\vec{s}_{i+1} := \vec{s}$.

(3) Suppose $t_i = \mathsf{ex}\dfrac{\overset{t}{\vec{\rho}, \sigma, \rho, \vec{\sigma} \Rightarrow \tau}}{\vec{\rho}, \rho, \sigma, \vec{\sigma} \Rightarrow \tau}$ 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 $t_i = \mathsf{cntr}\dfrac{\overset{t}{\vec{\sigma}, \sigma, \sigma \Rightarrow \tau}}{\vec{\sigma}, \sigma \Rightarrow \tau}$ and $\vec{s}_i = (\vec{s}, s)$. Then $t_{i+1} := t$ and $\vec{s}_{i+1} := (\vec{s}, s, s)$.

(5) Suppose $t_i = \mathsf{cut}\dfrac{\overset{t}{\vec{\sigma} \Rightarrow \rho} \quad \overset{t'}{\vec{\sigma}, \rho \Rightarrow \tau}}{\vec{\sigma} \Rightarrow \tau}$ and $\vec{s}_i = \vec{s}$. Then if $t\,\vec{s} \in \mathsf{HR}_\rho$ 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 $t_i = \mathsf{L}\dfrac{\overset{t}{\vec{\sigma} \Rightarrow \rho} \quad \overset{t'}{\vec{\sigma}, \sigma \Rightarrow \tau}}{\vec{\sigma}, \rho \to \sigma \Rightarrow \tau}$ 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 $t_i = \mathsf{R}\dfrac{\overset{t}{\vec{\sigma}, \sigma \Rightarrow \tau}}{\vec{\sigma} \Rightarrow \sigma \to \tau}$ and $\vec{s}_i = \vec{s}$. Let s be the least[18] 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 $t_i = \mathsf{cond}\dfrac{\overset{t}{\vec{\sigma} \Rightarrow \tau} \quad \overset{t'}{\vec{\sigma}, N \Rightarrow \tau}}{\vec{\sigma}, N \Rightarrow \tau}$ 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 $\vec{s}_{i+1} := (\vec{s}, \underline{n})$.

Note that certain arbitrary choices from the proof of Proposition 35 have been made canonical in the definition above. We also define inputs for cut and L directly, to ease the formalisation in arithmetic.

Proposition 75. *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.*

[18]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, by induction on that property.

$t_0\,\vec{s}_0 \notin \mathsf{HR}_{\tau_0}$ 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.

Proof. 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 ith 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 64. 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 74 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,

$$\exists (t_i, \vec{s}_i).(\mathrm{Gen}(i, (t_0, \vec{s}_0), (t_i, \vec{s}_i)) \wedge t_i\,\vec{s}_i \notin \mathsf{HR}_{\tau_i}) \tag{22}$$

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 64, 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 (22) 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 74 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 74.

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:

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

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 65. We have:

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

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

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

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 64, 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, namely Corollary 60, 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:

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

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

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

This concludes the proof. □

7.2. **Progressing coterms are hereditarily total.** We are now ready to show that progressing coterms are hereditarily total, i.e. that they belong to HR (and HE). Now that we have formalised the infinite 'non-total' branches of the proof of Proposition 35, relativised to the type structure HR, we continue to formalise the remainder of the argument.

Lemma 76 (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 74.*

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 some $n_i, n_{i+1} \in \mathbb{N}$, then:

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

Proof. Follows directly from inspection of Definition 74, and confluence of conversion, namely Corollary 60. □

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. The reason for this is that, even though non-total branches are well-defined by Proposition 75, we do not a priori have access to them as *sets* in extensions of RCA_0 by fragments of arithmetical induction, and so the lack of progressing threads along them does not directly contradict the fact that a coderivation is progressing. 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.

A similar issue underlies the notion of 'arithmetical acceptance' for a non-deterministic automaton in [Das20]. The fact that our arithmetical approximation suffices is borne out by the 'moreover' clause in the following result:

Proposition 77 (RCA_0)**.** *Suppose t_i and $\vec{s}_i$ are as in Lemma 76. 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.*

Proof. 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, Corollary 60. 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 76 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, Corollary 60. 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. □

Finally, we are ready to show that coterms of CT are indeed interpreted in the type structures we have presented. It is here that we will have to make use of the fact that, for each regular coderivation, RCA_0 proves whether it progresses and, moreoever, we shall take advantage of the aforementioned implied arithmetical approximation of progression in order to deduce the necessary contradiction without invoking additional set existence principles.

Theorem 78. *Let $t : \vec{\sigma} \Rightarrow \tau$ be a progressing coderivation containing only types of level $\leq n$ (i.e. a CT_n-coderivation). Then $\mathsf{RCA}_0 + I\Sigma^0_{n+2} \vdash t \in \mathsf{HR}_{\vec{\sigma} \to \tau}$.*

Proof. First, by Proposition 34 (from [Das20]), 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:[19]

$$\exists k.\forall m.\text{ there is a (finite) } N\text{-thread from } t_k \text{ progressing} > m \text{ times} \tag{23}$$

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 (23), 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 75, we can Δ^0_{n+2}-well-define the branch $(t_i)_i$ generated by t and $\vec{s}$. Thus we indeed have (23), contradicting Proposition 77. □

Corollary 79. HR *is a model of* $CT-\mathrm{ER}$*, and* HE *is a model of* CT*.*

Proof. All the axioms of $CT-\mathrm{ER}$ and CT are already satisfied in HR and HE respectively, inherited from Propositions 68 and 72 respectively. Thus the result follows immediately from Theorem 78 above (and soundness). □

Corollary 80. *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* $t'^{\mathfrak{N}} = t^{\mathfrak{N}}$*.*

Proof. By Theorem 78 we have, in particular, that:

$$\mathsf{RCA}_0 + I\Sigma^0_{n+2} \vdash \forall \vec{m} \exists n\, t\underline{\vec{m}} \approx \underline{n} \tag{24}$$

Since this is a Π^0_2 theorem, we may apply extraction, Proposition 14, to obtain the required term t' of T_{n+1} so that $t'\vec{m}$ witnesses the existential quantifier in (24). The result now follows by soundness of $\mathsf{RCA}_0 + I\Sigma^0_{n+2}$ and since $(\mathsf{HR}, \approx)$ is a substructure of the standard model $\mathfrak{N}$. □

7.3. **Interpretation of $CT_n(-\mathrm{ER})$ into $T_{n+1}(-\mathrm{ER})$.** 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 ACA_0 to fragments of T to interpret CT, the same coding carries over. At the risk of confusion, we shall suppress this formality in the statements of results that follow, in line with the exposition so far.

Theorem 81. *If* $CT_n - \mathrm{ER} \vdash s = t$ *then* $T_{n+1} - \mathrm{ER} \vdash s \approx t$*.*

Proof. Let us work in $\mathsf{RCA}_0 + I\Sigma^0_{n+2}$. By Theorem 78 we have that $s, t \in \mathsf{HR}_\sigma$, so suppose that $CT_n - \mathrm{ER} \vdash s = t$ (which is a Σ^0_1 relation). Now, invoking Theorem 62 and Lemma 67, we indeed have that $s \approx t$, by Σ^0_1-induction on the $CT_n - \mathrm{ER}$ proof of $s = t$.

Now, invoking the extraction theorem, Proposition 14, for the above paragraph, we can extract a T_{n+1}-term $d(\cdot)$ witnessing the following 'reflection' principle:

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

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

[19] The argument for this is similar to that of Proposition 6.2 from [Das20].

We also have an analogous statement in the presence of the extensionality rule though, as expected, we must restrict to only type 1 equations for the interpretation to hold:

Theorem 82. *Let* $\mathsf{lev}(s) = \mathsf{lev}(t) \leq 1$. *If* $CT_n \vdash s = t$ *then* $T_{n+1} \vdash s \simeq t$.

Proof. The proof is similar to the one above, only using the proof of Proposition 72 instead of Lemma 67 to simulate induction steps of CT_n in HE (now requiring $I\Sigma^0_{n+2}$). The restriction on levels is required in order to invoke the extraction theorem, since the logical complexity of $\simeq_\sigma$ grows with the type level of σ. In particular, $\simeq_1$ is a Π^0_2 relation, by Fact 70, and unwinding its definition we have $s \simeq_1 t$ is equivalent to $\forall \vec{r} \forall \vec{n} \forall \vec{d} : \vec{r} \approx \underline{\vec{n}}.\ \exists d :\ \vec{r} \approx t\,\vec{r}$. Thus we extract the following 'reflection' principle for some T_{n+1}-term d:

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

Again substituting a concrete CT_n proof P of $s = t$ into the above principle indeed gives us $T_{n+1} \vdash s \simeq t$, as required. □

8. Perspectives and further results

In this section we shall give some further discussion and results related to the system CT we have presented.

8.1. **On confluence and consistency.** We should point out that, from the point of view of just the extensional properties of extracted programs, it is not necessary to use the confluence result we presented in Section 6. We could simply assume consistency of CT as an axiom throughout Section 7 (indeed we shall take this direction in the next subsection for simplicity). Consistency of CT is a true Π^0_1 statement by meta-level reasoning, and we would be able to extract the same functionals, since Π^0_1 statements carry no computational content. This is particularly pertinent when extracting an infinitely descending sequence of natural numbers from a progressing thread in Lemma 76: we require these natural numbers to be uniquely defined, which follows by either confluence or consistency. However this approach would compromise our ultimate interpretations of CT_n, requiring the same consistency principle to be added to the target theory T_{n+1} (with and without ER).

Incidentally, note that our approach, by a special case of the simulations from Sections 5 and 7, implies the *equiconsistency* of CT_n and T_{n+1}, over a weak base theory.

8.2. **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. The classical way to prove this is to augment the rewrite system from Section 6 by fresh type 1 constant symbols that play the roles of the inputs, say of a type 2 functional, and adding appropriate reductions. An account of this is given in [Tro73], though there are several other known arguments, e.g. by showing that the strucutre of *total continuous functionals* forms a model for T (cf. [Sca71]) or, more recently, via an elegant form of 'syntactic continuity predicate' (cf. [Xu20]).

For the case of CT, we may actually formalise a variation of this argument within second-order arithmetic, extending the simulation of CT coterms within T to type 2 functionals. We shall here refrain from an analysis of abstraction complexity, for the sake of brevity, and also focus solely on the interpretation of (co)terms in the

standard model, disregarding their theories. Our exposition will be brief, since the finer details are adaptations of earlier results.

Let us fix a CT coderivation $t : \vec{\sigma} \Rightarrow N$ s.t. each $\sigma_i = \overbrace{N \to \cdots \to N}^{k_i} \to N$, and let us henceforth work in ACA_0, distinguishing second-order variables $f_i : \overbrace{\mathbb{N} \times \cdots \times \mathbb{N}}^{k_i} \to \mathbb{N}$, intuitively representing the inputs for t.

Within CT, introduce new (uninterpreted) constant symbols $\underline{f}_i : \overbrace{N \to \cdots \to N}^{k_i} \to N$ for each σ_i, and new reduction steps:

$$\underline{f}_i \, \underline{n}_1 \, \ldots \, \underline{n}_{k_i} \rightsquigarrow \underline{f_i(n_1, \ldots, n_{k_i})} \tag{25}$$

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 6 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, it carries no computational content and adding it to ACA_0 still admits extraction into T. (The drawback of this, as mentioned in the previous subsection, is that we do not recover any bona fide interpretation of CT into T.)

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 (25) above. By adapting our approach from Section 7, we may show the following:

Theorem 83 ($\mathsf{ACA}_0 + \mathrm{UNF}_N$)**.** $\forall \vec{f}. \, t \, \underline{\vec{f}} \in \mathsf{HR}^{\vec{f}}_N$

Proof sketch. The argument is essentially the same as that for Theorem 78. Assuming otherwise, for contradiction, we may generate a non-hereditarily-total branch is just as in Definition 74, and its well-definedness is shown just as in Proposition 75. 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 7. 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 76 for $\mathsf{HR}^{\vec{f}}_N$, this time using UNF_N above instead of confluence, and so the appropriate contradiction of the well-ordering property of $\mathbb{N}$ is readily obtained. □

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., [Koh08]), we may witness the existential by a term $t'(\vec{f})$ of T, so that $T + \mathrm{UNF}_N \vdash$

$t\,\vec{f} \approx \underline{t'(\vec{f})}$. Now, in the standard model $\mathfrak{N}$, we have that UNF_N is true and that $\approx$ is sound for equality (i.e. $s \approx t \implies s^{\mathfrak{N}} = t^{\mathfrak{N}}$), so we finally have:

Corollary 84. *If t is a level 2 coterm of CT, then there is a T term t' s.t. $t'^{\mathfrak{N}} = t^{\mathfrak{N}}$.*

It would be interesting to see if we could adapt this approach to give models of CT based on continuous functionals at higher types (cf. [LN15]), but such a development is beyond the scope of this work. We point out that forms of continuity at higher types for T are less canonical, cf. [Tro73]. Nonetheless, we shall show in Section 8.6 that, for every CT coterm there is a T term computing the same functional (in $\mathfrak{N}$).

8.3. **A 'term model' à la Tait and strong normalisation.** It is an immediate consequence of our results that CT-coterms are *weakly normalising*.

Proposition 85. *If $t \in \mathsf{HR}$ then t is weakly normalising.*

Proof sketch. We proceed by induction on the type of t:

- For t of type N, we rely on the confluence result, Theorem 51, and normality of numerals.
- If t has type $\sigma \to \tau$, then for some/any $s \in \mathsf{HR}_\sigma$ we have that $ts \in \mathsf{HR}_\tau$, by definition, and so is weakly normalising by the inductive hypothesis. We define a new normalisation sequence for t from one for ts by induction on its length, simply ignoring reductions that are not entirely inside t. □

Thus, by Theorem 78, we have:

Corollary 86. *Each closed CT coterm is weakly normalising. Moreover, any CT_n coterm is provably weakly normalising inside $\mathsf{RCA}_0 + I\Sigma^0_{n+2}$.*

Given that we also have a confluence for CT, we are not far from a strong normalisation result. It would be interesting if we could define an *increasing* measure for reduction to this end, perhaps induced by the progressing thread criterion. Instead, we show that Tait's 'convertibility' predicates may be suitably adapted for this purpose, yielding a minimal model for CT. We will not formalise our exposition within arithmetic, but expect it to go through in a suitable fragment of second-order arithmetic.

We will define a minimal 'coterm model' in a similar way to Tait's term models of sytem T [Tai67]. This is complementary to our development of HR and HE: while those structures were 'over-approximations' 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 87 (Convertibility)**.** We define the following sets of closed CT-coterms:

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

We can adapt suitable versions of Proposition 85 and Proposition 65 to the setting of $\mathcal{C}$:

Proposition 88. *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. ($\mathcal{C} \subseteq \mathrm{SN}$)*

Proof. (1) is immediate from the definition of $\mathcal{C}$.

(2) is proved by induction on type:

- Suppose $t \in \mathcal{C}_N$ and $t \rightsquigarrow t'$. By definition t is strongly normalising, so also t' is strongly normalising, and so $t' \in \mathcal{C}_N$ by definition.
- Suppose $t \in \mathcal{C}_{\sigma\to\tau}$ and $t \rightsquigarrow t'$. Let $s \in \mathcal{C}_\sigma$. Then $ts \rightsquigarrow t's$ by closure of $\rightsquigarrow$ under contexts, and so $t's \in \mathcal{C}_\tau$ by the inductive hypothesis. Since the choice of $s \in \mathcal{C}_\sigma$ was arbitrary, we have $t' \in \mathcal{C}_{\sigma\to\tau}$.

Finally, (3) is also proved by induction on type:

- Suppose $t \in \mathcal{C}_N$. Then t is strongly normalising by definition.
- Suppose $t \in \mathcal{C}_{\sigma\to\tau}$ and, for contradiction, let $t = t_0 \rightsquigarrow t_1 \rightsquigarrow \cdots$ be a diverging reduction sequence. Then for some/any $s \in \mathcal{C}_\sigma$, we have that $ts = t_0 s \rightsquigarrow t_1 s \rightsquigarrow \cdots$ is also a diverging reduction sequence, contradicting the inductive hypothesis. □

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 any redex of ts is either entirely in t or entirely in s. We also have the following expected characterisation of convertibility:

Lemma 89 (Convertibility lemma)**.** *Let t be neutral. If $\forall t' \leftsquigarrow t.\ t' \in \mathcal{C}_\tau$, then $t \in \mathcal{C}_\tau$.*

Proof. We proceed by induction on type. The base case, for type N, follows by definition of strong normalisation, so let us assume,

$$\forall t' \leftsquigarrow t.\ t' \in \mathcal{C}_{\sigma\to\tau} \tag{26}$$

and assume that the statement of the Proposition holds for all smaller types (IH), in particular σ and τ.

To prove $t \in \mathcal{C}_{\sigma\to\tau}$, let $s \in \mathcal{C}_\sigma$ and we show that $ts \in \mathcal{C}_\tau$. In fact, we will show,

$$\forall r \leftsquigarrow ts.\ r \in \mathcal{C}_\tau \tag{27}$$

whence $ts \in \mathcal{C}_\tau$ follows by the inductive hypothesis (IH) for τ. Since $s \in \mathcal{C}_\sigma$ and so is strongly normalising by Proposition 88, we may prove (27) by a sub-induction on the size of the complete reduction tree of s, say $\mathrm{RedTree}(s)$.[20]

- Suppose $r = t's$, so $t \rightsquigarrow t'$. Then $t' \in \mathcal{C}_{\sigma\to\tau}$ by (26), and so $t's = r \in \mathcal{C}_\tau$.
- Suppose $t = ts'$, so $s \rightsquigarrow s'$. Then we have $\mathrm{RedTree}(s') < \mathrm{RedTree}(s)$ so by the sub-inductive hypothesis (27) we have $\forall r' \leftsquigarrow ts'.\ r' \in \mathcal{C}_\tau$. Thus by the main inductive hypothesis (IH) for τ, we have $ts' = r \in \mathcal{C}_\tau$.

In all cases we have that $r \in \mathcal{C}_\tau$, yielding (27) as required. □

Now we can go on to define a non-converting branch, just like we did for the standard model $\mathfrak{N}$ in Proposition 35 (non-total branch), and for HR (also HE) in Definition 74 (non-hereditarily-recursive branch). As in the latter case, we need to prove well-definedness of such a branch, cf. Observation 73 and Proposition 75.

[20]Since regular progressing coterms are, in particular, finite applications of coderivations, constants and variables (i.e. farcs), there are only finitely many redexes in a CT coterm, and so the reduction tree is finitely branching. By König's lemma, the complete reduction tree is thus finite.

Proposition 90 (Preservation of convertibility)**.** *Let $\vec{r} \in \mathcal{C}_{\vec{\rho}}$ and $\vec{s} \in \mathcal{C}_{\vec{\sigma}}$. We have the following:*[21]

- *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 $\mathsf{cond}\, t_0\, t_1\, \vec{s}\, \mathsf{s}s \in \mathcal{C}_\tau$.*

Proof. We proceed by induction on $\mathrm{RedTree}(\vec{s})+\mathrm{RedTree}(s)+\mathrm{RedTree}(\vec{r})+\mathrm{RedTree}(r)$. In most cases there is a redex at the head, so we shall directly use the conversion lemma, rather showing that any term obtained from reduction is convertible.

- $\mathsf{id}\, s \rightsquigarrow \mathsf{id}\, s' \in \mathcal{C}_\sigma$ by the inductive hypothesis, and $\mathsf{id}\, s \rightsquigarrow s \in \mathcal{C}_\sigma$ by assumption. Thus $\mathsf{id}\, s \in \mathcal{C}_\sigma$ by Lemma 89.
- $\mathsf{ex}\, t\, \vec{r}\, r\, s\, \vec{s} \rightsquigarrow \mathsf{ex}\, t\, \vec{r}'\, r'\, s'\, \vec{s}' \in \mathcal{C}_\tau$ by inductive hypothesis, and $\mathsf{ex}\, t\, \vec{r}\, r\, s\, \vec{s} \rightsquigarrow t\, \vec{r}\, s\, r\, \vec{s} \in \mathcal{C}_\tau$ by assumption. Thus $\mathsf{ex}\, t\, \vec{r}\, r\, s\, \vec{s} \in \mathcal{C}_\tau$ by Lemma 89.
- $\mathsf{wk}\, t\, \vec{s}\, s \rightsquigarrow \mathsf{wk}\, t\, \vec{s}'\, s' \in \mathcal{C}_\tau$ by inductive hypothesis, and $\mathsf{wk}\, t\, \vec{s}\, s \rightsquigarrow t\, \vec{s} \in \mathcal{C}_\tau$ by assumption. Thus $\mathsf{wk}\, t\, \vec{s}\, s \in \mathcal{C}_\tau$ by Lemma 89.
- $\mathsf{cntr}\, t\, \vec{s}\, s \rightsquigarrow \mathsf{cntr}\, t\, \vec{s}'\, s' \in \mathcal{C}_\tau$ by inductive hypothesis, and $\mathsf{cntr}\, t\, \vec{s}\, s \rightsquigarrow t\, \vec{s}\, s\, s \in \mathcal{C}_\tau$ by assumption. Thus $\mathsf{cntr}\, t\, \vec{s}\, s \in \mathcal{C}_\tau$ by Lemma 89.
- $\mathsf{cut}\, t_0\, t_1\, \vec{s} \rightsquigarrow \mathsf{cut}\, t_0\, t_1\, \vec{s}' \in \mathcal{C}_\tau$ by inductive hypothesis, and $\mathsf{cut}\, t_0\, t_1\, \vec{s} \rightsquigarrow t_1\, \vec{s}\, (t_0\, \vec{s}) \in \mathcal{C}_\tau$ by assumptions. Thus $\mathsf{cut}\, t_0\, t_1\, \vec{s} \in \mathcal{C}_\tau$ by Lemma 89.
- $\mathsf{L}\, t_0\, t_1\, \vec{s}\, r \rightsquigarrow \mathsf{L}\, t_0\, t_1\, \vec{s}'\, r' \in \mathcal{C}_\tau$ by inductive hypothesis, and $\mathsf{L}\, t_0\, t_1\, \vec{s}\, r \rightsquigarrow t_1\, \vec{s}\, (r\, (t_0\, \vec{s})) \in \mathcal{C}_\tau$ by assumptions and closure of $\mathcal{C}$ under application, Proposition 88.(1). Thus $\mathsf{L}\, t_0\, t_1\, \vec{s}\, r \in \mathcal{C}_\tau$ by Lemma 89.
- To show $\mathsf{R}\, t\, \vec{s} \in \mathcal{C}_{\sigma\to\tau}$, let $s \in \mathcal{C}_\sigma$ and we shall show that $\mathsf{R}\, t\, \vec{s}\, s \in \mathcal{C}_\tau$. We proceed by induction on $\mathrm{RedTree}(s)$ (as well as $\mathrm{RedTree}(\vec{s})$). $\mathsf{R}\, t\, \vec{s}\, s \rightsquigarrow \mathsf{R}\, t\, \vec{s}'\, s' \in \mathcal{C}_\tau$ by inductive hypothesis, and $\mathsf{R}\, t\, \vec{s}\, s \rightsquigarrow t\, \vec{s}\, s \in \mathcal{C}_\tau$ by assumption. Thus $\mathsf{R}\, t\, \vec{s}\, s \in \mathcal{C}_\tau$ by Lemma 89, as required.
- $0 \in \mathcal{C}_N$ since it is already normal.
- There is no reduction at the head of $\mathsf{s}s$, so any reduction sequence for $\mathsf{s}s$ projects to one for s, and so terminates by assumption.
- $\mathsf{cond}\, t_0\, t_1\, \vec{s}\, 0 \rightsquigarrow \mathsf{cond}\, t_0\, t_1\, \vec{s}'\, 0 \in \mathcal{C}_\tau$ by inductive hypothesis, and $\mathsf{cond}\, t_0\, t_1\, \vec{s}\, 0 \rightsquigarrow t_0\, \vec{s} \in \mathcal{C}_\tau$ by assumption. Thus $\mathsf{cond}\, t_0\, t_1\, \vec{s}\, 0 \in \mathcal{C}_\tau$ by Lemma 89.
- $\mathsf{cond}\, t_0\, t_1\, \vec{s}\, \mathsf{s}s \rightsquigarrow \mathsf{cond}\, t_0\, t_1\, \vec{s}'\, \mathsf{s}s' \in \mathcal{C}_\tau$ by inductive hypothesis, and $\mathsf{cond}\, t_0\, t_1\, \vec{s}\, \mathsf{s}s \rightsquigarrow t_1\, \vec{s}\, s \in \mathcal{C}_\tau$ by assumption. Thus $\mathsf{cond}\, t_0\, t_1\, \vec{s}\, \mathsf{s}s \in \mathcal{C}_\tau$ by Lemma 89. □

As a consequence of our results in Sections 6 and 7, observe the following:

Observation 91. *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 78 that $s \approx \underline{n}$ for some $n \in \mathbb{N}$. By confluence, namely Corollary 60, we have that n is unique and furthermore $s \rightsquigarrow^* \underline{n}$. □

[21] All rules have type as presented in Figures 3, 5 and 8.

In fact, both our weak normalisation argument and our confluence result seem to be crucial for establishing the above property of normal elements of $\mathcal{C}_N$, as well as the fact that $\mathcal{C}$ 'under-approximates' the class of CT-coterms. It is otherwise not immediate how we rule out possibilities such as the coterm $\cdots \mathsf{sss}x$, reached at the 'limit' of reducing the following (non-progressing) coderivation (applied to x):[22]

$$\dfrac{\mathsf{s}\dfrac{}{N \Rightarrow N} \quad \mathsf{cut}\dfrac{\vdots}{N \Rightarrow N}\bullet}{\mathsf{cut}\dfrac{}{N \Rightarrow N}\bullet} \tag{28}$$

This is the same issue that we raised in Section 6.2, and in particular another problematic example was given in Remark 44, in the form of a coterm $\mathsf{cond}\,0\,1\,(\mathsf{cond}\,0\,1\,(\cdots))$. In contrast to that coterm, where there were two consistent interpretations in the standard model (0 or 1), the coterm $\cdots \mathsf{sss}x$ has no consistent interpretation in $\mathfrak{N}$.

Theorem 92 (Convertibility for CT)**.** *Any CT-coderivation $t : \vec{\sigma} \Rightarrow \tau$ is in $\mathcal{C}_{\vec{\sigma}\to\tau}$.*

Proof. 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 74 (or the proof of Proposition 35). The only difference is that we use Proposition 90 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 derivation,

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

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 91 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 89, 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 88.(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 88.(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. □

[22]Note that no induction on 'size' is available for general (non-wellfounded) coterms.

Since $\mathcal{C}$ is closed under application, Proposition 88.(1), we inherit $\mathcal{C}$ membership for all CT-coterms. Since elements of $\mathcal{C}$ are strongly normalising, Proposition 88.(3), and since reduction is confluent, Theorem 51, we finally have:

Corollary 93 (Strong normalisation for CT)**.** *Any closed CT coterm strongly normalises to a unique normal form.*

8.4. **Towards an infinitary λ-calculus for CT and cut-elimination.** From the point of view of the *Curry-Howard* correspondence, rules of the sequent calculus are usually associated with meta-level term forming operations for the λ-calculus, rather than comprising constants in their own right. Let us describe this in some more detail here.

First, the appropriate notion of term is defined as follows:

- 0 is a term of type N.
- p is a term of type $N \to N$.[23]
- If t is a term of type $\sigma \to \tau$ and s is a term of type σ then (ts) is a term of type τ.
- If t is a term of type τ and x is a variable of type σ then $\lambda x t$ is a term of type $\sigma \to \tau$.
- If r is a term of type N and s, t are terms of type τ then $\mathtt{if}\ r = 0\ \mathtt{then}\ s\ \mathtt{else}\ t$ is a term of type τ.

From here the typing rules of the sequent calculus can be recast in annotated style, with variable annotations on the LHS and term annotation on the RHS:

$$\mathsf{ex}\,\frac{\vec{x} : \vec{\rho}, y : \sigma, x : \rho, \vec{y} : \vec{\sigma} \Rightarrow t(\vec{x}, y, x, \vec{y}) : \tau}{\vec{x} : \vec{\rho}, x : \rho, y : \sigma, \vec{y} : \vec{\sigma} \Rightarrow t(\vec{x}, x, y, \vec{y}) : \tau} \qquad \mathsf{cntr}\,\frac{\vec{x} : \vec{\sigma}, x_0 : \sigma, x_1 : \sigma \Rightarrow t(\vec{x}, x_0, x_1) : \tau}{\vec{x} : \vec{\sigma}, x : \sigma \Rightarrow t(\vec{x}, x, x) : \tau}$$

$$\mathsf{id}\,\frac{}{x : \sigma \Rightarrow x : \sigma} \qquad \mathsf{wk}\,\frac{\vec{x} : \vec{\sigma} \Rightarrow t(\vec{x}) : \tau}{\vec{x} : \vec{\sigma}, x : \sigma \Rightarrow t(\vec{x}) : \tau} \qquad \mathsf{R}\,\frac{\vec{x} : \vec{\sigma}, x : \sigma \Rightarrow t(\vec{x}, x) : \tau}{\vec{x} : \vec{\sigma} \Rightarrow \lambda x.t(\vec{x}, x) : \sigma \to \tau}$$

$$\mathsf{cut}\,\frac{\vec{x} : \vec{\sigma} \Rightarrow s(\vec{x}) : \sigma \quad \vec{x} : \vec{\sigma}, x : \sigma \Rightarrow t(\vec{x}, x) : \tau}{\vec{x} : \vec{\sigma} \Rightarrow t(\vec{x}, s(\vec{x})) : \tau} \qquad \mathsf{L}\,\frac{\vec{x} : \vec{\sigma} \Rightarrow r(\vec{x}) : \rho \quad \vec{x} : \vec{\sigma}, x : \sigma \Rightarrow t(\vec{x}, x) : \tau}{\vec{x} : \vec{\sigma}, y : \rho \to \sigma \Rightarrow t(\vec{x}, y\, r(\vec{x})) : \tau}$$

$$0\,\frac{}{\Rightarrow 0 : N} \qquad \mathsf{p}\,\frac{}{x : N \Rightarrow \mathsf{p}x : N}$$

$$\mathsf{cond}\,\frac{\vec{x} : \vec{\sigma} \Rightarrow s(\vec{x}) : \tau \quad x : \vec{\sigma}, y : N \Rightarrow t(\vec{x}, y) : \tau}{\vec{x} : \vec{\sigma}, y : N \Rightarrow \mathtt{if}\ y = 0\ \mathtt{then}\ s(\vec{x})\ \mathtt{else}\ t(\vec{x}, \mathsf{p}y) : \tau}$$

As we mentioned from the start, the reason for not pursuing such an approach is that this association of a term to a derivation is not *continuous* and so does not a priori produce a well-defined coterm from arbitrary coderivations. Consider, for instance, the operation on terms induced by the cut rule: we may have to go further along the right branch to eventually print the root of any associated coterm. As an extreme example (an adaptation of) the coderivation from (28),

$$\mathsf{cut}\,\frac{\mathsf{p}\,\dfrac{}{N \Rightarrow N} \qquad \mathsf{cut}\,\dfrac{\vdots}{N \Rightarrow N}\bullet}{N \Rightarrow N}\bullet$$

[23]The interpretation of p is 'predecessor' in the standard model: $\mathsf{p}^{\mathfrak{N}}(0) := 0$ and $\mathsf{p}^{\mathfrak{N}}(n+1) := n$.

has no well-defined coterm representation induced by the type system above. One of the reasons for employing our combinatory approach was simply to avoid such problems; however we also point out that our notion of reduction from Section 6 gives us a consistent way to interpret the derivation above as an infinitary coterm: 'at the limit' it normalises to $\cdots \mathsf{ppp}x$ (after application to x). It would be interesting to make this idea more formal, in particular using tools from *infinitary term rewriting* (see, e.g., [KSSdV05, KS11, Sim12]).

We should mention also that the system and (co)terms above, as presented, are *not* strongly normalising, due to the presence of predcessor, p, instead of successor, s, and the corresponding cond rule. An appropriate treatment should seek to avoid this issue.

On a similar subject, the Curry-Howard correspondence also motivates the investigation of *cut-elimination* for our type system, cf. [BDS16, BDKS20]. The relationship to normalisation for our reduction system is not entirely clear, since the way we associate (co)derivations and (co)terms means that reduction is not internal for coderivations, i.e. if $t \rightsquigarrow s$ and t is a coderivation, s is not necessarily a coderivation. This is related to the fact that we, strictly speaking, distinguish cut and application. On the other hand it is this distinction that admits the aforementioned continuity property, as well as admitting the possibility of cut-free normal (co)terms. The precise relationship between normalisation of coterms and cut-elimination subsumes that for minimal logic, between normalisation for *natural deduction* (or simply typed λ-calculus) and cut-elimination in the sequent calculus, for which there is a substantial literature. See, e.g., [TS00] for a detailed account of such matters.

We should mention that it would make sense, for the pursuit of cut-elimination, to replace the s initial sequent with a corresponding rule, $\mathsf{s}\dfrac{\vec{\sigma} \Rightarrow N}{\vec{\sigma} \Rightarrow N}$, with the obvious interpretation, to admit an appropriate cut-reduction against the conditional rule, cond.

8.5. **Incorporating fixed point operators.** We could naturally extend our type system by (co)inductive fixed points, similarly to work such as [Cla10] for intuitionistic logic and [Bae12] for linear logic. Indeed, as we mentioned in the introduction, the circular proof theory of linear logic type systems with fixed points is increasingly well-developed, e.g. [BDS16, DS19, BDKS20].

At the level of expressivity, T already has the capacity to express a range of (co)inductive types. In particular, being the type theoretic counterpart to Peano Arithmetic, T admits recursion on effective well-orders of order type $< \varepsilon_0$ (see [Kre51, Tai68]), and so can encode (co)inductive types of closure ordinal $\omega^{\cdot^{\cdot^{\omega}}}$.[24]

Similarly to [BDS16], extensions of CT with (co)inductive types may be duly defined by demanding progressing threads on the LHS on least fixed points, or on the RHS on greatest fixed points. In terms of conservativity over T, the pertinent question is whether the encoding of (co)inductive types in T admits a corresponding coding of circular derivations, in particular preserving ancestry and the progressing thread criterion. This does not seem to be too technical, but a comprehensive treatment is beyond the scope of this work.

[24]Note, incidentally, that in a higher typed setting, coinductive types may themselves be encoded as inductive types.

However let us consider one pertinent example arising from the aforementioned work [KPP21], where a circular version of T was presented with a slightly different type language. In particular they include a *Kleene star* operator for list formation, along with the accompanying rules (ancestry indicated by colours):

$$*_l\ \frac{\vec{\sigma} \Rightarrow \tau \quad \vec{\sigma}, \sigma, \sigma^* \Rightarrow \tau}{\vec{\sigma}, \sigma^* \Rightarrow \tau} \qquad *_r\ \frac{}{\Rightarrow \tau^*} \qquad *_r\ \frac{\vec{\sigma} \Rightarrow \tau \quad \vec{\sigma} \Rightarrow \tau^*}{\vec{\sigma} \Rightarrow \tau^*}$$

The semantics of these constants are intuitive, and explained in [KPP21]. Being a least fixed point, the associated progressing thread condition is that each infinite branch has an infinite thread on a $*$-type on the LHS that is infinitely often principal. Semantically this induces a similar totality argument to ours for N-threads, cf. Proposition 35: at a progress point the corresponding *list* decreases in length.

As suggested above, we may duly encode the fixed point $*$, along with its rules and corresponding notion of progressing coderivation, within our type system for CT. First, temporarily using product types to ease the exposition, we may embed each type σ^* into a type $N \times (N \to \sigma)$.[25] Semantically the first component represents the length of the list, and the second component represents a stream from which the list is extracted, where we do not care about the values of elements beyond the length specified by the first component. Using this embedding we may duly derive the translations of the typing rules above. The first $*_r$ rule is translated to the following derivation:

$$\times_r\ \frac{0\ \dfrac{}{\Rightarrow N} \qquad \mathsf{R}\ \dfrac{\mathsf{wk}\ \dfrac{0_\tau\ \dfrac{}{\Rightarrow \tau}}{N \Rightarrow \tau}}{\Rightarrow N \to \tau}}{\Rightarrow N \times (N \to \tau)}$$

Here 0_τ is just the 0 function of type τ, and so we set the empty τ-list to be just the stream $(0_\tau, 0_\tau, \dots)$. Note that it would not matter if we set it to something else, since the first component tells us to ignore elements beyond the length, in this case the entire stream. The second $*_r$ rule is translated to the following derivation:

$$\mathsf{2cut}\ \frac{\vec{\sigma} \Rightarrow \tau \quad \vec{\sigma} \Rightarrow N \times (N \to \tau)}{\vec{\sigma} \Rightarrow N \times (N \to \tau)} \qquad \times_l\ \frac{\times_r\ \dfrac{\mathsf{s}\ \dfrac{}{N \Rightarrow N} \qquad \mathsf{R}\ \dfrac{\mathsf{cond}\ \dfrac{\mathsf{id}\ \dfrac{}{\tau \Rightarrow \tau} \qquad \mathsf{L}\ \dfrac{\mathsf{id}\ \dfrac{}{N \Rightarrow N} \qquad \mathsf{id}\ \dfrac{}{\tau \Rightarrow \tau}}{N \to \tau, N \Rightarrow \tau}}{\tau, N \to \tau, N \Rightarrow \tau}}{\tau, N \to \tau \Rightarrow N \to \tau}}{\tau, N, N \to \tau \Rightarrow N \times (N \to \tau)}}{\tau, N \times (N \to \tau) \Rightarrow N \times (N \to \tau)}$$

[25] Note that this embedding also induces a well-behaved notion of type level in the presence of Kleene $*$: we may set $\mathsf{lev}(\sigma^*) := \mathsf{lev}(\sigma)$, as long as $\mathsf{lev}(\sigma) > 0$. The type N^* may also be embedded at level 0 using a coding of sequences, but to preserve ancestry, similarly to the current encoding, we should nonetheless encode it as a pair $N \times N$, the first component still representing the length.

Finally the $*_l$ rule is translated to the following derivation:

$$\dfrac{\dfrac{\vec{\sigma} \Rightarrow \tau}{\vec{\sigma}, N \to \sigma \Rightarrow \tau}\,\mathsf{wk} \qquad \dfrac{\dfrac{0\ \dfrac{}{\Rightarrow N} \qquad \mathsf{id}\ \dfrac{}{\sigma \Rightarrow \sigma}}{N \to \sigma \Rightarrow \sigma}\,\mathsf{L} \qquad \dfrac{\dfrac{\dfrac{\mathsf{s}\ \dfrac{}{N \Rightarrow N} \qquad \mathsf{L}\ \dfrac{\mathsf{id}\ \dfrac{}{N \Rightarrow N} \qquad \mathsf{id}\ \dfrac{}{\sigma \Rightarrow \sigma}}{N \to \sigma, N \Rightarrow \sigma}}{N \to \sigma, N \Rightarrow \sigma}\,\mathsf{cut}}{N \to \sigma \Rightarrow N \to \sigma}\,\mathsf{R} \qquad \vec{\sigma}, \sigma, N, N \to \sigma \Rightarrow \tau}{\vec{\sigma}, \sigma, N, N \to \sigma \Rightarrow \tau}\,\mathsf{cut}}{\vec{\sigma}, N, N \to \sigma \Rightarrow \tau}\,\mathsf{cut}}{\vec{\sigma}, N, N \to \sigma \Rightarrow \tau}\,\mathsf{cond}$$

The verification of the semantics of these derivations is left as an exercise to the reader. Note that, as indicated by the colouring of type occurrences, ancestry is preserved by this translation. In particular, for the translation of σ^*, ancestry *and progressiveness* are preserved on the first component (the blue N) of a left rule.

It should be straightforward to formalise the ideas above to obtain an embedding of the system from [KPP21] into ours, thereby inheriting similar results, but a comprehensive development is beyond the scope of this work. It would be interesting, for future work, to more generally study extensions of CT by suitable fixed points, least and greatest, and in particular show that they may be interpreted back into CT using translations like the ones above. The key point above was to use the first component (length) to store a recursive parameter on which ancestry is preserved. For the case of Kleene $*$ this is straightforward since its closure ordinal is ω, however we must be more careful for fixed points with greater closure ordinals: while we can indeed code recursive ordinals by natural numbers, we must be able to implement 'circular recursion' on them in a way that preserves ancestry and progressiveness. Naturally, some use of higher types should be required, trading off ordinal complexity for abstraction complexity, cf. [Kre51, Tai68, Par72, Bus95].

8.6. **'Cyclic recursive functionals' are Gödel primitive recursive.** Besides the relationships between CT and T as theories and in the various models we have discussed, like HR and HE, it is natural to ask about their relationships in the standard model $\mathfrak{N}$. In this section we shall show that the interpretations of their terms in $\mathfrak{N}$ in fact comprise the same algebra of functionals, by reduction of CT computability to some form of higher type recursion on ordinals smaller than ε_0.

Recall that the Gödel primitive recursive functionals, PRF, are just the interpretations of T terms in $\mathfrak{N}$, i.e. $\{t^{\mathfrak{N}} \mid t \text{ a } T\text{-term}\}$. Now writing CRF ('cyclic recursive functionals') for the interpretations of CT terms in $\mathfrak{N}$, i.e. $\{t^{\mathfrak{N}} \mid t \text{ a } CT\text{-coterm}\}$, the main result of this subsection is:

Theorem 94. CRF = PRF. *I.e. for every CT coterm there is a T term computing the same functional (in $\mathfrak{N}$) and vice versa.*

Notice that the right-left inclusion follows readily from the encoding of primitive recursion from Example 23, so we concentrate on the left-right inclusion. The statement above could indeed be refined in terms of type level, as for the main results in this work, but we shall drop such a specialisation for the sake of brevity.

For the remainder of this section we shall work inside the standard model $\mathfrak{N}$. We employ (higher-order) recursion theoretic methods, and in particular take advantage of some well-known meta-recursion-theoretic results.

8.6.1. *Recursion schemes on well-founded relations.* Let $\lhd$ be a well-founded strict partial order on $\mathbb{N}$. We say that $f : N \to \tau$ is obtained from $g : N \to (N \to \tau) \to \tau$ by **recursion on** $\lhd$ (henceforth Rec($\lhd$)) if:

$$f(n) \;=\; g(n, \lambda z \lhd n.f(z)) \tag{29}$$

Here λ, as expected, refers to abstraction of arguments; the 'guarding' of the abstraction by $\lhd$ is formally defined as follows,

$$(\lambda z \lhd n.f(z))(m) \;:=\; \begin{cases} f(m) & \text{if } m \lhd n \\ 0_\tau & \text{otherwise} \end{cases}$$

where 0_τ is the 'zero' functional of type τ, setting $0_N := 0$ and $0_{\sigma\to\tau} : x \mapsto 0_\tau$.

We say that $\{f_i : N \to \tau_i\}_{i=1}^k$ are obtained by **simultaneous recursion on** $\lhd$ (henceforth SimRec($\lhd$)) from $\{h_i : N \to (N \to \tau_1) \to \cdots \to (N \to \tau_k)\}_{i=1}^k$ if:

$$f_i(n) \;=\; h_i(n, \lambda z \lhd n.f_1(z), \ldots, \lambda z \lhd n.f_k(z)) \tag{30}$$

The following is well-known:

Proposition 95 (Closure under simultaneous recursion)**.** *If* PRF *is closed under* Rec($\lhd$) *then* PRF *is also closed under* SimRec($\lhd$)*.*

Proof sketch. For simplicity, we shall make use of product types, thanks to usual primitive recursive (de)pairing operations (see, e.g., [Tro73] for more details). For x_i of type σ_i, we will write $\langle x_1, \ldots, x_k\rangle$ for an element of type $\sigma_1 \times \cdots \times \sigma_k$, and conversely for a list z of type $\sigma_1 \times \cdots \times \sigma_k$ we shall write $\pi_i(z)$ for the ith element of the list.

Let $f_1, \ldots, f_k$ be as above, satisfying (30), and define $f : N \to (\tau_1 \times \cdots \times \tau_k)$ by:

$$f(n) := \langle f_1(n), \ldots, f_k(n)\rangle$$

Note that we have,

$$\begin{aligned} f(n) &= \langle f_i(n)\rangle_{i=1}^k \\ &= \langle h_i(n, \lambda z \lhd n.f_1(z), \ldots, \lambda z \lhd n.f_k(z))\rangle_{i=1}^k \\ &= \langle h_i(n, \pi'_1(\lambda z \lhd n.f(z)), \ldots, \pi'_k(\lambda z \lhd n.f(z))))\rangle_{i=1}^k \end{aligned}$$

where $\pi'_j : (N \to (\tau_1 \times \cdots \times \tau_k)) \to N \to \tau_j$ by $\pi'_j(g, n) := \pi_j(g(n))$. This is an instance of Rec($\lhd$), so $f \in$ PRF. From here we indeed have, for $j = 1, \ldots, k$, that $f_j = \lambda n.(\pi_j(f(n))) \in$ PRF. □

8.6.2. *Closure under recursion on provably well-founded orders.* In this subsection we shall assume a standard primitive recursive representation of the ordinals up to ε_0 as natural numbers, written α, β etc., and the usual (strict) well-order on them, written $\prec$. Note that, while we may indeed represent $\prec$ primitive recursively in, say, RCA_0, we certainly cannot prove that it is a well-order on ε_0 even in ACA_0, being its proof-theoretic ordinal.

Recalling the recursion schemes of the previous subsubsection, we shall write simply Rec(α) for recursion on $\prec$ restricted to the initial segment α of ε_0, and Rec($\prec \varepsilon_0$) for the union of Rec(α) for $\alpha \prec \varepsilon_0$.

The following is a well-known result, originally due to Kreisel [Kre59] by means of Gödel's Dialectica functional interpretation (see also [Tai65, How80]):

Theorem 96 (Kreisel)**.** *If* $\alpha \prec \varepsilon_0$ *then* PRF *is closed under* Rec(α)*.*

We want to eventually reduce well-founded arguments on coderivations to some sort of recursion on ordinals $\prec \varepsilon_0$. For this, we shall exploit the notion of 'provably well-founded relations', going back to Gentzen:

Definition 97 ((Provably) inductive relations)**.** A relation $\lhd$ is **inductive** if the following holds: if whenever $\forall y \lhd x X(y)$ we have $X(x)$, then $\forall x X(x)$.

$\lhd$ is **provably inductive in** PA if,

$$\mathsf{PA}(X) \vdash \forall x(\forall y \lhd x X(y) \supset X(x)) \supset \forall x X(x)$$

where X is some fresh unary predicate symbol added to PA.

Gentzen already showed that any recursive well-order $\lhd$ on $\mathbb{N}$ provably inductive in PA has order type some $\alpha \prec \varepsilon_0$ [Gen43]. This result was arithmetised by Takeuti (and independently Harrington), who further showed the existence of an order-preserving embedding that is $\prec \varepsilon_0$-recursive (see [Tak87, FS95]), i.e. in PRF (by Kreisel's result above, Theorem 96). For our results we will need a generalisation of this result to well-founded *partial* orders, which are not necessarily total. Naturally every well-founded partial order can be order-preserving embedded into the ordinals, by well-founded induction, but, again, we need an arithmetised version of such a result to extract a suitable embedding in PRF. We expect that the particular well-founded partial orders we consider admit suitable arithmetisable linearisations, so as to directly apply Takeuti's and Harrington's result, but thankfully a more than suitable generalisation for well-founded partial orders has already been obtained by Arai:

Theorem 98 ([Ara98])**.** *Let $\lhd$ be a primitive recursive well-founded strict partial order on $\mathbb{N}$ that is provably inductive in* PA*. Then there is some $\alpha_\lhd \prec \varepsilon_0$ and some $\mu_\lhd : N \to N$ in* PRF *with $x \lhd y \implies \mu_\lhd(x) \prec \mu_\lhd(y) \prec \alpha_\lhd$.*

In fact Arai's result is much stronger: if $\lhd$ is elementary recursive, then so is $f_\lhd$, and the result above can actually be demonstrated within elementary recursive arithmetic, in particular $I\Delta_0 + \exp$. As it happens, the relations we shall consider (those induced by CT coderivations) will indeed all be elementary recursive, but we shall not need such a strengthening of Theorem 98.

Corollary 99. *Let $\lhd$ be as in Theorem 98 above. Then* PRF *is closed under* $\mathrm{Rec}(\lhd)$ *and* $\mathrm{SimRec}(\lhd)$*.*

Proof. By Proposition 95 it suffices to show closure under $\mathrm{Rec}(\lhd)$. Suppose $g : N \to (N \to \tau) \to \tau$ is in PRF and we will show that $f : N \to \tau$ is in PRF where:

$$f(n) \;=\; g(n, \lambda z \lhd n.f(z))$$

Letting $\mu_\lhd$ and $\alpha_\lhd \prec \varepsilon_0$ be as obtained by Theorem 98 above, let us write:

$$f'(\alpha, n) = \begin{cases} f(n) & \text{if } \mu(n) \prec \alpha \\ 0_\tau & \text{otherwise} \end{cases}$$

Notice that, as long as $\mu(n) \prec \alpha \prec \alpha_\lhd$, we have:

$$\begin{array}{rcl} f'(\alpha) & = & \lambda n.g(n, f'(\mu(n))) \\ & = & \lambda n.g(n, (\lambda \beta \prec \alpha.f'(\beta))(\mu(n))) \end{array}$$

This is an instance of $\mathrm{Rec}(\alpha_\lhd)$ and so $f' \in$ PRF by Theorem 96. From here we have $f(n) = f'(\mu_\lhd(n) + 1, n)$, and so indeed $f \in$ PRF, as required. $\square$

8.6.3. *A well-founded order on 'runs' of progressing coderivations.* For the remainder of this subsection, let us fix a CT coderivation t whose (finitely many) distinct sub-coderivations are $\{t_i : \vec{N}_i, \vec{\sigma}_i \Rightarrow \tau_i\}_{i=1}^n$, with all N occurrences indicated.

Let $T \subseteq \{0,1\}^*$ be the underlying (infinite) coderivation tree of t, and let $\{T_1, \dots, T_n\}$ partition T into the sets of nodes rooting $t_1, \dots, t_n$ respectively. Notice that all these sets are provably recursive (given t) in even RCA_0.

We define a binary relation $\twoheadrightarrow$ on $T \times \mathbb{N}^*$ (i.e. $\twoheadrightarrow \subseteq (T \times \mathbb{N}^*) \times (T \times \mathbb{N}^*)$) as follows. $(u, \vec{m}) \twoheadrightarrow (v, \vec{n})$ if:

- v is a child of u;
- u roots some t_i (i.e. $u \in T_i$) and v roots some t_j (i.e. $v \in T_j$).
- $|\vec{m}| = |\vec{N}_i|$ and $|\vec{n}| = |\vec{N}_j|$.
- if N_{jl} is an immediate ancestor of N_{ik} then $n_l \leq m_k$.
- if N_{ik} is principal for a cond step and N_{jl} is its immediate ancestor, then $n_l < m_k$.

We do not impose any other constraints on $\twoheadrightarrow$.

Example 100 (Examples of 'runs')**.** *Revisiting Example 23, suppose t is the coderivation on the RHS of* (7)*. We have:*

- $(\varepsilon, 6) \twoheadrightarrow (1, 5) \twoheadrightarrow (10, 5)$.
- $(\varepsilon, 6) \twoheadrightarrow (1, 3) \twoheadrightarrow (10, 2)$.
- $(\varepsilon, 6) \twoheadrightarrow (1, 2) \twoheadrightarrow (11, 8)$.

Revisiting Section 4.6, now suppose t is the coderivation from (10)*. We have:*

- $(\varepsilon, 5, 3) \twoheadrightarrow (0, 5, 5, 3) \twoheadrightarrow (00, 5, 3) \twoheadrightarrow (000, 3)$
- $(\varepsilon, 5, 3) \twoheadrightarrow (0, 5, 4, 3) \twoheadrightarrow (01, 4, 4, 3) \twoheadrightarrow (010, 4, 4) \twoheadrightarrow (0100, 4) \twoheadrightarrow (01001, 2)$
- $(\varepsilon, 5, 3) \twoheadrightarrow (0, 5, 5, 3) \twoheadrightarrow (01, 3, 5, 2) \twoheadrightarrow (011, 3, 4, 0) \twoheadrightarrow (0110, 4, 0)$
- $(\varepsilon, 5, 3) \twoheadrightarrow (0, 2, 4, 3) \twoheadrightarrow (01, 1, 4, 3) \twoheadrightarrow (011, 1, 4, 1) \twoheadrightarrow (0111, 1, 9)$

Note that $\twoheadrightarrow$ is clearly a polynomial-time recursive relation, and in particular is provably Δ^0_1 in even RCA_0. Since we have fixed t in advance, we may actually establish the well-foundedness of $\twoheadrightarrow$ within RCA_0:

Theorem 101 (RCA_0)**.** *$\twoheadrightarrow$ is terminating, i.e. $\forall f. \exists n. f(n) \not\twoheadrightarrow f(n+1)$.*

Proof sketch. Suppose, for contradiction, that $f : \mathbb{N} \to \mathbb{N}$ with $\forall n. f(n) \twoheadrightarrow f(n+1)$. Writing $f(n) = (u_n, \vec{m}_n)$, we have that $(u_n)_{n \in \omega}$ is (the tail of) a branch of t. By Proposition 34 (itself from [Das20]), RCA_0 proves that t is progressing, and so we have an infinitely progressing thread along $(u_n)_{n \in \omega}$. From this thread we can extract from $(\vec{m}_n)_{n \in \omega}$ a sequence of natural numbers $(m_{ni_n})_{n \in \omega}$ corresponding to the thread. However, by construction, $(m_{ni_n})_{n \in \omega}$ is monotone decreasing (by induction on n) and has no least element (since it follows a progressing thread). □

Let us note that the uniform version of the above result, quantifying over all CT coderivations t, requires $I\Sigma^0_2$, cf. [KMPS19a, Das20].

Corollary 102 (ACA_0)**.** *$\twoheadleftarrow$ is inductive, i.e. $(\forall x(\forall y \twoheadleftarrow x X(y) \supset X(x)) \supset \forall x X(x))$.*

Proof. Suppose, for contradiction, that $\neg X(n)$ and $\forall x(\forall y \twoheadleftarrow x X(y) \supset X(x))$, i.e. $\forall x(\neg X(x) \supset \exists y \twoheadleftarrow x \neg X(y))$. Since $\twoheadrightarrow$ is (provably) recursive, we may define n_i with $n_0 = n$ and n_{i+1} least such that $n_i \twoheadrightarrow n_{i+1}$ and $\neg X(n_{i+1})$, recursively in i. By comprehension we may have the graph of the function $f(i) := n_i$ for $i \in \omega$, contradicting termination, Theorem 101 above. □

Let us henceforth write $\overset{+}{\rightarrowtriangle}$ for the transitive closure of $\rightarrowtriangle$.

Corollary 103. PRF *is closed under* $\mathrm{Rec}(\overset{+}{\leftarrowtriangle})$*, and so also* $\mathrm{SimRec}(\overset{+}{\leftarrowtriangle})$.

Proof sketch. By Corollary 102 above, we immediately have that $\overset{+}{\leftarrowtriangle}$ is inductive, a fortiori, provably in ACA_0, and so also in $\mathsf{PA}(X)$ by conservativity. The result now follows by Corollary 99. □

8.6.4. *Main result.* We continue to work with the fixed regular coderivation t from the previous subsubsection and its distinct sub-coderivations $\{t_i : \vec{N}_i, \vec{\sigma}_i \Rightarrow \tau_i\}_{i=1}^n$.

Proof of Theorem 94. We shall assume some basic primitive recursive coding and decoding $\langle\cdot\rangle$ of lists, and suppress the explicit functions associated with it, namely those adding and extracting elements from lists.

We show that the functionals $\{g_i : N \to \vec{\sigma}_i \to \tau_i\}_{i=1}^n$ with $g_i(\langle u, \vec{m}\rangle) := t_i^{\mathfrak{N}}(\vec{m}_i)$, as long as $u \in T_i$, may be obtained by $\mathrm{SimRec}(\overset{+}{\leftarrowtriangle})$.

- If t_i has form $\mathsf{id}_N \dfrac{}{N \Rightarrow N}$ then $g_i(\langle u, m\rangle) = m$.
- If t_i has form $\mathsf{id}_\sigma \dfrac{}{\sigma \Rightarrow \sigma}$ with $\sigma \neq N$ then $g_i(\langle u\rangle) = \mathsf{id}_\sigma^{\mathfrak{N}}$.

Henceforth, we shall always assume that $\sigma \neq N$.

- If t_i has form $\mathsf{wk}_N \dfrac{\overset{t_j}{\vec{N}, \vec{\sigma} \Rightarrow \tau}}{\vec{N}, N, \vec{\sigma} \Rightarrow \tau}$ then:

$$\begin{aligned} g_i(\langle u, \vec{m}, m\rangle) &= g_j(\langle u0, \vec{m}\rangle) \\ &= (\lambda z \overset{+}{\leftarrowtriangle} \langle u, \vec{m}, m\rangle\, .g_j(z))(\langle u0, \vec{m}\rangle) \end{aligned}$$

Henceforth, we shall simply write recursive calls more compactly as $g_j(\langle u0, \vec{m}_0\rangle)$ or $g_k(\langle u1, \vec{m}_1\rangle)$ when $\langle u0, \vec{m}_0\rangle \overset{+}{\leftarrowtriangle} \langle u, \vec{m}\rangle$ and $\langle u1, \vec{m}_1\rangle \overset{+}{\leftarrowtriangle} \langle u, \vec{m}\rangle$, rather than fully writing $(\lambda z \overset{+}{\leftarrowtriangle} \langle u, \vec{m}\rangle\, .g_j(z))(\langle u0, \vec{m}_0\rangle)$ or $(\lambda z \overset{+}{\leftarrowtriangle} \langle u, \vec{m}\rangle\, .g_k(z))(\langle u1, \vec{m}_1\rangle)$.

- If t_i has form $\mathsf{wk} \dfrac{\overset{t_j}{\vec{N}, \vec{\sigma} \Rightarrow \tau}}{\vec{N}, \vec{\sigma}, \sigma \Rightarrow \tau}$ then $g_i(\langle u, \vec{m}\rangle) = \mathsf{wk}^{\mathfrak{N}}(g_j(\langle u0, \vec{m}\rangle))$, where the wk constant has the appropriate type, i.e. $(\vec{\sigma} \to \tau) \to \vec{\sigma} \to \sigma \to \tau$.

Henceforth, we shall omit the types of the constants we use, being determined by the context in which it appears.

- If t_i has form $\mathsf{cntr}_N \dfrac{\overset{t_j}{\vec{N}, N, N, \vec{\sigma} \Rightarrow \tau}}{\vec{N}, N, \vec{\sigma} \Rightarrow \tau}$ then $g_i(\langle u, \vec{m}, m\rangle) = g_j(\langle u0, \vec{m}, m, m\rangle)$.

- If t_i has form $\mathsf{cntr}\dfrac{\overset{t_j}{\vec{N},\vec{\sigma},\sigma,\sigma\Rightarrow\tau}}{\vec{N},\vec{\sigma},\sigma\Rightarrow\tau}$ then $g_i(\langle u,\vec{m}\rangle)=\mathsf{cntr}^{\mathfrak{N}}(g_j(\langle u0,\vec{m}\rangle))$.

- If t_i has form $\mathsf{cut}_N\dfrac{\overset{t_j}{\vec{N},\vec{\sigma}\Rightarrow N}\quad\overset{t_k}{\vec{N},N,\vec{\sigma}\Rightarrow\tau}}{\vec{N},\vec{\sigma}\Rightarrow\tau}$ then

$$g_i(\langle u,\vec{m}\rangle)=\lambda\vec{x}.g_k(\langle u1,\vec{m},g_j(\langle u0,\vec{m}\rangle,\vec{x})\rangle,\vec{x})$$

- If t_i has form $\mathsf{cut}\dfrac{\overset{t_j}{\vec{N},\vec{\sigma}\Rightarrow\sigma}\quad\overset{t_k}{\vec{N},\vec{\sigma},\sigma\Rightarrow\tau}}{\vec{N},\vec{\sigma}\Rightarrow\tau}$ then

$$g_i(\langle u,\vec{m}\rangle)=\mathsf{cut}^{\mathfrak{N}}(g_j(\langle u0,\vec{m}\rangle),g_k(\langle u1,\vec{m}\rangle))$$

- If t_i has form $\mathsf{L}_N\dfrac{\overset{t_j}{\vec{N},\vec{\sigma}\Rightarrow\rho}\quad\overset{t_k}{\vec{N},N,\vec{\sigma}\Rightarrow\tau}}{\vec{N},\vec{\sigma},\rho\to N\Rightarrow\tau}$ then

$$g_i(\langle u,\vec{m}\rangle)=\lambda\vec{x}.\lambda h.(g_k(\langle u1,\vec{m},h(g_j(\langle u0,\vec{m}\rangle,\vec{x}))\rangle,\vec{x}))$$

- If t_i has form $\mathsf{L}\dfrac{\overset{t_j}{\vec{N},\vec{\sigma}\Rightarrow\rho}\quad\overset{t_k}{\vec{N},\vec{\sigma},\sigma\Rightarrow\tau}}{\vec{N},\vec{\sigma},\rho\to\sigma\Rightarrow\tau}$ then

$$g_i(\langle u,\vec{m}\rangle)=\mathsf{L}^{\mathfrak{N}}(g_j(\langle u0,\vec{m}\rangle),g_k(\langle u1,\vec{m}\rangle))$$

- If t_i has form $\mathsf{R}_N\dfrac{\overset{t_j}{\vec{N},N,\vec{\sigma}\Rightarrow\tau}}{\vec{N},\vec{\sigma}\Rightarrow N\to\tau}$ then $g_i(\langle u,\vec{m}\rangle)=\lambda\vec{x}.\lambda m.g_j(\langle u0,\vec{m},m\rangle,\vec{x})$

- If t_i has form $\mathsf{R}\dfrac{\overset{t_j}{\vec{N},\vec{\sigma},\sigma\Rightarrow\tau}}{\vec{N},\vec{\sigma}\Rightarrow\sigma\to\tau}$ then $g_i(\langle u,\vec{m}\rangle)=\mathsf{R}^{\mathfrak{N}}(g_j(\langle u0,\vec{m}\rangle))$

- If t_i has form $0\dfrac{}{\Rightarrow N}$ then $g_i(\langle u\rangle)=0$.

- If t_i has form $\mathsf{s}\dfrac{}{N\Rightarrow N}$ then $g_i(\langle u,m\rangle)=\mathsf{s}m$.

- If t_i has form $\mathsf{cond}\dfrac{\overset{t_j}{\vec{N},\vec{\sigma}\Rightarrow\tau} \quad \overset{t_k}{\vec{N},N,\vec{\sigma}\Rightarrow\tau}}{\vec{N},N,\vec{\sigma}\Rightarrow\tau}$ then

$$g_i(\langle u,\vec{m},m\rangle) \;=\; \begin{cases} g_j(\langle u0,\vec{m}\rangle) & \text{if } m=0 \\ g_k(\langle u1,\vec{m},m-1\rangle) & \text{otherwise} \end{cases}$$

  where the conditional "if then else" operation is obtained, as usual, by primitive recursion.

From here we have $t_i^{\mathfrak{N}}(\vec{m},\vec{x}) = g_i(\langle u,\vec{m}\rangle,\vec{x})$, for some $u \in T_i$ (say the least one), so $t_i^{\mathfrak{N}} \in \mathrm{PRF}$ for $i = 1,\ldots,n$. Since our initial CT coderivation t is just some t_i, we thus indeed have $t^{\mathfrak{N}} \in \mathrm{PRF}$, so in particular $t^{\mathfrak{N}}$ is computed by a term of T. □

9. Conclusions

In this work we presented a circular version CT of Gödel's system T and investigated its expressivity at the level of abstraction complexity (i.e. type level). To this end, we showed that CT_n and T_{n+1} have the same logical and recursion-theoretic strength, by means of interpretations in each direction, over at least the type 1 quantifier-free theory.

We also gave several further results about the coterms of CT, for instance models of hereditarily computable functions, continuity at type 2, strong normalisation and confluence, and a translation to terms T computing the same funtional, at all types.

As mentioned in the Introduction, our ultimate motivation is to bring one of the hallmarks of 20th century proof theory to the setting of non-wellfounded proofs: a bona fide correspondence between theories in predicate logic and type systems for functional programming languages. The obtention of a Dialectica-style proof interpretation, cf. [Gö58], between circular versions of arithmetic [Sim17, BT17b, Das20] and the theory CT here presented is thus the natural next step.

References

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

[AL17] 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.

[Ara98] Toshiyasu Arai. Some results on cut-elimination, provable well-orderings, induction and reflection. *Annals of Pure and Applied Logic*, 95(1):93 – 184, 1998.

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

[BDKS20] David Baelde, Amina Doumane, Denis Kuperberg, and Alexis Saurin. Bouncing threads for infinitary and circular proofs. *CoRR*, abs/2005.08257, 2020.

[BDP11] 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.

[BDS16] 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.

[BGP12] 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.

[BS07] 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.

[BS11] James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *J. Log. Comput.*, 21(6):1177–1216, 2011.

[BT17a] Stefano Berardi and Makoto Tatsuta. Classical system of Martin-Löf's inductive definitions is not equivalent to cyclic proof system. In *Foundations of Software Science and Computation Structures - 20th International Conference, FOSSACS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, pages 301–317, 2017.

[BT17b] 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.

[Bus95] Samuel R. Buss. The witness function method and provably recursive functions of Peano arithmetic. In *Studies in Logic and the Foundations of Mathematics*, volume 134, pages 29–68. Elsevier, 1995.

[Bus98] Samuel R. Buss, editor. *Handbook of Proof Theory*. Studies in Logic and the Foundations of Mathematics 137. Elsevier, 1998.

[CH72] Seldin J.P Curry H.B., Hindley J.R. *Combinatory logic*, volume Vol. 2 of *Studies in logic and the foundations of mathematics*. North-Holland Pub. Co, 1972.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5(2):56–68, 1940.

[Cla10] Pierre Clairambault. Least and greatest fixpoints in game semantics. 01 2010.

[CR20] Liron Cohen and Reuben N. S. Rowe. Non-well-founded proof theory of transitive closure logic. *ACM Trans. Comput. Log.*, 21(4):31:1–31:31, 2020.

[Das20] Anupam Das. On the logical complexity of cyclic arithmetic. *Log. Methods Comput. Sci.*, 16(1), 2020.

[DDP18] 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.

[DHL06a] 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.

[DHL06b] 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.

[Dou17] 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.

[DP17] 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.

[DP18] 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.

[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.

[FS95] Harvey Friedman and Michael Sheard. Elementary descent recursion and proof theory. *Ann. Pure Appl. Log.*, 71(1):1–45, 1995.

[FS13] 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.

[Gen43] G. Gentzen. Beweisbarkeit und unbeweisbarkeit von anfangsfällen der transfiniten induktion in der reinen zahlentheorie. *Mathematische Annalen*, 119:140–161, 1943.

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

[Hir14] Denis R. Hirschfeldt. *Slicing the truth: On the computable and reverse mathematics of combinatorial principles*. World Scientific, 2014.

[How80] W. A. Howard. Ordinal analysis of terms of finite type. *The Journal of Symbolic Logic*, 45(3):493–504, 1980.

[HP93] Petr Hájek and Pavel Pudlák. *Metamathematics of First-Order Arithmetic*. Perspectives in mathematical logic. Springer, 1993.

[HS86] J. Roger Hindley and Jonathan P. Seldin. *Introduction to Combinators and λ-Calculus*. Cambridge University Press, USA, 1986.

[Kle80] S.C. Kleene. *Introduction to Metamathematics*. Bibliotheca Mathematica. North Holland, 7 edition, 1980.

[KMPS19a] 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.

[KMPS19b] Leszek Kołodziejczyk, Henryk Michalewski, Pierre Pradic, and Michał Skrzypczak. The logical strength of Büchi's decidability theorem. volume Volume 15, Issue 2, May 2019.

[Koh08] Ulrich Kohlenbach. *Applied Proof Theory - Proof Interpretations and their Use in Mathematics*. Springer Monographs in Mathematics. Springer, 2008.

[Koz83] Dexter Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[KPP21] Denis Kuperberg, Laureline Pinault, and Damien Pous. Cyclic Proofs, System T, and the Power of Contraction. *Proceedings of the ACM on Programming Languages*, 2021.

[Kre51] Georg Kreisel. On the interpretation of non-finitist proofs - part I. *J. Symb. Log.*, 16(4):241–267, 1951.

[Kre52] Georg Kreisel. On the interpretation of non-finitist proofs: Part II. interpretation of number theory. applications. *J. Symb. Log.*, 17(1):43–58, 1952.

[Kre59] Georg Kreisel. Interpretation of analysis by means of constructive functionals of finite types. In A. Heyting, editor, *Constructivity in Mathematics*, pages 101–128. Amsterdam: North-Holland Pub. Co., 1959.

[KS11] Jeroen Ketema and Jakob Grue Simonsen. Infinitary combinatory reduction systems. *Inf. Comput.*, 209(6):893–926, June 2011.

[KSSdV05] Richard Kennaway, Paula Severi, M. Ronan Sleep, and Fer-Jan de Vries. Infinitary rewriting: From syntax to semantics. In Aart Middeldorp, Vincent van Oostrom, Femke van Raamsdonk, and Roel C. de Vrijer, editors, *Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop, on the Occasion of His 60th Birthday*, volume 3838 of *Lecture Notes in Computer Science*, pages 148–172. Springer, 2005.

[Kuz19] Stepan Kuznetsov. The logic of action lattices is undecidable. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–9. IEEE, 2019.

[Lag09] Ugo Dal Lago. The geometry of linear higher-order recursion. *ACM Trans. Comput. Log.*, 10(2):8:1–8:38, 2009.

[LN15] John Longley and Dag Normann. *Higher-Order Computability*. Theory and Applications of Computability. Springer, 2015.

[Luc73] Horst Luckhardt. *Extensional Godel Functional Interpretation*. Lecture Notes in Mathematics. Springer, 1 edition, 1973.

[Min78] G. Mints. Finite investigations of transfinite derivations. *Journal of Soviet Mathematics*, 10:548–596, 1978.

[NW96] Damian Niwinski and Igor Walukiewicz. Games for the mu-calculus. *Theor. Comput. Sci.*, 163(1&2):99–116, 1996.

[Par72] Charles Parsons. On n-quantifier induction. *The Journal of Symbolic Logic*, 37(3):466–482, 1972.

[Pfe92] 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.

[RB17] Reuben N. S. Rowe and James Brotherston. Realizability in cyclic proof: Extracting ordering information for infinite descent. In *Automated Reasoning with Analytic Tableaux and Related Methods - 26th International Conference, TABLEAUX 2017, Brasília, Brazil, September 25-28, 2017, Proceedings*, pages 295–310, 2017.

[Sca71] B. Scarpellini. A model for barrecursion of higher types. *Compositio Mathematica*, 23(1):123–153, 1971.

[Sch75] Helmut Schwichtenberg. Elimination of higher type levels in definitions of primitive recursive functionals by means of transfinite recursion. In H.E. Rose and J.C. Shepherdson, editors, *Logic Colloquium '73*, volume 80 of *Studies in Logic and the Foundations of Mathematics*, pages 279 – 303. Elsevier, 1975.

[Sim09] Stephen G. Simpson. *Subsystems of second order arithmetic*, volume 1. Cambridge University Press, 2009.

[Sim12] Jakob Grue Simonsen. *Rewriting the Finite and the Infinite: Disputats for the Degree of Doctor Scientiarum at the University of Copenhagen*. PhD thesis, 2012.

[Sim17] 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.

[Stu08] Thomas Studer. On the proof theory of the modal mu-calculus. *Stud Logica*, 89(3):343–363, 2008.

[Tai65] W.W. Tait. Infinitely long terms of transfinite type. In J.N. Crossley and M.A.E. Dummett, editors, *Formal Systems and Recursive Functions*, volume 40 of *Studies in Logic and the Foundations of Mathematics*, pages 176 – 185. Elsevier, 1965.

[Tai67] William W. Tait. Intensional interpretations of functionals of finite type I. *J. Symb. Log.*, 32(2):198–212, 1967.

[Tai68] W.W. Tait. Constructive reasoning. In B. Van Rootselaar and J.F. Staal, editors, *Logic, Methodology and Philosophy of Science III*, volume 52 of *Studies in Logic and the Foundations of Mathematics*, pages 185 – 199. Elsevier, 1968.

[Tak87] Gaisi Takeuti. *Proof theory*. Studies in Logic and the Foundations of Mathematics 81. North-Holland, 2nd ed edition, 1987.

[Tak95] M. Takahashi. Parallel reductions in λ-calculus. *Information and Computation*, 118(1):120 – 127, 1995.

[Ter03] Terese. *Term rewriting systems.*, volume 55 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 2003.

[Tro73] Anne S. Troelstra. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*. Lecture Notes in Mathematics. Springer, 1 edition, 1973.

[TS00] Anne Sjerp Troelstra and Helmut Schwichtenberg. *Basic proof theory, Second Edition*, volume 43 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 2000.

[Xu20] Chuangjie Xu. A syntactic approach to continuity of T-definable functionals. *Logical Methods in Computer Science*, Volume 16, Issue 1, February 2020.

Appendix A. Partial cut-elimination for T

In this section we prove a (presumably folklore) result that derivations of T can be partially normalised so that all types occurring have level dominated by that of a recursor (cf. Proposition 16). It is well-known that the complexity of cut-elimination (at the level of provability of sequents) is 'superexponential' (see, e.g., [HP93]), so this argument should go through already inside RCA_0 (and even weaker theories), though we stop short of giving a complexity-theoretic analysis to this effect, so as not to complicate the exposition.

A.1. **Adapting the system for cut-elimination.** For simplicity, we no longer consider the exchange rule, but rather close all the instances of the other rules by composition with exchange, making formal how we were already informally typesetting rules in the main body of this paper. The axiomatisation of such combinators and their interpretations in the standard model $\mathfrak{N}$ are as expected, and we do not formalise this in detail.

We shall also assume that all identity initial sequents are in atomic form. Formally, we define id'_N as id_N, and inductively define $\mathsf{id}'_{\sigma\to\tau}$ as below, left. Clearly $\mathsf{id}'_N = \mathsf{id}_N$, and we give an argument by induction on type that $\mathsf{id}'_{\sigma\to\tau} = \mathsf{id}_{\sigma\to\tau}$ below, right (under ER):

$$\dfrac{\dfrac{\overset{\mathsf{id}'_\sigma}{\sigma \Rightarrow \sigma} \quad \overset{\mathsf{id}'_\tau}{\tau \Rightarrow \tau}}{\sigma \to \tau, \sigma \Rightarrow \tau}\,\mathsf{L}}{\sigma \to \tau \Rightarrow \sigma \to \tau}\,\mathsf{R}$$

$$\begin{array}{lcll} \mathsf{id}'_{\sigma\to\tau}\, x\, y & = & \mathsf{R}\,(\mathsf{L}\,\mathsf{id}'_\sigma \mathsf{id}'_\tau)\, x\, y & \text{by definition of } \mathsf{id}' \\ & = & \mathsf{L}\,\mathsf{id}'_\sigma \mathsf{id}'_\tau x\, y & \text{by } \mathsf{R} \text{ axiom} \\ & = & \mathsf{id}'_\tau(x\,(\mathsf{id}'_\sigma y)) & \\ & = & x\, y & \text{by inductive hypotheses} \\ & = & \mathsf{id}_{\sigma\to\tau}\, x\, y & \text{by } \mathsf{id} \text{ axiom} \end{array}$$

Furthermore, to simplify the termination argument for cut-elimination, we shall admit contraction by absorbing it into the other rules. For this, we shall need the following variant L' of the left-arrow rule, and its corresponding axioms:

$$\mathsf{L}'\,\dfrac{\vec{\sigma}, \rho \to \sigma \Rightarrow \rho \quad \vec{\sigma}, \rho \to \sigma, \sigma \Rightarrow \tau}{\vec{\sigma}, \rho \to \sigma \Rightarrow \tau} \qquad \mathsf{L}'\, s\, t\, \vec{x}\, y \;=\; t\, \vec{x}\, y\, (y\, (s\, \vec{x}\, y)) \tag{31}$$

Note that this rule is easily derivable in T, along with its corresponding equation, by combining the L and cntr rules:

$$\begin{array}{c} \mathsf{L}\,\dfrac{\vec{\sigma}, \rho \to \sigma \Rightarrow \rho \quad \vec{\sigma}, \rho \to \sigma, \sigma \Rightarrow \tau}{\mathsf{cntr}\,\dfrac{\vec{\sigma}, \rho \to \sigma, \underline{\rho \to \sigma} \Rightarrow \tau}{\vec{\sigma}, \underline{\rho \to \sigma} \Rightarrow \tau}} \end{array} \qquad \begin{array}{lcl} & & \mathsf{cntr}\,(\mathsf{L}\, s\, t)\, \vec{x}\, y \\ & = & \mathsf{L}\, s\, t\, \vec{x}\, y\, y \\ & = & t\, \vec{x}\, y (y\, (s\, \vec{x}\, y)) \end{array}$$

We may freely use L' in derivations, understanding it to be shorthand for the derivation above. A *contraction-free* derivation is one whose only cntr-steps are already part of L' steps.

Proposition 104. *For every T derivation $t : \vec{\sigma} \Rightarrow \tau$ there is a contraction-free derivation $t' : \vec{\sigma} \Rightarrow \tau$, possibly with more cut_N-occurrences, s.t. $T \vdash t = t'$.*

Proof sketch. Follows by a straightforward induction on the structure of t, commuting cntr steps over those above. The most interesting commutation is when a cntr

step is immediately below a rec step, for which we must introduce some N-cuts:

$$\mathsf{rec}\,\dfrac{\overset{s}{\triangledown}\quad \vec{\sigma}, N \Rightarrow \tau \qquad \overset{t}{\triangledown}\quad \vec{\sigma}, N, N, \tau \Rightarrow \tau}{\mathsf{cntr}\,\dfrac{\vec{\sigma}, N, \underline{N} \Rightarrow \tau}{\vec{\sigma}, N \Rightarrow \tau}} \quad\mapsto\quad \mathsf{rec}\,\dfrac{\mathsf{cut}\,\dfrac{0\,\dfrac{}{\Rightarrow N} \qquad \overset{s}{\triangledown}\quad \vec{\sigma}, N \Rightarrow \tau}{\vec{\sigma} \Rightarrow \tau} \qquad \mathsf{cut}\,\dfrac{\mathsf{s}\,\dfrac{}{N \Rightarrow N} \qquad \overset{t}{\triangledown}\quad \vec{\sigma}, N, N, \tau \Rightarrow \tau}{\vec{\sigma}, N, \tau \Rightarrow \tau}}{\vec{\sigma}, N \Rightarrow \tau}$$

We can verify that the derivation after transformation is provably equivalent in T by an object-level induction on the recursion parameter:

$$\begin{aligned}
\mathsf{rec}\,(\mathsf{cut}\,0\,s)\,(\mathsf{cut}\,\mathsf{s}\,t)\,\vec{x}\,0 &= \mathsf{cut}\,0\,s\,\vec{x} && \text{by } \mathsf{rec} \text{ axioms}\\
&= s\,\vec{x}\,0 && \text{by } \mathsf{cut} \text{ axiom}\\
&= \mathsf{rec}\,s\,t\,\vec{x}\,0\,0 && \text{by } \mathsf{rec} \text{ axioms}\\
&= \mathsf{cntr}\,(\mathsf{rec}\,s\,t)\,\vec{x}\,0 && \text{by } \mathsf{cntr} \text{ axiom}\\
\mathsf{rec}\,(\mathsf{cut}\,0\,s)\,(\mathsf{cut}\,\mathsf{s}\,t)\,\vec{x}\,\mathsf{s}y &= \mathsf{cut}\,\mathsf{s}\,t\,\vec{x}\,y\,(\mathsf{rec}\,(\mathsf{cut}\,0\,s)\,(\mathsf{cut}\,\mathsf{s}\,t)\,\vec{x}\,y) && \text{by } \mathsf{rec} \text{ axioms}\\
&= \mathsf{cut}\,\mathsf{s}\,t\,\vec{x}\,y\,(\mathsf{cntr}\,(\mathsf{rec}\,s\,t)\,\vec{x}\,y) && \text{by inductive hypothesis}\\
&= t\,\vec{x}\,\mathsf{s}y\,y\,(\mathsf{cntr}\,(\mathsf{rec}\,s\,t)\,\vec{x}\,y) && \text{by } \mathsf{cut} \text{ axiom}\\
&= \mathsf{rec}\,s\,t\,\vec{x}\,\mathsf{s}y\,\mathsf{s}y && \text{by } \mathsf{rec} \text{ axioms}\\
&= \mathsf{cntr}\,(\mathsf{rec}\,s\,t)\,\vec{x}\,\mathsf{s}y && \text{by } \mathsf{cntr} \text{ axiom}\quad \square
\end{aligned}$$

We shall also use a generalised version of the cut rule that incorporates both context sharing and context splitting behaviour:

$$(32)\qquad \mathsf{cut}'\,\dfrac{\vec{\rho}_0, \vec{\sigma}, \sigma \Rightarrow \tau \qquad \vec{\rho}_1, \vec{\sigma} \Rightarrow \tau}{\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma} \Rightarrow \sigma} \qquad \mathsf{cut}'s\,t\,\vec{x}_0\,\vec{x}_1\,\vec{y} \;=\; t\,\vec{x}_1\,\vec{y}\,(s\,\vec{x}_0\,\vec{y})$$

The point of this rule is to absorb extraneous weakening steps from cut-reductions into cuts. Note that we have been implicitly using this until now, e.g. as in the derivation in the proof sketch of Proposition 104. Similarly to L', the rule cut' and its corresponding equation can be derived from cut and wk:

$$\mathsf{cut}\,\dfrac{\mathsf{wk}\,\dfrac{\vec{\rho}_0, \vec{\sigma} \Rightarrow \sigma}{\vec{\rho}_0, \underline{\vec{\rho}_1}, \vec{\sigma} \Rightarrow \sigma} \qquad \mathsf{wk}\,\dfrac{\vec{\rho}_1, \vec{\sigma} \Rightarrow \tau}{\underline{\vec{\rho}_0}, \vec{\rho}_1, \vec{\sigma}, \sigma \Rightarrow \tau}}{\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma} \Rightarrow \sigma} \qquad \begin{aligned} & \mathsf{cut}\,(\mathsf{wk}^*s)\,(\mathsf{wk}^*t)\,\vec{x}_0\,\vec{x}_1\,\vec{y}\\ =\;& \mathsf{wk}^*t\,\vec{x}_0\,\vec{x}_1\,\vec{y}(\mathsf{wk}^*s\,\vec{x}_0\,\vec{x}_1\,\vec{y})\\ =\;& t\,\vec{x}_1\,\vec{y}\,(\mathsf{wk}^*s\,\vec{x}_0\,\vec{x}_1\,\vec{y})\\ =\;& t\,\vec{x}_1\,\vec{y}\,(s\,\vec{x}_0\,\vec{y}) \end{aligned}$$

Again, we may freely use cut' in derivations, understanding it to be shorthand for the derivation above. However, crucially, we will only count instances of L' and cut' as single steps, in order to facilitate the upcoming induction arguments.[26]

A.2. **Main free-cut elimination argument.** As usual, our overall argument will be by an induction on the complexity of cut-formulas:

- The *level* of a cut' on σ is just $\mathsf{lev}(\sigma)$.
- The d-level of a derivation is the multiset of all its cut-levels $> d$.

[26]Another approach could be to ignore all wk and cntr steps ('weak inferences') when counting the size/depth of derivations, but then we would need a lemma for commuting above those steps in order to reduce an induction measure based on size/depth.

We assume that multisets of natural numbers are ordered in the usual way.[27] We may write cut'_d, $\mathsf{cut}'_{\leq d}$ and $\mathsf{cut}'_{>d}$ for a cut' instance of level d, $\leq d$ or $> d$ respectively.

Lemma 105. *Let $t : \vec{\sigma} \Rightarrow \tau$ be a contraction-free T_n-derivation with at least one $\mathsf{cut}'_{>n}$. Then there is a contraction-free T_n-derivation $t' : \vec{\sigma} \Rightarrow \tau$ of lower n-level such that $T_n \vdash t = t'$.*

Proof. We proceed by induction on $|t|$, by inspection of a topmost $\mathsf{cut}_{>n}$. All permutation cases are given in the subsequent subsections, as well as arguments verifying that the derivations before and after transformation are provably equivalent in T_n. In each case, we may have to apply the induction hypothesis zero, one or two times to smaller subderivations. Note that the only transformation that introduces more cuts (even after applying inductive hypotheses) is the R-L' 'key' case. □

Proposition 16 is now an immediate consequence of the following result:

Theorem 106. *For any T_n-derivation $t : \vec{\sigma} \Rightarrow \tau$ there is a $\mathsf{cut}'_{>n}$-free T_n-derivation $t' : \vec{\sigma} \Rightarrow \tau$, such that $T \vdash t'\vec{x} \;=\; t\,\vec{x}$.*

Proof. Assume t is contraction-free by Proposition 104. The result follows by induction on the n-level of t, applying Lemma 105 above for the inductive steps. □

A.3. R-L' **key case.**

$$\mathsf{cut}'\,\dfrac{\mathsf{R}\,\dfrac{\overset{s}{\vec{\rho}_0, \vec{\sigma}, \rho \Rightarrow \sigma}}{\vec{\rho}_0, \vec{\sigma} \Rightarrow \rho \to \sigma} \qquad \mathsf{L}'\,\dfrac{\overset{r}{\vec{\rho}_1, \vec{\sigma}, \rho \to \sigma \Rightarrow \rho} \qquad \overset{t}{\vec{\rho}_1, \vec{\sigma}, \rho \to \sigma, \sigma \Rightarrow \tau}}{\vec{\rho}_1, \vec{\sigma}, \rho \to \sigma \Rightarrow \tau}}{\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma} \Rightarrow \tau}$$

is transformed to:

$$\mathsf{cut}\,\dfrac{\mathsf{cut}'\,\dfrac{\mathsf{cut}'\,\dfrac{\overset{s'}{\vec{\rho}_0, \vec{\sigma} \Rightarrow \rho \to \sigma} \qquad \overset{r}{\vec{\rho}_1, \vec{\sigma}, \rho \to \sigma \Rightarrow \rho}}{\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma} \Rightarrow \rho} \qquad \overset{s}{\vec{\rho}_0, \vec{\sigma}, \rho \Rightarrow \sigma}}{\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma} \Rightarrow \sigma} \qquad \mathsf{cut}'\,\dfrac{\overset{s'}{\vec{\rho}_0, \vec{\sigma} \Rightarrow \rho \to \sigma} \qquad \overset{t}{\vec{\rho}_1, \vec{\sigma}, \rho \to \sigma, \sigma \Rightarrow \tau}}{\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma}, \sigma \Rightarrow \tau}}{\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma} \Rightarrow \tau}$$

where we write s' for the derivation

$$\mathsf{R}\,\dfrac{\overset{s}{\vec{\rho}_0, \vec{\sigma}, \rho \Rightarrow \sigma}}{\vec{\rho}_0, \vec{\sigma} \Rightarrow \rho \to \sigma}$$

[27] It is well-known that for any well-order $<$ of order type α, the corresponding multiset order has order type ω^α. Thus the multiset order on even ω is already not available in RCA_0, whose proof-theoretic ordinal is ω^ω. However, cut-elimination arguments based on the multiset ordering are nonetheless typically formalisable in RCA_0 thanks to explicit complexity bounds on the multiset branching, i.e. the number of smaller number occurrences that replace a larger one when reducing.

We verify that T_n proves the equality of the derivations before and after transformation as follows:

$$\begin{array}{lll}
& \mathsf{cut}\,(\mathsf{cut}'(\mathsf{cut}'s'r)\,s)\,(\mathsf{cut}'s't)\,\vec{x}_0\,\vec{x}_1\,\vec{y} & \\
= & \mathsf{cut}'s't\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,(\mathsf{cut}'(\mathsf{cut}'s'r)\,s\,\vec{x}_0\,\vec{x}_1\,\vec{y}) & \text{by } \mathsf{cut} \text{ axiom} \\
= & \mathsf{cut}'s't\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,(s\,\vec{x}_0\,\vec{y}\,(\mathsf{cut}'s'r\,\vec{x}_0\,\vec{x}_1\,\vec{y})) & \text{by } \mathsf{cut}' \text{ axiom} \\
= & \mathsf{cut}'s't\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,(s\,\vec{x}_0\,\vec{y}\,(r\,\vec{x}_1\,\vec{y}\,(s'\vec{x}_0\,\vec{y}))) & \text{by } \mathsf{cut}' \text{ axiom} \\
= & t\,\vec{x}_1\,\vec{y}\,(s'\vec{x}_0\,\vec{y})\,(s\,\vec{x}_0\,\vec{y}\,(r\,\vec{x}_1\,\vec{y}\,(s'\vec{x}_0\,\vec{y}))) & \text{by } \mathsf{cut}' \text{ axiom} \\
= & t\,\vec{x}_1\,\vec{y}\,(s'\vec{x}_0\,\vec{y})\,(s'\vec{x}_0\,\vec{y}\,(r\,\vec{x}_1\,\vec{y}\,(s'\vec{x}_0\,\vec{y}))) & \text{by } \mathsf{R} \text{ axiom} \\
= & \mathsf{L}\,r\,t\,\vec{x}_1\,\vec{y}\,(s'\vec{x}_0\,\vec{y}) & \text{by } \mathsf{L} \text{ axiom} \\
= & \mathsf{cut}'s'(\mathsf{L}\,r\,t)\,\vec{x}_0\,\vec{x}_1\,\vec{y} & \text{by } \mathsf{cut}' \text{ axiom}
\end{array}$$

A.4. cut'**-rec commutative case.** The most interesting commutative case is commuting a $\mathsf{cut}'_{>n}$ above a $\mathsf{rec}_{\leq n}$ step:

$$\mathsf{cut}'\dfrac{\overset{r}{\triangledown}\ \vec{x}_0,\vec{\sigma},N\Rightarrow\sigma \qquad \mathsf{rec}_\tau\dfrac{\overset{s}{\triangledown}\ \vec{\rho}_1,\vec{\sigma},\sigma\Rightarrow\tau \quad \overset{t}{\triangledown}\ \vec{\rho}_1,\vec{\sigma},\sigma,N,\tau\Rightarrow\tau}{\vec{\rho}_1,\vec{\sigma},\sigma,N\Rightarrow\tau}}{\vec{\rho}_0,\vec{\rho}_1,\vec{\sigma},N\Rightarrow\tau}$$

is transformed to,

$$\mathsf{cntr}\dfrac{\mathsf{rec}_\tau\dfrac{\mathsf{cut}'\dfrac{\overset{r}{\triangledown}\ \vec{x}_0,\vec{\sigma},N\Rightarrow\sigma \quad \overset{s}{\triangledown}\ \vec{\rho}_1,\vec{\sigma},\sigma\Rightarrow\tau}{\vec{\rho}_0,\vec{\rho}_1,\vec{\sigma},N\Rightarrow\tau} \qquad \mathsf{cut}'\dfrac{\overset{r}{\triangledown}\ \vec{x}_0,\vec{\sigma},N\Rightarrow\sigma \quad \overset{t}{\triangledown}\ \vec{\rho}_1,\vec{\sigma},\sigma,N,\tau\Rightarrow\tau}{\vec{\rho}_0,\vec{\rho}_1,\vec{\sigma},N,N,\tau\Rightarrow\tau}}{\vec{\rho}_0,\vec{\rho}_1,\vec{\sigma},N,\underline{N}\Rightarrow\tau}}{\vec{\rho}_0,\vec{\rho}_1,\vec{\sigma},\underline{N}\Rightarrow\tau}$$

where we have underlined principal types and used colours to identify type occurrences according to ancestry.

We verify that T_n proves the equality of the derivations before and after transformation as follows. First we show that,

$$\mathsf{rec}\,(\mathsf{cut}'r\,s)\,(\mathsf{cut}'r\,t)\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,a\,z \;=\; \mathsf{rec}\,s\,t\,\vec{x}_1\,\vec{y}\,(r\,\vec{x}_0\,\vec{y}\,a)\,z \tag{33}$$

by (object-level) induction on z. For the base case:

$$\begin{array}{lll}
& \mathsf{rec}\,(\mathsf{cut}'r\,s)\,(\mathsf{cut}'r\,t)\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,a\,0 & \\
= & \mathsf{cut}'r\,s\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,a & \text{by } \mathsf{rec} \text{ axioms} \\
= & s\,\vec{x}_1\,\vec{y}\,(r\,\vec{x}_0\,\vec{y}\,a) & \text{by } \mathsf{cut}' \text{ axiom} \\
= & \mathsf{rec}\,s\,t\,\vec{x}_1\,\vec{y}\,(r\,\vec{x}_0\,\vec{y}\,a)\,0 & \text{by } \mathsf{rec} \text{ axioms}
\end{array}$$

For the inductive step:

$$\begin{array}{lll}
& \mathsf{rec}\,(\mathsf{cut}'r\,s)\,(\mathsf{cut}'r\,t)\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,a\,\mathsf{s}z & \\
= & \mathsf{cut}'r\,t\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,a\,z\,(\mathsf{rec}\,(\mathsf{cut}'r\,s)\,(\mathsf{cut}'r\,t)\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,a\,z) & \text{by } \mathsf{rec} \text{ axioms} \\
= & \mathsf{cut}'r\,t\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,a\,z\,(\mathsf{rec}\,s\,t\,\vec{x}_1\,\vec{y}\,(r\,\vec{x}_0\,\vec{y}\,a)\,z) & \text{by inductive hypothesis, (33)} \\
= & t\,\vec{x}_1\,\vec{y}\,(r\,\vec{x}_0\,\vec{y}\,a)\,z\,(\mathsf{rec}\,s\,t\,\vec{x}_1\,\vec{y}\,(r\,\vec{x}_0\,\vec{y}\,a)\,z) & \text{by } \mathsf{cut}' \text{ axiom} \\
= & \mathsf{rec}\,s\,t\,\vec{x}_1\,\vec{y}\,(r\,\vec{x}_0\,\vec{y}\,a)\,\mathsf{s}z & \text{by } \mathsf{rec} \text{ axioms}
\end{array}$$

From here we conclude the verification in T_n by the cntr and cut axioms.

A.5. cut'-cut **commutative cases.** Sometimes we have to commute a $\mathsf{cut}'_{>n}$ over a $\mathsf{cut}_{\leq n}$. It is notationally cumbersome to consider all splitting possibilities for commuting cut' over cut', so instead we just permute a $\mathsf{cut}'_{>n}$ step over a $\mathsf{cut}_{\leq n}$ step. From here, commutation over a $\mathsf{cut}'_{\leq n}$ step follows by further commutation over wk-steps (see next subsection).

$$\dfrac{\dfrac{\overset{r}{\triangledown} \quad \overset{s}{\triangledown}}{\vec{\rho}_0, \vec{\sigma} \Rightarrow \rho \quad \vec{\rho}_0, \vec{\sigma}, \rho \Rightarrow \sigma}\,\mathsf{cut}_{\leq n} \quad \overset{t}{\triangledown}}{\vec{\rho}_0, \vec{\sigma} \Rightarrow \sigma \qquad \vec{\rho}_1, \vec{\sigma}, \sigma \Rightarrow \tau}\,\mathsf{cut}'_{>n}$$
$$\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma} \Rightarrow \tau$$

is transformed to:

$$\dfrac{\overset{r}{\triangledown} \qquad \dfrac{\overset{s}{\triangledown} \quad \overset{t}{\triangledown}}{\vec{\rho}_0, \vec{\sigma}, \rho \Rightarrow \sigma \quad \vec{\rho}_1, \vec{\sigma}, \sigma \Rightarrow \tau}\,\mathsf{cut}'_{>n}}{\vec{\rho}_0, \vec{\sigma} \Rightarrow \rho \qquad \vec{\rho}_0, \vec{\rho}_1, \vec{\sigma}, \rho \Rightarrow \tau}\,\mathsf{cut}'_{\leq n}$$
$$\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma} \Rightarrow \tau$$

We verify that T_n proves the equality of the derivations before and after transformation as follows:

$$\begin{array}{lll}
& \mathsf{cut}' r\,(\mathsf{cut}' s\,t)\,\vec{x}_0\,\vec{x}_1\,\vec{y} & \\
= & \mathsf{cut}' s\,t\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,(r\,\vec{x}_0\,\vec{y}) & \text{by } \mathsf{cut}' \text{ axiom} \\
= & t\,\vec{x}_1\,\vec{y}\,(s\,\vec{x}_0\,\vec{y}\,(r\,\vec{x}_0\,\vec{y})) & \text{by } \mathsf{cut}' \text{ axiom} \\
= & t\,\vec{x}_1\,\vec{y}\,(\mathsf{cut}\,r\,s\,\vec{x}_0, \vec{y}) & \text{by } \mathsf{cut} \text{ axiom} \\
= & \mathsf{cut}'(\mathsf{cut}\,r\,s)\,t\,\vec{x}_0\,\vec{x}_1\,\vec{y} & \text{by } \mathsf{cut}' \text{ axiom}
\end{array}$$

There is also a similar cut-commutation the other way around, when the right premiss of a cut' step ends with cut. There is no commutation of cut' above the left side of a rec step, since that would immediately imply that the level of the cut is bounded by that of a recursor.

A.6. cut'-wk **key and commutative case.** There are two possible interactions between cut' and wk, depending on whether the cut-formula is weakened or not. Both are relatively simple.

$$\dfrac{\overset{s}{\triangledown} \qquad \dfrac{\overset{t}{\triangledown}}{\vec{\rho}_1, \vec{\sigma} \Rightarrow \tau}\,\mathsf{wk}}{\vec{\rho}_0, \vec{\sigma} \Rightarrow \sigma \qquad \vec{\rho}_1, \vec{\sigma}, \sigma \Rightarrow \tau}\,\mathsf{cut}'$$
$$\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma} \Rightarrow \tau$$

is transformed to:

$$\dfrac{\overset{t}{\triangledown}}{\vec{\rho}_1, \vec{\sigma} \Rightarrow \tau}\,\mathsf{wk}$$
$$\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma} \Rightarrow \tau$$

We verify that T_n proves the equality of the derivations before and after transformation as follows:

$$\begin{array}{lll} & \mathsf{wk}^* t\,\vec{x}_0\,\vec{x}_1\,\vec{y} & \\ = & t\,\vec{x}_1\,\vec{y} & \text{by } \mathsf{wk} \text{ axioms} \\ = & \mathsf{wk}\,t\,\vec{x}_1, \vec{y}\,(s\,\vec{x}_0\,\vec{y}) & \text{by } \mathsf{wk} \text{ axiom} \\ = & \mathsf{cut}'s\,(\mathsf{wk}\,t)\,\vec{x}_0\,\vec{x}_1, \vec{y} & \text{by } \mathsf{cut}' \text{ axiom} \end{array}$$

$$\mathsf{cut}'\dfrac{s\quad \mathsf{wk}\dfrac{t}{\vec{\rho}_1, \vec{\sigma}, \sigma \Rightarrow \tau}}{\vec{\rho}_0, \rho, \vec{\sigma} \Rightarrow \sigma \qquad \rho, \vec{\rho}_1, \vec{\sigma}, \sigma \Rightarrow \tau}$$
$$\vec{\rho}_0, \rho, \vec{\rho}_1, \vec{\sigma} \Rightarrow \tau$$

is transformed to either the left or right derivations below, depending on whether the purple ρ is present or not, respectively:

$$\mathsf{cut}'\dfrac{s \qquad t}{\vec{\rho}_0, \rho, \vec{\sigma} \Rightarrow \sigma \qquad \vec{\rho}_1, \vec{\sigma}, \sigma \Rightarrow \tau}$$
$$\vec{\rho}_0, \rho, \vec{\rho}_1, \vec{\sigma} \Rightarrow \tau$$

$$\mathsf{cut}'\dfrac{s \qquad t}{\vec{\rho}_0, \vec{\sigma} \Rightarrow \sigma \qquad \vec{\rho}_1, \vec{\sigma}, \sigma \Rightarrow \tau}$$
$$\mathsf{wk}\dfrac{\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma} \Rightarrow \tau}{\vec{\rho}_0, \rho, \vec{\rho}_1, \vec{\sigma} \Rightarrow \tau}$$

We verify that T_n proves the equality of the derivations before and after transformation, respectively, as follows:

$$\begin{array}{lll} & \mathsf{cut}'s\,t\,\vec{x}_0\,x\,\vec{x}_1\,\vec{y} & \\ = & t\,\vec{x}_1\,\vec{y}\,(s\,\vec{x}_0\,x\,\vec{y}) & \text{by } \mathsf{cut}' \text{ axiom} \\ = & \mathsf{wk}\,t\,x\,\vec{x}_1\,\vec{y}\,(s\,\vec{x}_0\,x\,\vec{y}) & \text{by } \mathsf{wk} \text{ axiom} \\ = & \mathsf{cut}'s\,(\mathsf{wk}\,t)\,\vec{x}_0\,x\,\vec{x}_1\,\vec{y} & \text{by } \mathsf{cut}' \text{ axiom} \end{array}$$

$$\begin{array}{lll} & \mathsf{wk}\,(\mathsf{cut}'\,s\,t)\,\vec{x}_0\,x\,\vec{x}_1\,\vec{y} & \\ = & \mathsf{cut}'\,s\,t\,\vec{x}_0\,\vec{x}_1\,\vec{y} & \text{by } \mathsf{wk} \text{ axiom} \\ = & t\,\vec{x}_1\,\vec{y}\,(s\,\vec{x}_0\,\vec{y}) & \text{by } \mathsf{cut}' \text{ axiom} \\ = & \mathsf{wk}\,t\,x\,\vec{x}_1\,\vec{y}\,(s\,\vec{x}_0\,\vec{y}) & \text{by } \mathsf{wk} \text{ axiom} \\ = & \mathsf{cut}'s\,(\mathsf{wk}\,t)\,\vec{x}_0\,x\,\vec{x}_1\,\vec{y} & \text{by } \mathsf{cut}' \text{ axiom} \end{array}$$

There are similar cases when the left premiss of a cut' concludes a wk step.

A.7. cut'-L' **commutative cases.** We have to treat the cases when a L' step is on the left or on the right separately.

$$\mathsf{cut}'\dfrac{\mathsf{L}'\dfrac{r \qquad s}{\vec{\rho}_0, \vec{\sigma}, \rho \to \sigma \Rightarrow \rho \qquad \vec{\rho}_0, \sigma, \rho \to \sigma, \sigma \Rightarrow \pi} \qquad t}{\vec{\rho}_0, \vec{\sigma}, \rho \to \sigma \Rightarrow \pi \qquad \vec{\rho}_1, \vec{\sigma}, \rho \to \sigma, \pi, \Rightarrow \tau}$$
$$\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma}, \rho \to \sigma \Rightarrow \tau$$

is transformed to,

$$\mathsf{L}'\dfrac{\mathsf{wk}\dfrac{r}{\vec{\rho}_0, \vec{\sigma}, \rho \to \sigma \Rightarrow \rho} \qquad \mathsf{cut}'\dfrac{s \qquad t}{\vec{\rho}_0, \sigma, \rho \to \sigma, \sigma \Rightarrow \pi \qquad \vec{\rho}_1, \vec{\sigma}, \rho \to \sigma, \pi \Rightarrow \tau}}{\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma}, \rho \to \sigma, \sigma \Rightarrow \tau}$$
$$\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma}, \rho \to \sigma \Rightarrow \tau$$

where the purple occurrence of $\rho \to \sigma$ after transformation is present only if it is present before transformation.

We verify that T_n proves the equality of the derivations before and after transformation as follows,

$$\begin{array}{lll} & \mathsf{L}'(\mathsf{wk}^* r)\,(\mathsf{cut}' s\,t)\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,z & \\ = & \mathsf{cut}' s\,t\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,z\,(z\,(\mathsf{wk}^* r\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,z)) & \text{by } \mathsf{L}' \text{ axiom} \\ = & \mathsf{cut}' s\,t\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,z\,(z\,(r\,\vec{x}_0\,\vec{y}\,z)) & \text{by } \mathsf{wk} \text{ axioms} \\ = & t\,\vec{x}_1\,\vec{y}\,z\,(s\,\vec{x}_0\,\vec{y}\,z\,(z\,(r\,\vec{x}_0\,\vec{y}\,z))) & \text{by } \mathsf{cut}' \text{ axiom} \\ = & t\,\vec{x}_1\,\vec{y}\,z\,(\mathsf{L}' r\,s\,\vec{x}_0\,\vec{y}\,z) & \text{by } \mathsf{L}' \text{ axiom} \\ = & \mathsf{cut}'(\mathsf{L}' r\,s)\,t\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,z & \end{array}$$

where, again, the purple z is present just if the purple $\rho \to \sigma$ are present in the derivations.

$$\mathsf{cut}'\dfrac{\overset{r}{\triangledown}\ \vec{\rho}_0, \vec{\sigma}, \rho \to \sigma \Rightarrow \pi \qquad \mathsf{L}'\dfrac{\overset{s}{\triangledown}\ \vec{\rho}_1, \vec{\sigma}, \rho \to \sigma, \pi \Rightarrow \rho \qquad \overset{t}{\triangledown}\ \vec{\rho}_1, \vec{\sigma}, \rho \to \sigma, \sigma, \pi \Rightarrow \tau}{\vec{\rho}_1, \vec{\sigma}, \rho \to \sigma, \pi \Rightarrow \tau}}{\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma}, \rho \to \sigma \Rightarrow \tau}$$

is transformed to,

$$\mathsf{L}'\dfrac{\mathsf{cut}'\dfrac{\overset{r}{\triangledown}\ \vec{\rho}_0, \vec{\sigma}, \rho \to \sigma \Rightarrow \pi \qquad \overset{s}{\triangledown}\ \vec{\rho}_1, \vec{\sigma}, \rho \to \sigma, \pi \Rightarrow \rho}{\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma}, \rho \to \sigma \Rightarrow \rho} \qquad \mathsf{cut}'\dfrac{\overset{r}{\triangledown}\ \vec{\rho}_0, \vec{\sigma}, \rho \to \sigma \Rightarrow \pi \qquad \overset{t}{\triangledown}\ \vec{\rho}_1, \vec{\sigma}, \rho \to \sigma, \sigma, \pi \Rightarrow \tau}{\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma}, \rho \to \sigma, \sigma \Rightarrow \tau}}{\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma}, \rho \to \sigma \Rightarrow \tau}$$

where the purple occurrences of $\rho \to \sigma$ after transformation are present just if they are before transformation.

We verify that T_n proves the equality of the derivations before and after transformation as follows,

$$\begin{array}{lll} & \mathsf{L}'(\mathsf{cut}' r\,s)\,(\mathsf{cut}' r\,t)\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,z & \\ = & \mathsf{cut}' r\,t\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,z\,(z\,(\mathsf{cut}' r\,s\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,z)) & \text{by } \mathsf{L}' \text{ axiom} \\ = & \mathsf{cut}' r\,t\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,z\,(z\,(s\,\vec{x}_1\,\vec{y}\,z\,(r\,\vec{x}_0\,\vec{y}\,z))) & \text{by } \mathsf{cut}' \text{ axiom} \\ = & t\,\vec{x}_1\,\vec{y}\,z\,(z\,(s\,\vec{x}_1\,\vec{y}\,z\,(r\,\vec{x}_0\,\vec{y}\,z)))\,(r\,\vec{x}_0\,\vec{y}\,z) & \text{by } \mathsf{cut}' \text{ axiom} \\ = & \mathsf{L}' s\,t\,\vec{x}_1\,\vec{y}\,z\,(r\,\vec{x}_0\,\vec{y}\,z) & \text{by } \mathsf{L}' \text{ axiom} \\ = & \mathsf{cut}' r\,(\mathsf{L}' s\,t)\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,z & \text{by } \mathsf{cut}' \text{ axiom} \end{array}$$

where the purple z occurrences are present only if the purple $\rho \to \sigma$ occurrences are present in the derivations.

A.8. cut'-R commutative case.

$$\mathsf{cut}'\dfrac{\overset{s}{\triangledown}\ \vec{\rho}_0, \vec{\sigma} \Rightarrow \sigma \qquad \mathsf{R}\dfrac{\overset{t}{\triangledown}\ \vec{\rho}_1, \vec{\sigma}, \sigma, \rho \Rightarrow \tau}{\vec{\rho}_1, \vec{\sigma}, \sigma \Rightarrow \rho \to \tau}}{\vec{\rho}_0, \vec{\rho}_1, \vec{\sigma} \Rightarrow \rho \to \tau}$$

is transformed to:

$$\mathsf{R}\,\dfrac{\mathsf{cut}'\,\dfrac{\overset{s}{\triangledown}\quad\overset{t}{\triangledown}}{\vec{\rho}_0,\vec{\sigma}\Rightarrow\sigma \qquad \vec{\rho}_1,\vec{\sigma},\sigma,\rho\Rightarrow\tau}}{\vec{\rho}_0,\vec{\rho}_1,\vec{\sigma},\rho\Rightarrow\tau}}{\vec{\rho}_0,\vec{\rho}_1,\vec{\sigma}\Rightarrow\rho\to\tau}$$

We verify that T_n proves the equality of the derivations before and after transformation as follows:

$$\begin{array}{lll}
 & \mathsf{R}\,(\mathsf{cut}'s\,t)\vec{x}_0\,\vec{x}_1\,\vec{y}\,z & \\
= & \mathsf{cut}'s\,t\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,z & \text{by } \mathsf{R} \text{ axiom} \\
= & t\,\vec{x}_1\,\vec{y}\,(s\,\vec{x}_0\,\vec{y})\,z & \text{by } \mathsf{cut}' \text{ axiom} \\
= & \mathsf{R}\,t\,\vec{x}_1\,\vec{y}\,(s\,\vec{x}_0\,\vec{y})\,z & \text{by } \mathsf{R} \text{ axiom} \\
= & \mathsf{cut}'\,s\,(\mathsf{R}\,t)\,\vec{x}_0\,\vec{x}_1\,\vec{y}\,z & \text{by } \mathsf{cut}' \text{ axiom}
\end{array}$$