Non-well-founded Proof Theory of Transitive Closure Logic

LIRON COHEN, Ben-Gurion University, Israel
REUBEN N. S. ROWE, Royal Holloway, University of London, UK

Supporting inductive reasoning is an essential component is any framework of use in computer science. To do so, the logical framework must extend that of first-order logic. Transitive closure logic is a known extension of first-order logic which is particularly straightforward to automate. While other extensions of first-order logic with inductive definitions are a priori parametrized by a set of inductive definitions, the addition of a single transitive closure operator has the advantage of uniformly capturing all finitary inductive definitions. To further improve the reasoning techniques for transitive closure logic we here present an *infinitary* proof system for it which is an *infinite descent*-style counterpart to the existing (explicit induction) proof system for the logic. We show that the infinitary system is complete for the standard semantics and subsumes the explicit system. Moreover, the uniformity of the transitive closure operator allows semantically meaningful complete restrictions to be defined using simple syntactic criteria. Consequently, the restriction to regular infinitary (i.e. *cyclic*) proofs provides the basis for an effective system for automating inductive reasoning.

CCS Concepts: • **Theory of computation → Logic and verification**; **Proof theory**; **Automated reasoning**.

Additional Key Words and Phrases: Induction, Transitive Closure, Infinitary Proof Systems, Cyclic Proof Systems, Soundness, Completeness, Standard Semantics, Henkin Semantics

ACM Reference Format:
Liron Cohen and Reuben N. S. Rowe. 2020. Non-well-founded Proof Theory of Transitive Closure Logic. 1, 1 (October 2020), 30 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Induction is a core reasoning technique, especially in computer science, where it plays a central role in reasoning about recursive data and computations. There is therefore a constant attempt to increase and improve the armoury of techniques available for automated inductive reasoning. This battle is waged along two intertwined fronts: firstly finding the right logical framework, and secondly developing effective associated proof methods. In other words, we are concerned with both being able to formalize as much mathematical reasoning as possible, and also with doing so in an effective way.

In terms of the logical framework, one should aim for a logic powerful enough to capture finitary[1] inductive principles, yet as simple as possible in order to facilitate automation. *Transitive closure* (TC) logic has been identified as a minimal, general purpose logic for inductive reasoning that is also very suitable for automation [2, 15, 16]. TC adds to first-order logic a single operator for forming

[1]We here mean finitary as opposed to transfinitary.

Authors' addresses: Liron Cohen, Dept. of Computer Science, Ben-Gurion University, Be'er Sheva, Israel, cliron@cs.bgu.ac.il; Reuben N. S. Rowe, Dept of Computer Science, Royal Holloway, University of London, Egham, Surrey, UK, reuben.rowe@rhul.ac.uk.

XXXX-XXXX/2020/10-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

binary relations: specifically, the transitive closures of arbitrary formulas (or, more precisely, the transitive closure of the binary relation induced by a formula with respect to two distinct variables). It is thus able to express, e.g., unbounded reachability; on the other hand, it cannot express, e.g., well-foundedness of relations. Thus TC logic is intermediate between first- and second-order logic.

Despite its minimality TC logic retains enough expressivity to capture inductive reasoning, as well as to subsume arithmetics (see Section 6.2.1). Moreover, it provides a *uniform* way of capturing inductive principles. If an induction scheme is expressed by a formula φ, then the elements of the inductive collection it defines are those 'reachable' from the base elements x via the iteration of the induction scheme. That is, those y's for which (x, y) is in the transitive closure of φ. Accordingly, while other extensions of first-order logic with inductive definitions are a priori parametrized by a set of inductive definitions (see, e.g., [11, 27, 29, 38]), bespoke induction principles do not need to be added to, or embedded within, transitive closure logic; instead, all induction schemes are available within a single, unified language. In this respect, the transitive closure operator resembles the W-type [28], which also provides a single type constructor from which one can uniformly define a variety of inductive types. This conciseness of the logic makes it of particular interest from an automation point of view. The use of only one constructor of course comes with a price: namely, formalizations (mostly of non-linear induction schemes) may be somewhat complex. However, they generally do not require as complex an encoding as in arithmetics, since the TC operator can be applied on any formula and thus (depending on the underlying signature) more naturally encode induction on sets more complex than the natural numbers.

Since its expressiveness entails that TC logic subsumes arithmetics, by Gödel's result, any effective proof system for it must necessarily be incomplete for the standard semantics. This poses a major challenge for the second of our stated objectives in the programme of developing automated inductive reasoning, i.e. finding effective proof machinery for our chosen logic. Notwithstanding, a natural, effective proof system which is sound for TC logic was shown to be complete with respect to a generalized form of Henkin semantics [14]. That system, in keeping with standard practice, captures the notion of inductive reasoning via an explicit inference rule that expresses the general induction principle of the operator.

Aiming to enhance the opportunities for automating formal reasoning in TC logic, this paper presents an *infinitary* proof theory for TC logic in the same vein as similar recent developments for other logics with fixed point constructions [1, 3, 9–11, 21, 22, 33, 35]). This, as far as we know, is the first system for TC logic that is (cut-free) complete with respect to its standard semantics. More specifically we employ recent techniques from non-well-founded proof theory, which embodies the philosophy of infinite descent, as an alternative to explicit induction. Such systems incorporate infinite-height, rather than infinite-width proofs (see Section 4). The soundness of such infinitary proof theories is underpinned by the principle of *infinite descent*: proofs are permitted to be infinite, non-well-founded trees, but subject to the restriction that every infinite path in the proof admits some infinite descent. The descent is witnessed by tracing terms or formulas for which we can give a correspondence with elements of a well-founded set. In particular, we can trace terms that denote elements of an inductively defined (well-founded) set. For this reason, such theories are considered systems of implicit induction, as opposed to those which employ explicit rules for applying induction principles. While a full infinitary proof theory is clearly not effective, in the aforementioned sense, such a system can be obtained by restricting consideration to only the *regular* infinite proofs. These are precisely those proofs that can be finitely represented as (possibly cyclic) graphs.

These infinitary proof theories generally subsume systems of explicit induction in expressive power, but also offer a number of advantages. Most notably, they can ameliorate the primary challenge for inductive reasoning: finding an induction *invariant*. In explicit induction systems,

this must be provided *a priori*, and is often much stronger than the goal one is ultimately interested in proving. However, in implicit systems the inductive arguments and hypotheses may be encoded in the cycles of a proof, so cyclic proof systems seem better for automation. The cyclic approach has also been used to provide an optimal cut-free complete proof system for Kleene algebra [20], providing further evidence of its utility for automation.

In the setting of TC logic, we observe some further benefits over more traditional formal systems of inductive definitions and their infinitary proof theories (cf. LKID [11, 27]). TC (with a pairing function) has all first-order definable finitary inductive definitions immediately 'available' within the language of the logic: as with inductive hypotheses, one does not need to 'know' in advance which induction schemes will be required. Moreover, the use of a single transitive closure operator provides a uniform treatment of all induction schemes. That is, instead of having a proof system parameterized by a set of inductive predicates and rules for them (as is the case in LKID), TC offers a single proof system with a single rule scheme for induction. This has immediate advantages for developing the metatheory: the proofs of completeness for standard semantics and adequacy (i.e. subsumption of explicit induction) for the infinitary system presented in this paper are simpler and more straightforward. Moreover, it permits a further refinement of the cyclic system, which also subsumes explicit induction, to be defined via a simple structural criterion that we call *normality*. This restriction in the search space of possible proofs further enhances the potential for automation. TC logic seems more expressive in other ways, too. For instance, the transitive closure operator may be applied to arbitrarily complex formulas, and thus inductive definitions are not restricted to consist only of conjunctions of atomic formulas (i.e. Horn clauses) as in e.g. [9, 11]. Conversely, since the TC operator is a particular instance of a least fixed point operator, it is itself subsumed by fixed-point logics such as the μ-calculus [25].

We show that the explicit and cyclic TC systems are equivalent under arithmetic, as is the case for LKID [7, 34]. However, there are cases in which the cyclic system for LKID is strictly more expressive than the explicit induction system [6]. To obtain a similar result for TC, the fact that all induction schemes are available poses a serious challenge. For one, the counter-example used in [6] does not serve to show that this result holds for TC. If this strong inequivalence indeed holds also for TC, it must be witnessed by a more subtle and complex counter-example. Conversely, it may be that the explicit and cyclic systems do coincide for TC. In either case, this points towards some interesting subtleties of these LKID results in the TC setting, stemming from lifting the restriction of having the system parameterized by a fixed set of inductive definitions.

The rest of the paper is organised as follows. In Section 2 we reprise the definition of transitive closure logic and both its standard and Henkin-style semantics. Sections 3 and 4 present, respectively, the existing explicit induction proof system and our new infinitary and cyclic proof systems for TC logic, along with their soundness, completeness and cut-admissibility results. In Section 5 we consider how our treatment extends to two important variants of transitive closure logic: one with a pairing function and one without explicit equality. To complete the picture, Section 6 then compares the expressive power of the infinitary system (and its cyclic subsystem) with the explicit system. Finally, Section 7 concludes and examines the remaining open questions for our system as well as future work. This paper is an extended version of [17].

2 TRANSITIVE CLOSURE LOGIC AND ITS EXPRESSIVNESS

This section reviews the language of transitive closure logic, and defines both its standard and Henkin-style semantics. We also illustrate the usefulness of the logic in various applications in computer science.

For simplicity of presentation we assume (as is standard practice) a designated equality symbol in the language. We denote by $v[x_1 := a_n, \ldots, x_n := a_n]$ the variant of the assignment v which

assigns a_i to x_i for each i, and by $\varphi\left\{\frac{t_1}{x_1},\ldots,\frac{t_n}{x_n}\right\}$ the result of simultaneously substituting each t_i for the free occurrences of x_i in φ. Note also that we use an operator denoting the reflexive transitive closure; however the reflexive and non-reflexive forms are equivalent in the presence of equality.

2.1 The Syntax and Semantics

Definition 2.1 (The language $\mathcal{L}_{\mathsf{RTC}}$). Let σ be a first-order signature with equality, whose terms are ranged over by s and t and predicates by P, and let x, y, z, etc. range over a countable set $\mathcal{V}$ of variables. The language $\mathcal{L}_{\mathsf{RTC}}$ consists of the formulas defined by the grammar:

$$\varphi,\psi ::= s = t \mid P(t_1,\ldots,t_n) \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \forall x.\varphi \mid \exists x.\varphi \mid (RTC_{x,y}\,\varphi)(s,t)$$

As usual, $\forall x$ and $\exists x$ bind free occurrences of the variable x and we identify formulas up to renaming of bound variables, so that capturing of free variables during substitution does not occur. Note that in the formula $(RTC_{x,y}\,\varphi)(s,t)$ free occurrences of x and y in φ are also bound (but not those in s and t).

Definition 2.2 (Standard Semantics). Let $M = \langle D, I\rangle$ be a first-order structure (i.e. D is a non-empty domain and I an interpretation function), and v an assignment in M which we extend to terms in the obvious way. The satisfaction relation $\models$ between model-valuation pairs $\langle M, v\rangle$ and formulas is defined inductively on the structure of formulas by:

- $M, v \models s = t$ if $v(s) = v(t)$;
- $M, v \models P(t_1,\ldots,t_n)$ if $(v(t_1),\ldots,v(t_n)) \in I(P)$;
- $M, v \models \neg\varphi$ if $M, v \not\models \varphi$;
- $M, v \models \varphi_1 \wedge \varphi_2$ if both $M, v \models \varphi_1$ and $M, v \models \varphi_2$;
- $M, v \models \varphi_1 \vee \varphi_2$ if either $M, v \models \varphi_1$ or $M, v \models \varphi_2$;
- $M, v \models \varphi_1 \rightarrow \varphi_2$ if $M, v \models \varphi_1$ implies $M, v \models \varphi_2$;
- $M, v \models \exists x.\varphi$ if $M, v[x := a] \models \varphi$ for some $a \in D$;
- $M, v \models \forall x.\varphi$ if $M, v[x := a] \models \varphi$ for all $a \in D$;
- $M, v \models (RTC_{x,y}\,\varphi)(s,t)$ if $v(s) = v(t)$, or there exist $a_0,\ldots,a_n \in D$ $(n > 0)$ s.t. $v(s) = a_0$, $v(t) = a_n$, and $M, v[x := a_i, y := a_{i+1}] \models \varphi$ for $0 \leq i < n$.

We say that a formula φ is valid with respect to the standard semantics when $M, v \models \varphi$ holds for all models M and valuations v.

We next recall the concepts of frames and Henkin structures (see, e.g., [24]). A frame is a first-order structure together with some subset of the powerset of its domain (called its set of admissible subsets).

Definition 2.3 (Frames). A frame M is a triple $\langle D, I, \mathcal{D}\rangle$, where $\langle D, I\rangle$ is a first-order structure, and $\mathcal{D} \subseteq \wp(D)$.

Definition 2.4 (Frame Semantics). $\mathcal{L}_{\mathsf{RTC}}$ formulas are interpreted in frames as in Definition 2.2 above, except for:

- $M, v \models (RTC_{x,y}\,\varphi)(s,t)$ if for every $A \in \mathcal{D}$, if $v(s) \in A$ and for every $a, b \in D$: $a \in A$ and $M, v[x := a, y := b] \models \varphi$ implies $b \in A$, then $v(t) \in A$.

Note that if $\mathcal{D} = \wp(D)$, the frame is identified with a standard first-order structure.

We now consider Henkin structures, which are frames whose set of admissible subsets is closed under parametric definability.

Definition 2.5 (Henkin structures). A Henkin structure $M = \langle D, I, \mathcal{D}\rangle$ is a frame such that for every formula φ and valuation v in M we have $\{a \in D \mid M, v[x := a] \models \varphi\} \in \mathcal{D}$.

We refer to the semantics induced by quantifying over the (larger) class of Henkin structures as the Henkin semantics.

2.2 Applications of Transitive Closure Logic

Transitive Closure Logic offers a variety of applications in different areas in computer science, such as program verification, database query languages, descriptive complexity, etc. We briefly describe some of these here.

Inductive Numerical Predicates. Transitive Closure logic enables complex numerical induction schemes to be expressed concisely and naturally. This supports the application of TC in a multitude of other areas, since numerical theories often form the foundation of more complex formalisms.

Assuming a signature containing a constant $\mathbf{0}$ for zero, and a successor function $\mathbf{s}$, a predicate N characterising natural numbers can easily be defined by $\mathsf{N}(t) \equiv (RTC_{u,v}\, v = \mathbf{s}(u))(\mathbf{0}, t)$. Now consider a binary predicate H defined by the following induction scheme over natural numbers x and y considered in [6]:

i) $\mathsf{H}(\mathbf{0}, \mathbf{0})$, $\mathsf{H}(\mathbf{s}(\mathbf{0}), \mathbf{0})$, and $\mathsf{H}(x, \mathbf{s}(\mathbf{0}))$ hold;
ii) if $\mathsf{H}(x, y)$ holds then so does $\mathsf{H}(\mathbf{s}(x), \mathbf{s}(\mathbf{s}(y)))$;
iii) if $\mathsf{H}(\mathbf{s}(x), x)$ holds then so do $\mathsf{H}(\mathbf{0}, \mathbf{s}(\mathbf{s}(x)))$ and $\mathsf{H}(\mathbf{s}(\mathbf{s}(x)), \mathbf{0})$.

This predicate is a binary version of the Kirby-Paris Hydra game [26] that considers a Hydra with two heads. It can be expressed, using a pairing function $\langle \cdot, \cdot \rangle$, by the TC formula $\mathsf{H}(t_1, t_2)$ defined by:

$$\mathsf{H}(t_1, t_2) \equiv \exists n_1, n_2 \,.\, (RTC_{x,y}\, \varphi_1 \vee \varphi_2 \vee \varphi_3)(\langle n_1, n_2 \rangle, \langle t_1, t_2 \rangle) \\ \wedge ((n_1 = \mathbf{0} \wedge n_2 = \mathbf{0}) \vee (n_1 = \mathbf{s}(\mathbf{0}) \wedge n_2 = \mathbf{0}) \vee (\mathsf{N}(n_1) \wedge n_2 = \mathbf{s}(\mathbf{0})))$$

where the formulas φ_1–φ_3 are defined as follows:

$$\varphi_1 \equiv \exists z_1 \,.\, \exists z_2 \,.\, x = \langle z_1, z_2 \rangle \wedge y = \langle \mathbf{s}(z_1), \mathbf{s}(\mathbf{s}(z_2)) \rangle$$
$$\varphi_2 \equiv \exists z \,.\, x = \langle z, \mathbf{s}(z) \rangle \wedge y = \langle \mathbf{0}, \mathbf{s}(\mathbf{s}(z)) \rangle$$
$$\varphi_3 \equiv \exists z \,.\, x = \langle z, \mathbf{s}(z) \rangle \wedge y = \langle \mathbf{s}(\mathbf{s}(z)), \mathbf{0} \rangle$$

and express the inductive steps of the scheme, cf. items (ii) and (iii). $\mathsf{H}(t_1, t_2)$ asserts that the pair of terms (t_1, t_2) may be reached via (some arbitrary number of) applications of these steps from one of the base cases, cf. item (i). It is easy to show that any such pair of terms must consist of natural numbers (i.e. $\mathsf{N}(t_1)$ and $\mathsf{N}(t_2)$ hold). Moreover, the H predicate is total over the natural numbers, a fact that is derivable in each of the proof systems we present for TC.

Temporal Logic. It is possible to encode temporal logics in TC, since temporal operators such as 'eventually', 'globally' (in the past, or future) and 'until' essentially denote reachability properties between temporal states. For example, consider LTL (linear temporal logic) [31] over some signature. To encode it in TC we essentially make the state-based semantics explicit by:

- extending the arity of each predicate symbol to capture how its interpretation changes over time (e.g. $\mathbf{p}(t)$ becomes $\mathbf{p}(t, s)$, true if and only if $\mathbf{p}(t)$ is true in the state denoted by s);
- introducing a fresh unary function $\mathbf{next}(s)$ to denote the state immediately following s; and
- assuming a term constant $\mathbf{s_{init}}$ to indicate the initial state.

The formula $(RTC_{x,y}\, y = \mathbf{next}(x))(s, s')$ then expresses that state s' occurs *after* state s, which we will abbreviate as $s \leq s'$. We can then define a translation $T[s]$ of LTL formulas, with respect to a 'state' variable s. For atomic formulas, we define $T[s](\mathbf{q}(t_1, \ldots, t_n)) = \mathbf{q}(t_1, \ldots, t_n, s)$. For standard first-order logical connectives, the translation is defined straightforwardly by induction.

The translation of the temporal operators **X** (next), **F** (finally, or eventually), **G** (globally, or always), and **U** (until), are as follows:

$$\begin{aligned} T[s](\mathbf{X}\,\phi) &= \exists s' \,.\, s' = \mathbf{next}(s) \wedge T[s'](\phi) \\ T[s](\mathbf{F}\,\phi) &= \exists s' \,.\, s \leq s' \wedge T[s'](\phi) \\ T[s](\mathbf{G}\,\phi) &= \forall s' \,.\, s \leq s' \rightarrow T[s'](\phi) \\ T[s](\phi\ \mathbf{U}\ \psi) &= \exists s' \,.\, s \leq s' \wedge T[s'](\psi) \wedge \forall s'' \,.\, (s \leq s'' \wedge s'' \leq s') \rightarrow T[s''](\phi) \end{aligned}$$

Finally, an LTL formula ψ is interpreted as $T[\mathbf{s_{init}}](\psi)$. This is essentially equivalent to the standard translation of modal logic into first-order logic given in, e.g., [4, 8] which takes the temporal ordering relation as a primitive. Here, the ordering relation arises from taking the transitive closure of the **next** function. The TC approach is more flexible, since we may also take the transitive closure of temporal *relations*, and so represent branching-time temporal logics in the same framework.

Program Verification. TC can be used for the specification and verification of properties of linked data structures and the operation of programs that manipulate them, because it offers a unified constructor for reasoning over both the pointer structures in memory and the sequences of transitions between program states. More concretely, given some definable state transition relation $R(x, y)$ and an initial state s_0, the formula $(RTC_{x,y}\,R)(s_0, s)$ defines all the states the program execution can reach. Additionally, if n_s is a function associating to each memory location its successor in state s, then the formula $(RTC_{x,y}\,y = n_s(x))(x, y)$ defines reachability in memory at state s.

The use of the same constructor for both aspects of the program offers a major improvement on the current formal frameworks which usually use qualitatively different formalisms for describing the operational semantics of programs and the data operated on by the program. For instance, many formalisms employ separation logic to describe the data structures manipulated by programs, but encode the relationship between the program's memory and its operational behaviour via bespoke symbolic-execution inference rules. Another improvement of TC over, e.g., separation logic is its ability to reason over non-tree-like structures such as directed acyclic, or even general, graphs.

3 A FINITARY PROOF SYSTEM FOR $\mathcal{L}_{\mathsf{RTC}}$

We briefly summarise a variation of the finitary proof system for $\mathcal{L}_{\mathsf{RTC}}$ presented in [15, 16]. The key component of the system is an explicit induction rule for *RTC* formulas. All of the systems for $\mathcal{L}_{\mathsf{RTC}}$ presented in the sequel are extensions of $\mathcal{LK}_{=}$, the sequent calculus for classical first-order logic with equality [23, 36], whose proof rules we show in Fig. 1.[2] Sequents are expressions of the form $\Gamma \Rightarrow \Delta$, for finite sets of formulas Γ and Δ. We write Γ, Δ and Γ, φ as a shorthand for $\Gamma \cup \Delta$ and $\Gamma \cup \{\varphi\}$ respectively, and $\mathsf{fv}(\Gamma)$ for the set of free variables of the formulas in the set Γ. Note that since sequents consist of sets of formulas, there is no need for explicit exchange and contraction rules. A sequent $\Gamma \Rightarrow \Delta$ is valid if and only if the formula $\bigwedge_{\varphi \in \Gamma} \varphi \rightarrow \bigvee_{\psi \in \Delta} \psi$ is. We write $\varphi(x_1, \ldots, x_n)$ to emphasise that the formula φ may contain $x_1, \ldots, x_n$ as free variables.

3.1 The Proof System RTC_G

Definition 3.1. The proof system RTC_G for $\mathcal{L}_{\mathsf{RTC}}$ is defined by adding to $\mathcal{LK}_{=}$ the following inference rules where, for Rule (3), $x \notin \mathsf{fv}(\Gamma, \Delta)$ and $y \notin \mathsf{fv}(\Gamma, \Delta, \psi)$:

$$\frac{}{\Gamma \Rightarrow \Delta, (RTC_{x,y}\,\varphi)(s, s)} \tag{1}$$

[2]Here we take $\mathcal{LK}_{=}$ to include the substitution rule, which was not a part of the original systems.

$$\text{(Axiom)}\ \frac{}{\varphi \Rightarrow \varphi} \qquad \text{(WL)}\ \frac{\Gamma \Rightarrow \Delta}{\Gamma, \varphi \Rightarrow \Delta} \qquad \text{(WR)}\ \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \varphi}$$

$$(=\text{L}_1)\ \frac{\Gamma \Rightarrow \varphi\{\frac{s}{x}\}, \Delta}{\Gamma, s = t \Rightarrow \varphi\{\frac{t}{x}\}, \Delta} \qquad (=\text{L}_2)\ \frac{\Gamma \Rightarrow \varphi\{\frac{t}{x}\}, \Delta}{\Gamma, s = t \Rightarrow \varphi\{\frac{s}{x}\}, \Delta} \qquad (=\text{R})\ \frac{}{\Rightarrow t = t}$$

$$\text{(Cut)}\ \frac{\Gamma \Rightarrow \varphi, \Delta \quad \Sigma, \varphi \Rightarrow \Pi}{\Gamma, \Sigma \Rightarrow \Delta, \Pi} \qquad \text{(Subst)}\ \frac{\Gamma \Rightarrow \Delta}{\Gamma\{\frac{t_1}{x_1}, \ldots, \frac{t_n}{x_n}\} \Rightarrow \Delta\{\frac{t_1}{x_1}, \ldots, \frac{t_n}{x_n}\}}$$

$$(\vee\text{L})\ \frac{\Gamma, \varphi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \varphi \vee \psi \Rightarrow \Delta} \qquad (\wedge\text{L})\ \frac{\Gamma, \varphi, \psi \Rightarrow \Delta}{\Gamma, \varphi \wedge \psi \Rightarrow \Delta} \qquad (\rightarrow\text{L})\ \frac{\Gamma \Rightarrow \varphi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \varphi \rightarrow \psi \Rightarrow \Delta}$$

$$(\vee\text{R})\ \frac{\Gamma \Rightarrow \varphi, \psi, \Delta}{\Gamma \Rightarrow \varphi \vee \psi, \Delta} \qquad (\wedge\text{R})\ \frac{\Gamma \Rightarrow \varphi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \varphi \wedge \psi, \Delta} \qquad (\rightarrow\text{R})\ \frac{\Gamma, \varphi \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \varphi \rightarrow \psi, \Delta}$$

$$(\exists\text{L})\ \frac{\Gamma, \varphi \Rightarrow \Delta}{\Gamma, \exists x.\varphi \Rightarrow \Delta}\ x \notin \mathsf{fv}(\Gamma, \Delta) \qquad (\forall\text{L})\ \frac{\Gamma, \varphi\{\frac{t}{x}\} \Rightarrow \Delta}{\Gamma, \forall x.\varphi \Rightarrow \Delta} \qquad (\neg\text{L})\ \frac{\Gamma \Rightarrow \varphi, \Delta}{\Gamma, \neg\varphi \Rightarrow \Delta}$$

$$(\exists\text{R})\ \frac{\Gamma \Rightarrow \varphi\{\frac{t}{x}\}, \Delta}{\Gamma \Rightarrow \exists x.\varphi, \Delta} \qquad (\forall\text{R})\ \frac{\Gamma \Rightarrow \varphi, \Delta}{\Gamma \Rightarrow \forall x.\varphi, \Delta}\ x \notin \mathsf{fv}(\Gamma, \Delta) \qquad (\neg\text{R})\ \frac{\Gamma, \varphi \Rightarrow \Delta}{\Gamma \Rightarrow \neg\varphi, \Delta}$$

Fig. 1. Proof rules for the sequent calculus $\mathcal{LK}_=$ with substitution.

$$\frac{\Gamma \Rightarrow \Delta, (RTC_{x,y}\,\varphi)(s, r) \quad \Gamma \Rightarrow \Delta, \varphi\{\frac{r}{x}, \frac{t}{y}\}}{\Gamma \Rightarrow \Delta, (RTC_{x,y}\,\varphi)(s, t)} \tag{2}$$

$$\frac{\Gamma, \psi(x), \varphi(x, y) \Rightarrow \Delta, \psi\{\frac{y}{x}\}}{\Gamma, \psi\{\frac{s}{x}\}, (RTC_{x,y}\,\varphi)(s, t) \Rightarrow \psi\{\frac{t}{x}\}, \Delta} \tag{3}$$

Rule (3) is a generalized induction principle. It states that if an extension of formula ψ is closed under the relation induced by φ, then it is also closed under the reflexive transitive closure of that relation. In the case of arithmetic it captures the induction rule of Peano's Arithmetics PA [16].

3.2 Soundness and Completeness

The rich expressiveness of TC logic entails that the effective system RTC_G which is sound w.r.t. the standard semantics, cannot be complete (much like the case for LKID). It is however both sound and complete w.r.t. Henkin semantics.

Theorem 3.2 (Soundness and Completeness of RTC_G [14]). *RTC_G is sound for standard semantics, and also sound and complete for Henkin semantics.*

We remark that the soundness proof of LKID is rather complex since it must handle different types of mutual dependencies between the inductive predicates. For RTC_G the proof is much simpler due to the uniformity of the rules for the *RTC* operator. Nonetheless, the completeness proof given in [14] does not establish cut admissibility while the proof for LKID does. We believe that by adapting the technique used in the proof of the completeness of LKID one can obtain cut admissibility for an equivalent system to RTC_G, in which the formalization of Rule (3) is slightly modified, like the induction rule for LKID in [11], to incorporate a cut with the induction formula

ψ as follows:

$$\frac{\Gamma \Rightarrow \Delta, \psi\{\frac{s}{x}\} \quad \Gamma, \psi(x), \varphi(x,y) \Rightarrow \Delta, \psi\{\frac{y}{x}\} \quad \Gamma, \psi\{\frac{t}{x}\} \Rightarrow \Delta}{\Gamma, (RTC_{x,y}\,\varphi)(s,t) \Rightarrow \Delta}$$

where $x \notin \mathrm{fv}(\Gamma, \Delta)$ and $y \notin \mathrm{fv}(\Gamma, \Delta, \psi)$. However, the trade-off is that the cut-free system presented here no longer enjoys the sub-formula property (for a generalized notion of a subformula that incorporates substitution instances), as in $\mathcal{LK}_{=}$. Nonetheless, since the explicit system is not the main focus of the current work, we leave obtaining cut admissibility for future work.

4 INFINITARY PROOF SYSTEMS FOR $\mathcal{L}_{\mathrm{RTC}}$

This section introduces an infinitary proof system, in which *RTC* formulas are simply unfolded, and inductive arguments are represented via infinite descent-style constructions. We establish soundness and completeness of the proof system with respect to the standard semantics, and further identify a subsystem restricted to regular proofs.

4.1 The Proof System RTC_G^ω

Definition 4.1. The infinitary proof system RTC_G^ω for $\mathcal{L}_{\mathrm{RTC}}$ is defined like RTC_G, but replacing Rule (3) by the following case-split rule:

$$\frac{\Gamma, s = t \Rightarrow \Delta \quad \Gamma, (RTC_{x,y}\,\varphi)(s,z), \varphi\{\frac{z}{x}, \frac{t}{y}\} \Rightarrow \Delta}{\Gamma, (RTC_{x,y}\,\varphi)(s,t) \Rightarrow \Delta} \tag{4}$$

where z is fresh, i.e. does not occur free in Γ, Δ, or $(RTC_{x,y}\,\varphi)(s,t)$. The formula $(RTC_{x,y}\,\varphi)(s,z)$ in the right-hand premise is called the *immediate ancestor* (cf. [12, §1.2.3]) of the principal formula, $(RTC_{x,y}\,\varphi)(s,t)$, in the conclusion.

There is an asymmetry between Rule (2), in which the intermediary is an arbitrary term r, and Rule (4), where we use a variable z. This is necessary to obtain the soundness of the cyclic proof system. It is used to show that when there is a counter-model for the conclusion of a rule, then there is also a counter-model for one of its premises that is, in a sense that we make precise below, 'smaller'. In the case that $s \neq t$, using a fresh z allows us to pick from *all* possible counter-models of the conclusion, from which we may then construct the required counter-model for the right-hand premise. If we allowed an arbitrary term r instead, this might restrict the counter-models we can choose from, only leaving ones 'larger' than the one we had for the conclusion. See Lemma 4.7 below for more details.

Proofs in this system are possibly infinite derivation trees. However, not all infinite derivations are proofs: only those that admit an infinite descent argument. Thus we use the terminology 'pre-proof' for derivations.

Definition 4.2 (Pre-proofs). An RTC_G^ω *pre-proof* is a possibly infinite (i.e. non-well-founded) derivation tree formed using the inference rules. A *path* in a pre-proof is a possibly infinite sequence of sequents $s_0, s_1, \ldots (, s_n)$ such that s_0 is the root sequent of the proof, and s_{i+1} is a premise of s_i for each $i < n$.

The following definitions tell us how to track *RTC* formulas through a pre-proof, and allow us to formalize inductive arguments via infinite descent.

Definition 4.3 (Trace Pairs). Let τ and τ' be *RTC* formulas occurring in the left-hand side of the conclusion s and a premise s', respectively, of (an instance of) an inference rule. (τ, τ') is said to be a *trace pair* for (s, s') if the rule is:

- the (Subst) rule, and $\tau = \tau'\theta$ where θ is the substitution associated with the rule instance;

- Rule (4), and either:
  a) τ is the principal formula of the rule instance and τ' is the immediate ancestor of τ, in which case we say that the trace pair is *progressing*;
  b) otherwise, $\tau = \tau'$.
- any other rule, and $\tau = \tau'$.

Definition 4.4 (Traces). A *trace* is a (possibly infinite) sequence of *RTC* formulas. We say that a trace $\tau_1, \tau_2, \ldots (, \tau_n)$ *follows* a path $s_1, s_2, \ldots (, s_m)$ in a pre-proof $\mathcal{P}$ if, for some $k \geq 0$, each consecutive pair of formulas (τ_i, τ_{i+1}) is a trace pair for (s_{i+k}, s_{i+k+1}). If (τ_i, τ_{i+1}) is a progressing pair then we say that the trace *progresses* at i, and we say that the trace is *infinitely progressing* if it progresses at infinitely many points.

Proofs, then, are pre-proofs which satisfy a global trace condition.

Definition 4.5 (Infinite Proofs). A RTC_G^ω *proof* is a pre-proof in which every infinite path is followed by some infinitely progressing trace.

4.2 Soundness and Completeness

The infinitary system RTC_G^ω, in contrast to the finitary system RTC_G, is both sound and complete w.r.t. the standard semantics. To prove soundness, we make use of the following notion of *measure* for *RTC* formulas.

Definition 4.6 (Degree of RTC Formulas). For $\phi \equiv (RTC_{x,y}\,\varphi)(s,t)$, we define $\delta_\phi(M,v) = 0$ if $v(s) = v(t)$, and $\delta_\phi(M,v) = n$ if $v(s) \neq v(t)$ and $a_0, \ldots, a_n$ is a minimal-length sequence of elements in the semantic domain D such that $v(s) = a_0$, $v(t) = a_n$, and $M, v[x := a_i, y := a_{i+1}] \models \varphi$ for $0 \leq i < n$. We call $\delta_\phi(M,v)$ the *degree* of ϕ with respect to the model M and valuation v.

Soundness then follows from the following fundamental lemma.

Lemma 4.7 (Descending Counter-models). *If there exists a standard model M and valuation v that invalidates the conclusion s of (an instance of) an inference rule, then 1) there exists a standard model M' and valuation v' that invalidates some premise s' of the rule; and 2) if (τ, τ') is a trace pair for (s, s') then $\delta_{\tau'}(M', v') \leq \delta_\tau(M, v)$. Moreover, if (τ, τ') is a progressing trace pair then $\delta_{\tau'}(M', v') < \delta_\tau(M, v)$.*

Proof. The cases for the standard $\mathcal{LK}_=$ and substitution rules are straightforward adaptations of those found in e.g. [11].

- The case for Rule (1) follows trivially since it follows immediately from Definition 2.2 that $M, v \models (RTC_{x,y}\,\varphi)(s,s)$ for all M and v.

- For Rule (2), since $M, v \not\models (RTC_{x,y}\,\varphi)(s,t)$ it follows that either $M, v \not\models (RTC_{x,y}\,\varphi)(s,r)$ or $M, v \not\models \varphi\left\{\frac{r}{x}, \frac{t}{y}\right\}$. To see this, suppose for contradiction that both $M, v \models (RTC_{x,y}\,\varphi)(s,r)$ or $M, v \models \varphi\left\{\frac{r}{x}, \frac{t}{y}\right\}$; but then it would follow by Definition 2.2 that $M, v \models (RTC_{x,y}\,\varphi)(s,t)$. We thus take $M' = M$ and $v' = v$, and either the left- or right-hand premise according to whether $M, v \not\models (RTC_{x,y}\,\varphi)(s,r)$ or $M, v \not\models \varphi\left\{\frac{r}{x}, \frac{t}{y}\right\}$.

- For Rule (4), since $M, v \models (RTC_{x,y}\,\varphi)(s,t)$ there are two cases to consider:
  (i) If $v(s) = v(t)$ then we take the left-hand premise with model $M' = M$ and valuation $v' = v$, and so the degree of any *RTC* formula in Γ with respect to M' and v' remains the same.
  (ii) Otherwise, if there are $a_0, \ldots, a_n \in D$ $(n > 0)$ such that $v(s) = a_0$ and $v(t) = a_n$ with $M, v[x := a_i, y := a_{i+1}] \models \varphi$ for $0 \leq i < n$, we then take the right-hand premise, the model $M' = M$ and valuation $v' = v[z := a_{n-1}]$. Note that, without loss of generality, we may assume

a sequence $a_0, \ldots, a_n$ of minimal length, and thus surmise $\delta_{(RTC_{x,y}\,\varphi)(s,t)}(M, v) = n$. Since z is fresh, it follows that $M', v' \models \varphi\left\{\frac{z}{x}, \frac{t}{y}\right\}$ and $M', v'[x := a_i, y := a_{i+1}] \models \varphi$ for $0 \leq i < n-1$. If $n = 1$ then $v'(s) = v'(z) = a_0$ and so $M, v' \models (RTC_{x,y}\,\varphi)(s, z)$; otherwise this is witnessed by the sequence $a_0, \ldots, a_{n-1}$. Thus we also have that $\delta_{(RTC_{x,y}\,\varphi)(s,z)}(M', v') = n - 1$. To conclude, it also follows from z fresh that $M', v' \models \psi$ for all $\psi \in \Gamma$ and $M', v' \not\models \phi$ for all $\phi \in \Delta$; moreover, the degree of any *RTC* formula in Γ remains unchanged with respect to M' and v'. □

As is standard for infinite descent inference systems [9–11, 20, 32, 37], the above result entails the local soundness of the inference rules (in our case, for standard first-order models). The presence of infinitely progressing traces for each infinite path in a $\mathsf{RTC}^{\omega}_{G}$ proof ensures soundness via a standard infinite descent-style construction.

Theorem 4.8 (Soundness of $\mathsf{RTC}^{\omega}_{G}$). *If there is a* $\mathsf{RTC}^{\omega}_{G}$ *proof of* $\Gamma \Rightarrow \Delta$*, then* $\Gamma \Rightarrow \Delta$ *is valid (w.r.t. the standard semantics).*

Proof. Suppose, for contradiction, that $\Gamma \Rightarrow \Delta$ is not valid. Then by Lemma 4.7 there exists an infinite path $\{s_i\}_{i>0}$ in the proof and an infinite sequence of model-valuation pairs $\{\langle M_i, v_i\rangle\}_{i>0}$ such that $\langle M_i, v_i\rangle$ invalidates s_i for each $i > 0$. Since the proof is a valid $\mathsf{RTC}^{\omega}_{G}$ proof, this infinite path is followed by an infinitely progressing trace $\{\tau_i\}_{i>0}$ for which we can take the degree of each formula with respect to its corresponding counter-model to obtain an infinite sequence of natural numbers $\{\delta_{\tau_i}(M_{k+i}, v_{k+i})\}_{i>0}$ (for some $k \geq 0$). By Lemma 4.7 this sequence is decreasing and, moreover, since the trace is infinitely progressing the sequence strictly decreases infinitely often. From the fact that the natural numbers are a well-founded set we derive a contradiction, and thus conclude that $\Gamma \Rightarrow \Delta$ is indeed valid. □

Following a standard technique (as used in e.g. [11]), we can show cut-free completeness of $\mathsf{RTC}^{\omega}_{G}$ with respect to the standard semantics.

Definition 4.9 (Schedule). A *schedule element* E is defined as any of the following:

- a formula of the form $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi$;
- a pair of the form $\langle \forall x\, \varphi, t\rangle$ or $\langle \exists x\, \varphi, t\rangle$ where $\forall x\, \varphi$ and $\exists x\, \varphi$ are formulas and t is a term;
- a tuple of the form $\langle (RTC_{x,y}\,\varphi)(s,t), r, z, \Gamma, \Delta\rangle$ where $(RTC_{x,y}\,\varphi)(s,t)$ is a formula, r is a term, Γ and Δ are finite sequences of formulas, and z is a variable not occurring free in Γ, Δ, or $(RTC_{x,y}\,\varphi)(s,t)$; or
- a tuple of the form $\langle s = t, x, \varphi, n\rangle$ where s, t are terms, x a variable, φ a formula, and $n \in \{1, 2\}$.

A *schedule* is a recursive enumeration of schedule elements in which every schedule element appears infinitely often (these exist since our language is countable).

Each schedule corresponds to an exhaustive search strategy for a cut-free proof for each sequent $\Gamma \Rightarrow \Delta$, via the following notion of a 'search tree'.

Definition 4.10 (Search Tree). Given a schedule $\{E_i\}_{i>0}$, for each sequent $\Gamma \Rightarrow \Delta$ we inductively define an infinite sequence of (possibly open) derivation trees, $\{T_i\}_{i>0}$, such that T_1 consists of the single open node $\Gamma \Rightarrow \Delta$, and each T_{i+1} is obtained by replacing all suitable open nodes in T_i with applications of first axioms and then the left and right inference rules for the formula in the i^{th} schedule element.[3] We show the cases for building T_{i+1} for when E_i corresponds to an *RTC* formula and an equality formula. The cases for when E_i corresponds to a standard compound first-order formula are similar.

- When E_i is of the form $\langle (RTC_{x,y}\,\varphi)(s,t), r, z, \Gamma, \Delta\rangle$, then T_{i+1} is obtained by:

[3]Note that since sequents consist of sets (as opposed to multisets), inference rules are applied with an implicit contraction, i.e., the principal formula is also part of the context and is therefore kept in the premises.

(1) first closing as such any open node that is an instance of an axiom (after left and right weakening, if necessary);

(2) next, replacing every open node $\Gamma', (RTC_{x,y}\,\varphi)(s,t) \Rightarrow \Delta'$ of the resulting tree for which $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ with the derivation:

$$\frac{\Gamma', (RTC_{x,y}\,\varphi)(s,t), s = t \Rightarrow \Delta' \qquad \Gamma', (RTC_{x,y}\,\varphi)(s,t), (RTC_{x,y}\,\varphi)(s,z), \varphi\left\{\frac{z}{x}, \frac{t}{y}\right\} \Rightarrow \Delta'}{\Gamma', (RTC_{x,y}\,\varphi)(s,t) \Rightarrow \Delta'} \quad (4)$$

(3) finally, replacing every open node $\Gamma' \Rightarrow \Delta', (RTC_{x,y}\,\varphi)(s,t)$ of the resulting tree with the derivation:

$$\frac{\Gamma' \Rightarrow \Delta', (RTC_{x,y}\,\varphi)(s,t), (RTC_{x,y}\,\varphi)(s,r) \quad \Gamma' \Rightarrow \Delta', (RTC_{x,y}\,\varphi)(s,t), \varphi\left\{\frac{r}{x}, \frac{t}{y}\right\}}{\Gamma' \Rightarrow \Delta', (RTC_{x,y}\,\varphi)(s,t)} \quad (2)$$

• When E_i is of the form $\langle s = t, x, \varphi, n\rangle$, then T_{i+1} is then obtained by first closing as such any open node that is an instance of an axiom (after left and right weakening, if necessary); and next, if $n = 1$ (resp. $n = 2$), replacing every open node $\Gamma, s = t \Rightarrow \Delta$ in the resulting tree for which $\varphi\left\{\frac{s}{x}\right\} \in \Delta$ (resp. $\varphi\left\{\frac{t}{x}\right\} \in \Delta$) with the appropriate one of the following derivations:

$$\frac{\Gamma, s = t \Rightarrow \Delta, \varphi\left\{\frac{t}{x}\right\}, \varphi\left\{\frac{s}{x}\right\}}{\Gamma, s = t \Rightarrow \Delta, \varphi\left\{\frac{t}{x}\right\}} \ (=\mathrm{L}_1) \qquad \frac{\Gamma, s = t \Rightarrow \Delta, \varphi\left\{\frac{s}{x}\right\}, \varphi\left\{\frac{t}{x}\right\}}{\Gamma, s = t \Rightarrow \Delta, \varphi\left\{\frac{s}{x}\right\}} \ (=\mathrm{L}_2)$$

The limit of the sequence $\{T_i\}_{i>0}$ is a possibly infinite (and possibly open) derivation tree called the *search tree* for $\Gamma \Rightarrow \Delta$ with respect to the schedule $\{E_i\}_{i>0}$, and denoted by T_ω.

Search trees are, by construction, recursive and cut-free. We construct special 'sequents' out of search trees, called *limit sequents*, as follows.

Definition 4.11 (Limit Sequents). When a search tree T_ω is not an RTC_G^ω proof, either: (1) it is not even a pre-proof, i.e. it contains an open node; or (2) it is a pre-proof but contains an infinite branch that fails to satisfy the global trace condition. In case (1) it contains an open node to which, necessarily, no schedule element applies (e.g. a sequent containing only atomic formulas), for which we write $\Gamma_\omega \Rightarrow \Delta_\omega$. In case (2) the global trace condition fails, so there exists an infinite path $\{\Gamma_i \Rightarrow \Delta_i\}_{i>0}$ in T_ω which is followed by no infinitely progressing traces; we call this path an *untraceable branch* of T_ω. We then take the left-most open node ν or untraceable branch β and define $\Gamma_\omega = \bigcup_{i>0} \Gamma_i$ and $\Delta_\omega = \bigcup_{i>0} \Delta_i$. We call $\Gamma_\omega \Rightarrow \Delta_\omega$ the *limit sequent*.

Note that use of the word 'sequent' here is an abuse of nomenclature, since limit sequents may be infinite and thus technically not sequents. However when we say that such a limit sequent is provable, we mean that it has a finite sub-sequent that is provable.

Lemma 4.12. *Limit sequents $\Gamma_\omega \Rightarrow \Delta_\omega$ are not cut-free provable.*

Proof. Straightforward adaptation of the proof of [11, Lemma 6.3]. □

As standard, we use a limit sequent to induce a counter-interpretation, consisting of a Herbrand model of open terms quotiented by the equalities found in the limit sequent.

Definition 4.13 (Quotient Relation). For a limit sequent $\Gamma_\omega \Rightarrow \Delta_\omega$, the relation $\sim$ is defined as the smallest congruence relation on terms such that $s \sim t$ whenever $s = t \in \Gamma_\omega$. We write $[t]$ for the $\sim$-equivalence class of t, i.e. $[t] = \{u \mid t \sim u\}$.

The following property holds of the quotient relation.

Lemma 4.14. *If $t \sim u$, then $\Gamma_\omega \Rightarrow F\left\{\frac{t}{x}\right\}$ is cut-free provable in RTC^ω_G if and only if $\Gamma_\omega \Rightarrow F\left\{\frac{u}{x}\right\}$ is cut-free provable in RTC^ω_G.*

Proof. By induction on the conditions defining $\sim$. We use $\equiv$ to denote syntactic equality on terms, in order to distinguish from formulas $s = t$ asserting equality between (interpretations of) terms.

($t \sim t$): Immediate, since then $t \equiv u$.

($t = u \in \Gamma_\omega$): Assume $\Gamma_\omega \Rightarrow F\left\{\frac{t}{x}\right\}$ is cut-free provable, then we can apply the ($=\mathrm{L}_1$) rule to derive (without cut) $\Gamma_\omega, t = u \Rightarrow F\left\{\frac{u}{x}\right\}$; however notice that $\Gamma_\omega, t = u$ is simply Γ_ω since $t = u \in \Gamma_\omega$ already. The converse direction is symmetric, using rule ($=\mathrm{L}_2$).

($t \sim u \Rightarrow u \sim t$): Immediate, by induction.

($t \sim u \wedge u \sim v \Rightarrow t \sim v$): Straightforward, by induction.

($t_1 \sim u_1 \wedge \ldots \wedge t_n \sim u_n \Rightarrow f(t_1, \ldots, t_n) \sim f(u_1, \ldots, u_n)$): Consider the formula F; clearly there exist formulas $G_1, \ldots, G_n$ and some variable y such that $G_i\left\{\frac{t}{y}\right\} \equiv F\left\{\frac{f(u_1,\ldots,u_{i-1},t_i,\ldots,t_n)}{x}\right\}$ for each $i \leq n$. By induction, each sequent $\Gamma_\omega \Rightarrow G_i\left\{\frac{t_i}{y}\right\}$ is cut-free provable if and only if so too is $\Gamma_\omega \Rightarrow G_i\left\{\frac{u_i}{y}\right\}$. The result then follows since $F\left\{\frac{f(t_1,\ldots,t_n)}{x}\right\} \equiv G_1\left\{\frac{t_1}{y}\right\}$ and $F\left\{\frac{f(u_1,\ldots,u_n)}{x}\right\} \equiv G_n\left\{\frac{u_n}{y}\right\}$, and also $G_i\left\{\frac{u_i}{y}\right\} \equiv G_{i+1}\left\{\frac{t_{i+1}}{y}\right\}$ for each $i < n$. □

We define the counter-interpretation as follows.

Definition 4.15 (Counter-interpretations). Assume a search tree T_ω which is not a RTC^ω_G proof with limit sequent $\Gamma_\omega \Rightarrow \Delta_\omega$. Define a structure $M_\omega = \langle D, I\rangle$ as follows:

- $D = \{[t] \mid \text{t is a term}\}$ (i.e. the set of terms quotiented by the relation $\sim$).
- For every k-ary function symbol f: $I(f)([t_1], \ldots, [t_k]) = [f(t_1, \ldots, t_k)]$
- For every k-ary relation symbol q: $I(q) = \{([t_1], \ldots, [t_k]) \mid q(t_1, \ldots, t_k) \in \Gamma_\omega\}$

We also define a valuation ρ_ω for M_ω by $\rho_\omega(x) = [x]$ for all variables x.

The counter-interpretation $\langle M_\omega, \rho_\omega\rangle$ has the following property, which entails that M_ω is a counter-model for the corresponding sequent $\Gamma \Rightarrow \Delta$ if its search tree T_ω is not a proof.

Lemma 4.16. *If $\psi \in \Gamma_\omega$ then $M_\omega, \rho_\omega \models \psi$; and if $\psi \in \Delta_\omega$ then $M_\omega, \rho_\omega \not\models \psi$.*

Proof. By well-founded induction using the lexicographic ordering of the number of binders (i.e. $\exists$, $\forall$, and *RTC*) in ψ and the structure of ψ. Notice that, by definition, $\rho_\omega(t) = [t]$ for all terms t.

For ψ atomic (i.e. of the form $q(t_1, \ldots, t_k)$), if $\psi \in \Gamma_\omega$ then it follows immediately by Definition 4.15 that $M_\omega, \rho_\omega \models q(t_1, \ldots, t_k)$. If, on the other hand, $\psi \in \Delta_\omega$ then assume for contradiction that indeed $M_\omega, \rho_\omega \models q(t_1, \ldots, t_k)$. It then follows from Definition 4.15 that there are terms $u_1, \ldots, u_k$ such that $q(u_1, \ldots, u_k) \in \Gamma_\omega$ and $u_i \sim t_i$ for each $i \leq k$. Notice that then we can prove $\Gamma_\omega \Rightarrow q(u_1, \ldots, u_k)$ axiomatically, and so it follows by (k applications of) Lemma 4.14 that $\Gamma_\omega \Rightarrow q(t_1, \ldots, t_k)$ is cut-free provable. However, since $q(t_1, \ldots, t_k) \in \Delta_\omega$, this would mean that the limit sequent $\Gamma_\omega \Rightarrow \Delta_\omega$ is cut-free provable, which contradicts Lemma 4.12. Thus we conclude that in fact $M_\omega, \rho_\omega \not\models q(t_1, \ldots, t_k)$.

For ψ an equality formula $s = t$, if $\psi \in \Gamma_\omega$ then we have immediately by Definition 4.15 that $\rho_\omega(s) = \rho_\omega(t)$ and thus that $M_\omega, \rho_\omega \models s = t$ by Definition 2.2. If, on the other hand, $s = t \in \Delta_\omega$, suppose for contradiction that indeed $M_\omega, \rho_\omega \models s = t$. It then follows from Definition 4.15 that $s \sim t$. Since we may derive $\Gamma_\omega \Rightarrow s = s$ axiomatically, it thus follows from Lemma 4.14 that there is a cut-free proof of $\Gamma_\omega \Rightarrow s = t$. However, since $s = t \in \Delta_\omega$ this would mean that the limit sequent $\Gamma_\omega \Rightarrow \Delta_\omega$ is cut-free provable, which contradicts Lemma 4.12. We thus conclude that in fact $M_\omega, \rho_\omega \not\models s = t$.

The cases where ψ is a standard compound first-order formula follow straightforwardly by induction.

In case $\psi = (RTC_{x,y}\,\varphi)(s,t)$, we reason as follows.

• For the first part of the lemma assume $(RTC_{x,y}\,\varphi)(s,t) \in \Gamma_\omega$. Then, by the construction of T_ω, there is at least one occurrence of rule (4) with active formula ψ in the untraceable branch; thus there are two cases:

i) The branch follows the left-hand premise, so there is $s = t \in \Gamma_\omega$. Therefore, by Definition 4.15, $\rho_\omega(s) = \rho_\omega(t)$ and so it follows immediately from Definition 2.2 that $M_\omega, \rho_\omega \models (RTC_{x,y}\,\varphi)(s,t)$.

ii) The branch follows the right-hand premise and so there is some variable z_1 such that both $(RTC_{x,y}\,\varphi)(s,z_1) \in \Gamma_\omega$ and $\varphi\left\{\frac{z_1}{x},\frac{t}{y}\right\} \in \Gamma_\omega$. Again, by the construction of T_ω, the branch must subsequently traverse an instance of rule (4) now with $(RTC_{x,y}\,\varphi)(s,z_1)$ as the principal formula. In fact, since there is no infinitely progressing trace along the untraceable branch, it must traverse only a finite number of instances of rule (4) for which the principal formula is connected via a trace to the formula $(RTC_{x,y}\,\varphi)(s,t)$. Thus there are a finite number of distinct variables $z_1,\ldots,z_n$ $(n > 0)$ with $\varphi\left\{\frac{z_1}{x},\frac{t}{y}\right\} \in \Gamma_\omega$, $\varphi\left\{\frac{z_{i+1}}{x},\frac{z_i}{y}\right\} \in \Gamma_\omega$, for each $i < n$, and $(RTC_{x,y}\,\varphi)(s,z_1) \in \Gamma_\omega$, where the untraceable branch traverses the left-hand branch of an instance of rule (4) with the latter formula as principal from which it follows that also $s = z_n \in \Gamma_\omega$. By the inductive hypothesis $M_\omega, \rho_\omega \models \varphi\left\{\frac{z_1}{x},\frac{t}{y}\right\}$, and $M_\omega, \rho_\omega \models \varphi\left\{\frac{z_{i+1}}{x},\frac{z_i}{y}\right\}$ for each $i < n$. So $M_\omega, \rho_\omega[x := [z_1], y := [t]] \models \varphi$, and $M_\omega, \rho_\omega[x := [z_{i+1}], y := [z_i]] \models \varphi$ for each $i < n$. Moreover, since $s = z_n \in \Gamma_\omega$, we have that $\rho_\omega(s) = \rho_\omega(z_1) = [z_1]$. We then have from Definition 2.2 that $M_\omega, \rho_\omega \models \psi$.

• For the second part of the lemma we first prove, by an inner induction on n, the following auxiliary result for all terms s and t and elements $a_0,\ldots,a_n \in D$ $(n > 0)$:

if $(RTC_{x,y}\,\varphi)(s,t) \in \Delta_\omega$, with $\rho_\omega(s) = a_0$ and $\rho_\omega(t) = a_n$, then there exists some $i < n$ such that $M_\omega, \rho_\omega[x := a_i, y := a_{i+1}] \not\models \varphi$.

($n = 1$): Since $(RTC_{x,y}\,\varphi)(s,t) \in \Delta_\omega$, we have $\varphi\left\{\frac{s}{x},\frac{t}{y}\right\} \in \Delta_\omega$ by construction as the untraceable branch must traverse an instance of rule (2) with $r \equiv s$ and moreover must traverse the right-hand premise (otherwise, we would have $(RTC_{x,y}\,\varphi)(s,s) \in \Delta_\omega$ resulting in the branch being closed by an instance of rule (1)). Thus by the outer induction it follows that $M_\omega, \rho_\omega \not\models \varphi\left\{\frac{s}{x},\frac{t}{y}\right\}$ and thence that $M_\omega, \rho_\omega[x := \rho_\omega(s), y := \rho_\omega(t)] \not\models \varphi$ as required.

($n = k+1$, $k > 0$): Then there exists some term r such that $a_k = [r] = \rho_\omega(r)$. If $(RTC_{x,y}\,\varphi)(s,t) \in \Delta_\omega$, then by construction of the search tree T_ω we also have that either $(RTC_{x,y}\,\varphi)(s,r) \in \Delta_\omega$ or $\varphi\left\{\frac{r}{x},\frac{t}{y}\right\} \in \Delta_\omega$, as the untraceable branch must traverse an instance of rule (2) for the term r. In the case of the former, the required result holds by the inner induction. In the case of the latter, we have $M_\omega, \rho_\omega \not\models \varphi\left\{\frac{r}{x},\frac{t}{y}\right\}$ by the outer induction and thence that $M_\omega, \rho_\omega[x := \rho_\omega(r), y := \rho_\omega(t)] \not\models \varphi$; i.e. $M_\omega, \rho_\omega[x := a_k, y := a_{k+1}] \not\models \varphi$ as required.

We now show that the primary result holds. Assume $(RTC_{x,y}\,\varphi)(s,t) \in \Delta_\omega$ and suppose for contradiction that $M_\omega, \rho_\omega \models (RTC_{x,y}\,\varphi)(s,t)$ holds. Thus, by Definition 2.2, there are two cases to consider.

– If $\rho_\omega(s) = \rho_\omega(t)$ then $s \sim t$. Thus since we may derive $\Gamma_\omega \Rightarrow (RTC_{x,y}\,\varphi)(s,s)$ by applying rule (1), by Lemma 4.14 there must also be a cut-free proof of $\Gamma_\omega \Rightarrow (RTC_{x,y}\,\varphi)(s,t)$. However, since $(RTC_{x,y}\,\varphi)(s,t) \in \Delta_\omega$ this would imply that $\Gamma_\omega \Rightarrow \Delta_\omega$ is cut-free provable, which contradicts Lemma 4.12.

– Otherwise, there are $a_0, \ldots, a_n \in D$ $(n > 0)$ such that $\rho_\omega(s) = a_0$, $\rho_\omega(t) = a_n$ and $M_\omega, \rho_\omega[x := a_i, y := a_{i+1}] \models \varphi$ for each $i < n$. However this directly contradicts the auxiliary result proved above.

In both cases, we have derived a contradiction, and so we conclude that $M_\omega, \rho_\omega \not\models (RTC_{x,y}\, \varphi)(s, t)$ as required. □

The completeness result therefore follows since, by construction, a sequent S is contained within its corresponding limit sequents.

Theorem 4.17 (Completeness). RTC^ω_G *is complete for standard semantics.*

Proof. Now given any sequent S, if some search tree T_ω contracted for S is not an RTC^ω_G proof then it follows from Lemma 4.16 that S is not valid (M_ω is a counter model for it). Thus if S is valid, then T_ω is a recursive RTC^ω_G proof for it. □

We obtain admissibility of cut for the full infinitary system as the search tree T_ω is cut-free.

Corollary 4.18 (Cut admissibility). *Cut is admissible in* RTC^ω_G.

4.3 The Proof System CRTC^ω_G

In general, one cannot reason effectively about infinite proofs as in RTC^ω_G. In order to do so we need to restrict our attention to those proof trees which are finitely representable. These are the *regular* infinite proof trees, which contain only finitely many *distinct* subtrees. They can be specified as systems of recursive equations or, alternatively, as cyclic *graphs* [19]. Note that a given regular infinite proof may have many different graph representations. One possible way of formalizing such proof graphs is as standard proof trees containing open nodes (called buds), to each of which is assigned a syntactically equal internal node of the proof (called a companion).

The cyclic proof system CRTC^ω_G for $\mathcal{L}_{\mathsf{RTC}}$ is (essentially) the subsystem of RTC^ω_G comprising of all and only the finite and *regular* infinite proofs (i.e. those proofs that can be represented as finite, possibly cyclic, graphs).

Definition 4.19 (Cyclic Proofs). A CRTC^ω_G *pre-proof* is a pair $\langle P, f \rangle$, where P is a finite derivation tree formed using the inference rules of RTC^ω_G and f is a function assigning a companion to every bud node in P. The graph associated with a pre-proof is the one induced by P by identifying each bud node with its companion. A CRTC^ω_G *proof* is then a CRTC^ω_G pre-proof whose graph satisfies the global trace condition, i.e., every infinite path is followed by some infinitely progressing trace.

It is decidable whether a cyclic pre-proof satisfies the global trace condition, using a construction involving an inclusion between Büchi automata (see, e.g., [9, 34]). However since this requires complementing Büchi automata (a PSPACE procedure), our system cannot be considered a proof system in the Cook-Reckhow sense [18]. The problem of deciding whether the global trace condition holds for a cyclic proofs in a fragment of linear logic with fixed points has in fact recently been shown to be PSPACE-complete [30]. We also think it likely that a similar technique will serve to give similar lower bounds for our system, as well as others. Notwithstanding, checking the trace condition for cyclic proofs found in practice is not prohibitive [32, 37].

Since every CRTC^ω_G proof is also a RTC^ω_G proof, soundness of CRTC^ω_G is an immediate corollary of Theorem 4.8.

Corollary 4.20 (Soundness of CRTC^ω_G). *If there is a* CRTC^ω_G *proof of* $\Gamma \Rightarrow \Delta$, *then* $\Gamma \Rightarrow \Delta$ *is valid (w.r.t. the standard semantics).*

Notice that the construction of Definition 4.10 does not necessarily produce CRTC_G^ω pre-proofs, and so we do not obtain a completeness result using this technique. Indeed, since one may encode arithmetic in TC and the set of CRTC_G^ω proofs is recursively enumerable, CRTC_G^ω cannot be complete w.r.t. the standard semantics. In Section 6 below we show that CRTC_G^ω subsumes RTC_G, which entails its Henkin-completeness. However, the question of whether CRTC_G^ω is *sound* w.r.t. Henkin semantics remains open.

5 VARIANTS OF $\mathcal{L}_{\mathsf{RTC}}$

We now present two important variants of TC logic—one with an assumed pairing function, and one without a designated equality symbol—and show how the theory of the previous sections extends to them.

5.1 $\mathcal{L}_{\mathsf{RTC}}$ with Pairs

To obtain full inductive expressivity we must allow the formation of the transitive closure of not only binary relations, but any $2n$-ary relation. In [2] it was shown that taking a $2n$-ary operator RTC^n for every $n \geq 1$ results in a more expressive logic, namely one that captures all finitary first-order definable inductive definitions and relations. However one may argue that, from a proof theoretical point of view, having infinitely many such operators is sub-optimal: the language is no longer generated using a finite signature and proof systems must also contain an infinite number of rule schemata.

Instead, we incorporate the notion of ordered pairs and use it to encode such operators. For example, writing $\langle x, y\rangle$ for the application of the pairing function $\langle\rangle(x, y)$, the $2n$-ary RTC-formula $(RTC^2_{x_1,x_2,y_1,y_2}\,\varphi)(s_1, s_2, t_1, t_2)$ can be encoded by:

$$(RTC_{x,y}\,\exists x_1, x_2, y_1, y_2\,.\,x = \langle x_1, x_2\rangle \wedge y = \langle y_1, y_2\rangle \wedge \varphi)(\langle s_1, s_2\rangle, \langle t_1, t_2\rangle)$$

Accordingly, we may assume languages that explicitly contain a pairing function, providing that we (axiomatically) restrict to structures that interpret it as such (i.e. the *admissible* structures). For such languages we can consider two induced semantics: admissible standard semantics and admissible Henkin semantics, obtained by restricting the (first-order part of the) structures to be admissible.

The proof systems we have considered above can be extended to capture ordered pairs as follows.

Definition 5.1. For a signature containing at least one constant c, and a binary function symbol denoted by $\langle\rangle$, the proof systems $\langle\mathsf{RTC}\rangle_G$, $\langle\mathsf{RTC}\rangle_G^\omega$, and $\langle\mathsf{CRTC}\rangle_G^\omega$ are obtained from RTC_G, RTC_G^ω, CRTC_G^ω (respectively) by the addition of the following rules:

$$\frac{\Gamma \Rightarrow \langle x, y\rangle = \langle u, v\rangle, \Delta}{\Gamma \Rightarrow x = u \wedge y = v, \Delta} \qquad \frac{}{\Gamma, \langle x, y\rangle = c \Rightarrow \Delta}$$

The proofs of Theorems 3.2 and 4.17 can easily be extended to obtain the following results for languages with a pairing function. For completeness, the key observation is that the model comprising the counter-interpretation is one in which every binary function is a pairing function. That is, the interpretation of any binary function is such that satisfies the standard pairing axioms. Therefore, the model of the counter-interpretation is an admissible structure.

Theorem 5.2 (Soundness and Completeness of $\langle\mathsf{RTC}\rangle_G$ and $\langle\mathsf{RTC}\rangle_G^\omega$). *The proof systems $\langle\mathsf{RTC}\rangle_G$ and $\langle\mathsf{RTC}\rangle_G^\omega$ are both sound and complete for the admissible forms of Henkin and standard semantics, respectively.*

5.2 $\mathcal{L}_{\mathsf{RTC}}$ without Equality

For the sake of simplicity of presentation we have thus far included a designated equality symbol in the language $\mathcal{L}_{\mathsf{RTC}}$. Nonetheless, under both the standard semantics and Henkin semantics for

the language, the equality relation is definable and thus there is no need to explicitly include it in our languages. The proof-theoretic results obtained in the previous sections for languages that do assume an equality symbol can be adapted without too much difficulty to hold for languages without equality. This is a noteworthy added degree of expressivity over other logics.

5.2.1 Adaptations of the Language and Proof Rules. We remove $s = t$ as atomic formulas of the language, and instead include a constant $\bot$, whose interpretation is defined such that $M, v \models \bot$ never holds.[4] For the Henkin semantics, we must also add to the parametric closure condition in the definition of Henkin structures (Definition 2.5) the requirement that $\{a\} \in \mathcal{D}$ for all $a \in D$. This is to ensure that Henkin models are fine-grained enough to allow individual elements to be distinguished via formula-definability. Equality is then definable under both semantics by:

$$s = t := (RTC_{x,y}\, \bot)(s, t). \tag{5}$$

We also modify the proof systems slightly by removing the three equality rules of $\mathcal{LK}_=$ and instead including the following rule:

$$(\bot)\ \frac{}{\bot \Rightarrow}$$

While the rules pertaining to *RTC* formulas in the finitary system remain unchanged, cf. Rules (1) to (3), in the infinitary systems Rule (4) is reformulated as follows

$$\frac{\Gamma'\left\{\frac{s}{v}, \frac{t}{w}\right\} \Rightarrow \Delta'\left\{\frac{s}{v}, \frac{t}{w}\right\} \quad \Gamma, (RTC_{x,y}\, \varphi)(s, z), \varphi\left\{\frac{z}{x}, \frac{t}{y}\right\} \Rightarrow \Delta}{\Gamma, (RTC_{x,y}\, \varphi)(s, t) \Rightarrow \Delta} \tag{6}$$

where z is fresh, i.e. z does not occur free in Γ, Δ, or $(RTC_{x,y}\, \varphi)(s, t)$; and Γ' and Δ' are such that $\Gamma = \Gamma'\left\{\frac{t}{v}, \frac{s}{w}\right\}$ and $\Delta = \Delta'\left\{\frac{t}{v}, \frac{s}{w}\right\}$. This formulation essentially combines the *RTC* case-split rule with the equality rules: in the left-hand premise, corresponding to the case that $s = t$, we may swap occurrences of s in the context for t, and vice-versa. As a consequence, the definition of trace pairs for this rule becomes slightly more complex.

Definition 5.3 (Trace Pairs for $\mathcal{L}_{\mathrm{RTC}}$ without Equality). Let τ and τ' be *RTC* formulas occurring in the left-hand side of the conclusion s and a premise s', respectively, of (an instance of) an inference rule. (τ, τ') is said to be a *trace pair* for (s, s') if the rule is:

- the (Subst) rule, and $\tau = \tau'\theta$ where θ is the substitution associated with the rule instance;
- Rule (6), and either:
  a) τ is the principal formula of the rule instance and τ' is the immediate ancestor of τ, in which case we say that the trace pair is *progressing*;
  b) s' is the right-hand premise and $\tau = \tau'$; or
  c) s' is the left-hand premise and $\tau' = \tau''\left\{\frac{s}{v}, \frac{t}{w}\right\}$ for some $\tau'' \in \Gamma'$.[5]
- any other rule, and $\tau = \tau'$.

The proof of soundness (Theorem 4.8) can be trivially adapted for this modified infinitary system.

5.2.2 Derivability of the Equality Rules. In these variations of both the finitary and infinitary systems, the equality rules of $\mathcal{LK}_=$ become derivable using the *RTC*-defined equality. Rule (=R) is derivable since $t = t$ stands for $(RTC_{x,y}\, \bot)(t, t)$, which is provable using Rule (1). In the finitary variation, Rule ($=\mathrm{L}_1$) is derivable as follows:

$$\dfrac{\Gamma \Rightarrow \Delta, \psi\left\{\frac{s}{x}\right\} \quad \dfrac{\dfrac{}{\Gamma, \psi, \bot \Rightarrow \Delta, \psi\left\{\frac{y}{x}\right\}}\ (\bot\text{+WL+WR}) \quad \Gamma, \psi\left\{\frac{s}{x}\right\}, (RTC_{x,y}\, \bot)(s, t) \Rightarrow \Delta, \psi\left\{\frac{t}{x}\right\}}{\Gamma, (RTC_{x,y}\, \bot)(s, t) \Rightarrow \Delta, \psi\left\{\frac{t}{x}\right\}}\ (3)}{\Gamma, (RTC_{x,y}\, \bot)(s, t) \Rightarrow \Delta, \psi\left\{\frac{t}{x}\right\}}\ (\text{Cut})$$

[4]Note that the inclusion of $\bot$ is itself also only a notational convenience since it may be encoded via any contradiction.
[5]Here, Γ', v, w, s and t refer to the instantiations of these same meta-variables appearing in the schema of Rule (6).

and Rule (=L_2) is derived dually, as follows:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, \psi\{\frac{s}{x}\} \Rightarrow \Delta, \psi\{\frac{s}{x}\}}\,(\text{Ax+WL+WR})}{\Gamma \Rightarrow \Delta, \psi\{\frac{s}{x}\}, \neg\psi\{\frac{s}{x}\}}\,(\neg\text{R}) \qquad \dfrac{}{\Gamma, \psi, \bot \Rightarrow \Delta, \psi\{\frac{y}{x}\}}\,(\bot\text{+WL+WR})}{\Gamma, \neg\psi\{\frac{s}{x}\}, (RTC_{x,y}\,\bot)(s,t) \Rightarrow \Delta, \neg\psi\{\frac{t}{x}\}}\,(3) \qquad \dfrac{\Gamma \Rightarrow \Delta, \psi\{\frac{t}{x}\}}{\Gamma, \neg\psi\{\frac{t}{x}\} \Rightarrow \Delta}\,(\neg\text{L})}{\dfrac{\Gamma, (RTC_{x,y}\,\bot)(s,t) \Rightarrow \Delta, \psi\{\frac{s}{x}\}, \neg\psi\{\frac{t}{x}\}}{\Gamma, (RTC_{x,y}\,\bot)(s,t) \Rightarrow \Delta, \psi\{\frac{s}{x}\}}\,(\text{Cut})}\,(\text{Cut})$$

In the infinitary variations, both Rule (=L_1) and Rule (=L_2) are simply instances of Rule (6). For example, Rule (=L_1) is derivable as follows:

$$\dfrac{\Gamma \Rightarrow \Delta, \psi\{\frac{s}{x}\} \quad \dfrac{}{\Gamma, (RTC_{x,y}\,\bot)(s,z), \bot \Rightarrow \Delta, \psi\{\frac{t}{x}\}}\,(\bot\text{+WL+WR})}{\Gamma, (RTC_{x,y}\,\bot)(s,t) \Rightarrow \Delta, \psi\{\frac{t}{x}\}}\,(6)$$

where the only occurrences of t in the context of the conclusion that are swapped for s in the left-hand premise are those in the formula ψ (and no occurrences of s in the conclusion are swapped for t). Rule (=L_2) is derived symmetrically.

5.2.3 Completeness. The adaptation of the completeness proof for the finitary variation is standard. Completeness of the infinitary system follows from the fact that, in addition to the equality rules derived above, we can also derive the original form of the case-split rule as follows.

$$\dfrac{\dfrac{}{\Rightarrow (RTC_{x,y}\,\bot)(s,s)}\,(1) \qquad \dfrac{\Gamma, (RTC_{x,y}\,\bot)(s,t) \Rightarrow \Delta \qquad \dfrac{\Gamma, (RTC_{x,y}\,\varphi)(s,z), \varphi\{\frac{z}{x}, \frac{t}{y}\} \Rightarrow \Delta}{\Gamma, (RTC_{x,y}\,\bot)(s,s), (RTC_{x,y}\,\varphi)(s,z), \varphi\{\frac{z}{x}, \frac{t}{y}\} \Rightarrow \Delta}\,(\text{WL})}{\Gamma, (RTC_{x,y}\,\bot)(s,s), (RTC_{x,y}\,\varphi)(s,t) \Rightarrow \Delta}\,(6)}{\Gamma, (RTC_{x,y}\,\varphi)(s,t) \Rightarrow \Delta}\,(\text{Cut})$$

Notice that this uses a cut, so we do not obtain cut-free completeness this way. The technique used in Section 4.2 to show cut-admissibility of RTC_G^ω does not immediately apply in the setting without equality since the limit sequent no longer contains formulas explicitly witnessing equalities between terms. Although we believe the technique can be adapted for the case without explicit equality, since this is tangential to our main contributions we leave this for future work.

6 RELATING THE FINITARY AND INFINITARY PROOF SYSTEMS

This section discusses the relation between the explicit and the cyclic system for TC. In Section 6.1 we show that the former is contained in the latter. The converse direction, which is much more subtle, is discussed in Section 6.2.

6.1 Inclusion of RTC_G in CRTC_G^ω

Provability in the explicit induction system implies provability in the cyclic system. The key property is that we can derive the explicit induction rule in the cyclic system, as shown in Figure 2.

Lemma 6.1. *Rule (3) is derivable in* CRTC_G^ω.

This leads to the following result (an analogue to [11, Theorem 7.6]).

Theorem 6.2. $\mathsf{CRTC}_G^\omega \supseteq \mathsf{RTC}_G$*, and is thus complete w.r.t. Henkin semantics.*

Proof. Let $\mathcal{P}$ be a proof in RTC_G and $\mathcal{P}'$ be the corresponding pre-proof in CRTC_G^ω obtained by replacing each instance of Rule (3) by the corresponding instance of the proof schema given in Lemma 6.1. We argue that $\mathcal{P}'$ is a valid CRTC_G^ω proof. Since the only cycles in $\mathcal{P}'$ are internal to the subproofs that simulate Rule (3), any infinite path in $\mathcal{P}'$ must eventually end up traversing one of these cycles infinitely often. Therefore, it suffices to show that there is an infinitely progressing

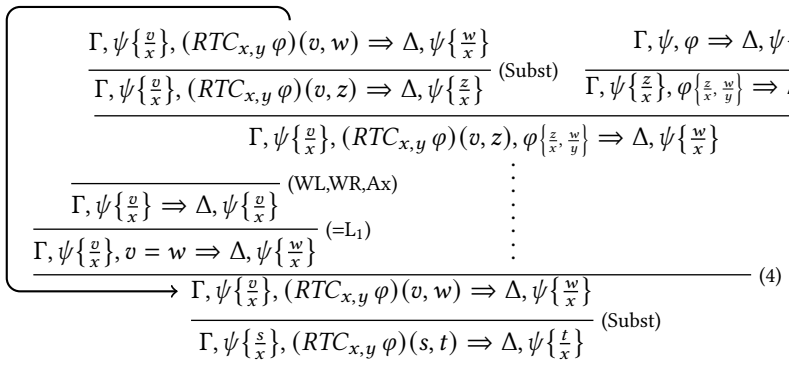


Fig. 2. $\mathsf{CRTC}^{\omega}_{G}$ derivation simulating Rule (3). The variables v and w are fresh (i.e. not free in Γ, Δ, φ, or ψ).

trace following each such path. This is clearly the case since we can trace the active *RTC* formulas along these paths, which progress once each time around the cycle, across Rule (4). □

Lemma 6.1 is the TC counterpart of [11, Lemma 7.5]. It is interesting to note that the simulation of the explicit LKID induction rule in the cyclic LKID system is rather complex since each predicate has a slightly different explicit induction rule, which depends on the particular productions defining it. Thus, the construction for the cyclic LKID system must take into account the possible forms of arbitrary productions. In contrast, $\mathsf{CRTC}^{\omega}_{G}$ provides a single, uniform way to unfold an *RTC* formula: the construction given in Fig. 2 is the cyclic representation of the *RTC* operator semantics, with the variables v and w implicitly standing for arbitrary terms, which we then subsequently substitute for the particular terms begin reasoned over.

This uniform syntactic translation of the explicit RTC_G induction rule into $\mathsf{CRTC}^{\omega}_{G}$ allows us to *syntactically* identify a proper subset of cyclic proofs which is also complete w.r.t. Henkin semantics.[6] The criterion we use is based on the notion of *overlapping cycles*. Recall the definition of a *basic* cycle, which is a path in a (proof) graph starting and ending at the same point, but containing no other repeated nodes. We say that two distinct basic cycles (i.e. ones not identical up to cyclic permutation) *overlap* if they share any nodes in common; that is, at some point they both traverse the same path in the graph. We say that a cyclic proof is *non-overlapping* whenever no two distinct basic cycles it contains overlap. The restriction to non-overlapping proofs has an advantage for automation, since one has only to search for cycles in one single branch.

Definition 6.3 (Normal Cyclic Proofs). The *normal* cyclic proof system $\mathsf{NCRTC}^{\omega}_{G}$ is the subsystem of $\mathsf{RTC}^{\omega}_{G}$ comprising of all and only the non-overlapping cyclic proofs.

The following theorem is immediate due to the fact that the translation of an RTC_G proof into $\mathsf{CRTC}^{\omega}_{G}$, using the construction shown in Figure 2, results in a proof with no overlapping cycles.

Theorem 6.4. $\mathsf{NCRTC}^{\omega}_{G} \supseteq \mathsf{RTC}_G$.

Henkin-completeness of the normal cyclic system then follows from Theorem 6.4 and Theorem 3.2.

6.2 Inclusions of $\mathsf{CRTC}^{\omega}_{G}$ in RTC_G

This section addresses the question of whether the cyclic system is equivalent to the explicit one, or strictly stronger. In [11] it was conjectured that for the system with inductive definitions, LKID

[6] Note that it is not clear that a similar complete structural restriction is possible for LKID.

and CLKID$^\omega$ are equivalent. Later, it was shown that they are indeed equivalent when containing arithmetics [7, 34]. We obtain a corresponding theorem in Section 6.2.1 for the TC systems. However, it was also shown in [6] that in the general case the cyclic system is stronger than the explicit one. We discuss the general case for TC and its subtleties in Section 6.2.2.

6.2.1 The Case of Arithmetics. Here we show equivalence of the systems RTC_G+A and CRTC^ω_G+A for languages $\mathcal{L}_{\mathsf{RTC}}$ based on the signature $\{0, \mathbf{s}, +\}$, obtained by adding to RTC_G and CRTC^ω_G, respectively, the standard axioms of Peano arithmetic (PA) together with the *RTC* characterization of the natural numbers[7], i.e.:

i) $\mathbf{s}x = 0 \Rightarrow$
ii) $\mathbf{s}x = \mathbf{s}y \Rightarrow x = y$
iii) $\Rightarrow x + 0 = x$
iv) $\Rightarrow x + \mathbf{s}y = \mathbf{s}(x + y)$
v) $\Rightarrow (RTC_{w,u}\, \mathbf{s}w = u)(0, x)$

Note that we do not need to assume multiplication explicitly in the signature, nor add axioms for it, since multiplication is definable in $\mathcal{L}_{\mathsf{RTC}}$ and its standard axioms are derivable in RTC_G+A [2, 16] and thus also in CRTC^ω_G+A. In the presence of pairs, addition is also definable in $\mathcal{L}_{\mathsf{RTC}}$ for languages based on the signature $\{0, \mathbf{s}\}$, and corresponding versions of axioms (iii) and (iv) are derivable.

Furthermore, recall that we can express facts about sequences of numbers in PA by using a β-function such that for any finite sequence $k_0, k_1, ..., k_n$ there is some c such that for all $i \leq n$, $\beta(c, i) = k_i$. Accordingly, let B be a well-formed formula of the language of PA with three free variables which captures in PA a β-function. The β-translation of a formula φ in $\mathcal{L}_{\mathsf{RTC}}$ is defined inductively. For atomic formulas $\varphi^\beta = \varphi$, the translation is homomorphic with respect to the first-order logical connectives, and for *RTC*-formulas $((RTC_{x,y}\, \varphi)(s, t))^\beta$ is defined as:

$$s = t \vee \left(\exists z, c\, .\, B(c, 0, s) \wedge B(c, \mathbf{s}z, t) \wedge \left(\forall u \leq z\, .\, \exists v, w\, .\, B(c, u, v) \wedge B(c, \mathbf{s}u, w) \wedge \varphi^\beta\left\{\tfrac{v}{x}, \tfrac{w}{y}\right\}\right)\right).$$

Proof Outline. We demonstrate the equivalence of RTC_G+A and CRTC^ω_G+A in two stages. First, we use a result from [13, 16] that RTC_G+A is equivalent to Gentzen's system PA_G for Peano arithmetic, modulo translation via the β-function. It is mainly based on the fact that in RTC_G+A all instances of the PA_G induction rule are derivable.

Theorem 6.5 (cf. [16]). *The following hold.*
(1) $\vdash_{\mathsf{RTC}_G+\mathsf{A}} \varphi \Leftrightarrow \varphi^\beta$.
(2) $\vdash_{\mathsf{RTC}_G+\mathsf{A}} \Gamma \Rightarrow \Delta$ *iff* $\vdash_{\mathsf{PA}_G} \Gamma^\beta \Rightarrow \Delta^\beta$.

Secondly we show that, also modulo translation via the β-function, CRTC^ω_G+A is equivalent to Simpson's Gentzen-style cyclic system CA_G for arithmetic. Simpson's result [34] that CA_G is equivalent to PA_G provides the final link in the chain. This is summarised in Fig. 3 below.

Cyclic Arithmetic. We reprise the salient details of the cyclic system CA_G for arithmetic introduced in [34]. Sequents $\Gamma \Rightarrow \Delta$ of CA_G consist of sets Γ and Δ of formulas of the language over the signature $\{0, \mathbf{s}, +, \cdot, <, =\}$. Pre-proofs of CA_G are the regular, non-wellfounded derivation trees formed using the standard inference rules of Gentzen's system $\mathcal{LK}$ (with substitution), as well as the following inference rules

$$\text{(Eq)}\ \frac{\Gamma\left\{\tfrac{u}{x}, \tfrac{t}{y}\right\} \Rightarrow \Delta\left\{\tfrac{u}{x}, \tfrac{t}{y}\right\}}{\Gamma\left\{\tfrac{t}{x}, \tfrac{u}{y}\right\}, t = u \Rightarrow \Delta\left\{\tfrac{t}{x}, \tfrac{u}{y}\right\}} \qquad \text{(Ind)}\ \frac{\Gamma, t = \mathbf{s}x \Rightarrow \Delta}{\Gamma, 0 < t \Rightarrow \Delta}\ x \text{ is fresh}$$

[7] Axioms are added as new inference rules without any premises.

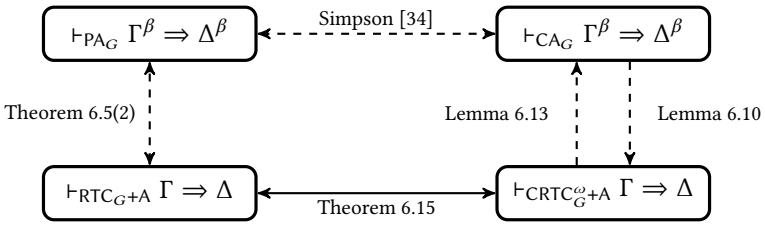


Fig. 3. The structure of the equivalence proof for RTC_G+A and CRTC^ω_G+A.

and the following axiom schemas:

$$
\begin{array}{lll}
t < u, u < v \Rightarrow t < v & t < 0 \Rightarrow & \Rightarrow t + 0 = t \\
t < u, u < t \Rightarrow & t < u \Rightarrow \mathbf{s}t < \mathbf{s}u & \Rightarrow t + \mathbf{s}u = \mathbf{s}(t + u) \\
t < u, u < \mathbf{s}t \Rightarrow & \Rightarrow t < \mathbf{s}t & \Rightarrow t \cdot 0 = 0 \\
 & \Rightarrow t < u, t = u, u < t & \Rightarrow t \cdot \mathbf{s}u = (t \cdot u) + t
\end{array}
$$

Traces of CA_G consist of *terms*, rather than formulas as in CRTC^ω_G. If sequents S and S' are the conclusion and premise, respectively, of a CA_G inference rule, and t and t' are terms occurring, respectively, in S and S' (possibly as subterms), then we say that t' is a *precursor* of t when the following holds:

- if S and S' are the conclusion and premise of an instance of the (Subst) rule, then $t = t'\theta$, where θ is the substitution applied in the rule instance;
- if S and S' are the conclusion and premise of an instance of the (Eq) rule with principal formula $u = v$, then $t = t''\left\{\frac{u}{x}, \frac{v}{y}\right\}$ and $t' = t''\left\{\frac{v}{x}, \frac{u}{y}\right\}$ for some term t''; and
- if S and S' are the conclusion and premise of an instance of any other rule, then $t' = t$.

A pair of terms (t, t') is a CA_G trace pair for (S, S') if either t' is a precursor of t or there is a formula $t' < t''$ in the antecedent of S' with t'' a precursor of t. In the latter case, the trace pair is called progressing. We say that a CA_G trace $t_0, t_1, \ldots (, t_n)$ follows a path $S_0, S_1, \ldots (, S_m)$ in a CA_G pre-proof when there is some $k \geq 0$ such that (t_i, t_{i+1}) is a trace pair for (S_{i+k}, S_{i+k+1}), for all $i \geq 0$. A CA_G proof is a pre-proof in which every infinite path is followed by a trace containing infinitely many progressing trace pairs.

Inclusion of CA_G in CRTC^ω_G+A. We give a translation of CA_G proofs into CRTC^ω_G+A proofs. Since CRTC^ω_G subsumes RTC_G in the general case (Theorem 6.2), this translation is not actually necessary to prove the main result of this section (Theorem 6.15). However we show the inclusion of CA_G in CRTC^ω_G+A as part of a proof of the equivalence of CA_G and CRTC^ω_G+A, which stands as an interesting result in its own right. This provides a fuller picture, mirroring the equivalence between PA_G and RTC_G+A, and establishes a correlation between the different forms of tracing in the two systems: namely via terms and via formulas.

Technically, the signature of CA_G includes the relation symbol $<$ for strict ordering, and the function symbol '$\cdot$' for multiplication. As mentioned above, multiplication (and its axioms) are derivable in RTC_G+A. The strict ordering on natural numbers $s < t$ is definable in $\mathcal{L}_{\mathsf{RTC}}$ as $s \neq t \wedge (RTC_{x,y}\, \mathbf{s}x = y)(s, t)$, and its standard axioms are also derivable in RTC_G+A. Therefore, in the following result, we implicitly assume that all CA_G terms of the form $s \cdot t$ and $s < t$ are translated in CRTC^ω_G into their defining formulas in $\mathcal{L}_{\mathsf{RTC}}$.

We call an occurrence of a (sub)term *free* if it does not contain any variables bound by quantifiers under whose scope it occurs.

Definition 6.6 ($[\cdot]^*_X$-translation). Let X be a finite set of variables, and fix an injection, $\hat{\cdot}$, from the set of terms into the set of variables $\mathcal{V} \setminus X$. Then, for a set of formulas Γ, define Γ^*_X to be the smallest set satisfying $\Gamma \subseteq \Gamma^*_X$ and, for all free (sub)terms t occurring in formulas in Γ, $t = \hat{t} \in \Gamma^*_X$ and $(RTC_{x,y}\, \mathbf{s}x = y)(\mathbf{0}, \hat{t}) \in \Gamma^*_X$. For $S = \Gamma \Rightarrow \Delta$, we define $S^*_X = \Gamma^*_X \Rightarrow \Delta$.

To prove the inclusion, notice that it suffices to prove that if X is the set of free variables occurring in a CA_G proof of $\Gamma \Rightarrow \Delta$ then there is a CRTC^ω_G+A proof of $\Gamma^*_X \Rightarrow \Delta$. Thence we may obtain a CRTC^ω_G+A proof of $\Gamma \Rightarrow \Delta$ by cutting all the added formulas $(RTC_{x,y}\, \mathbf{s}x = y)(0, \hat{t})$ in Γ^*_X using instances of axiom (v), then introducing existential quantifiers binding all variables $\hat{t}$ and cutting the resulting formulas $\exists z.t = z$.

The key to this is showing that the $[\cdot]^*_X$-translations of the CA_G axioms and inference rules are derivable in CRTC^ω_G+A in such a way that (progressing) CA_G trace pairs are simulated by (progressing) CRTC^ω_G traces.

Lemma 6.7. *Let S and $S_1, \ldots, S_n$ be the conclusion and the premises, respectively, of an instance of a CA_G inference rule or axiom, and X a superset of the set of free variables occurring therein. The following inference rule is derivable in CRTC^ω_G+A*

$$\frac{(S_1)^*_X \quad \ldots \quad (S_n)^*_X}{(S)^*_X}$$

such that the following hold.

1. *If t' is a precursor (in S_i) of t (in S) then each path in the derived inference rule from the conclusion to the premise corresponding to S_i is followed by a CRTC^ω_G trace starting with the formula $(RTC_{x,y}\, \mathbf{s}x = y)(0, \hat{t})$ and ending with the formula $(RTC_{x,y}\, \mathbf{s}x = y)(0, \hat{t'})$. Furthermore, all infinite paths in the derived inference rule are followed by infinitely progressing (CRTC^ω_G) traces.*
2. *If (t, t') is a (progressing) CA_G trace pair for (S, S_i) then each path in the derived inference rule from the conclusion to the premise corresponding to S_i is followed by a (progressing) CRTC^ω_G trace starting with the formula $(RTC_{x,y}\, \mathbf{s}x = y)(0, \hat{t})$ and ending with the formula $(RTC_{x,y}\, \mathbf{s}x = y)(0, \hat{t'})$.*

Proof. The axioms of CA_G can be derived straightforwardly using axioms (i) to (v) and reasoning inductively (i.e. with cycles) over the *RTC* definition of the ordering relation $<$. For axiomatic rules the properties (1) and (2) hold trivially: there are no precursors or trace pairs since there are no premises.

For non-axiomatic rules, we build the required derivations in two stages. Firstly the (Eq) rule and standard $\mathcal{LK}$ inference rules can be applied directly to the $[\cdot]^*_X$-translations, and cuts used to introduce formulas $(RTC_{x,y}\, \mathbf{s}x = y)(0, \hat{t})$ and $t = \hat{t}$ in the antecedents of the premises for any new terms t that appear there. The (Ind) rule may be derived using an instance of Rule (4) that unfolds the *RTC*-formula in the translation of $0 < t$. Property (1) immediately holds, since we can trace occurrences of formulas $(RTC_{x,y}\, \mathbf{s}x = y)(0, \hat{t})$ in the conclusion to their ancestors in the premises.

Secondly, to show that property (2) holds, for each premise we construct a further derivation containing progressing CRTC^ω_G traces simulating each CA_G progressing trace pair. That is, for each pair of terms t and t' with $t' < t$ occurring in the premise, the subderivation contains progressing traces between formulas $(RTC_{x,y}\, \mathbf{s}x = y)(0, \hat{t})$ and $(RTC_{x,y}\, \mathbf{s}x = y)(0, \hat{t'})$. The schema for these subderivations is shown in Figs. 4 and 5, where (for a term u) the notation $\mathsf{N}\, u$ abbreviates the

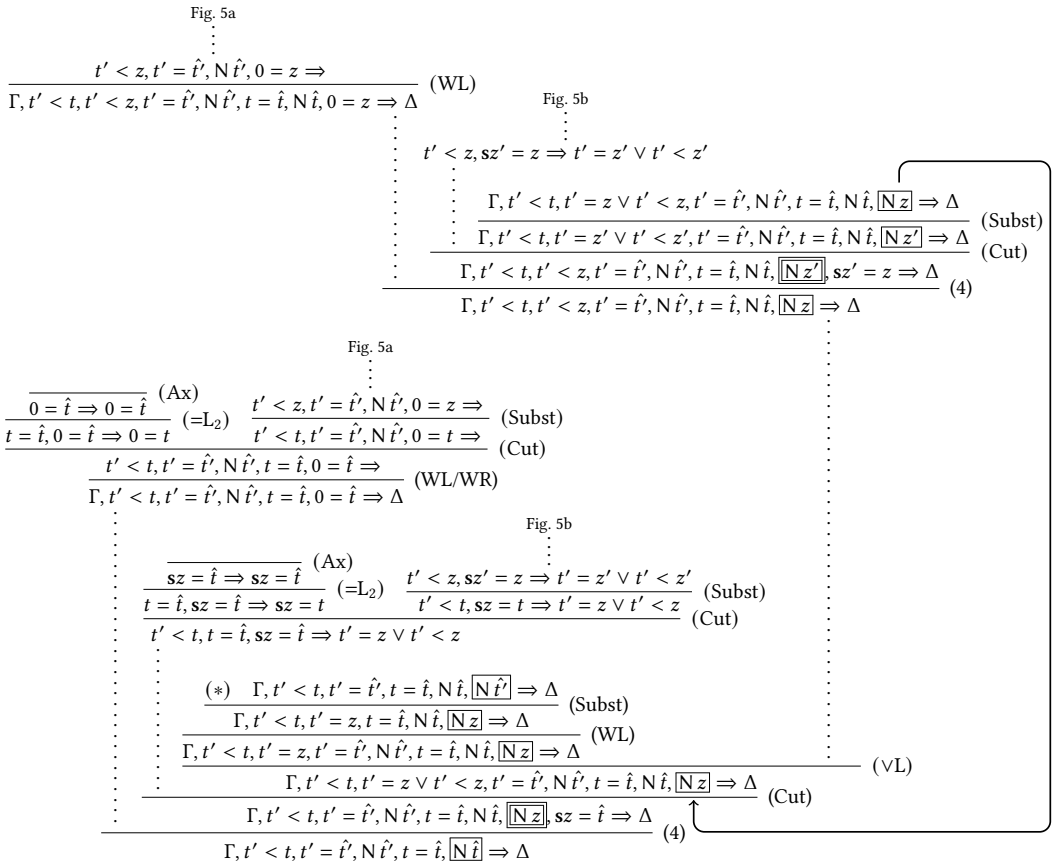


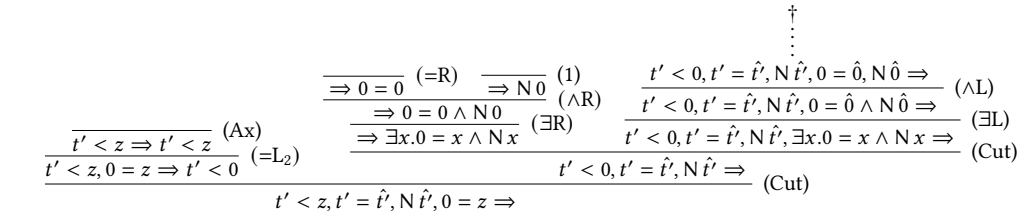
Fig. 4. A derivation schema simulating a CA_G trace progression point in CRTC^{ω}_G+A.

RTC-formula $(RTC_{x,y}\, \mathbf{s}x = y)(0, u)$ and the symbols †, ‡ and ✠ (in Fig. 5) denote straightforward CRTC^{ω}_G proofs of the $*$-translations of the appropriate axioms. Fig. 4 presents the core of the derivation, and Fig. 5 presents auxiliary subderivations. The CA_G progression point from t to t' is simulated by the CRTC^{ω}_G traces consisting of the boxed formulas in Fig. 4, following paths from the conclusion to the premise (marked with $*$). These traces progress at the point indicated by the doubly boxed formula. Thus, in the resulting CRTC^{ω}_G derivation, each CA_G trace pair (consisting of terms) is simulated by CRTC^{ω}_G traces (consisting of formulas). □

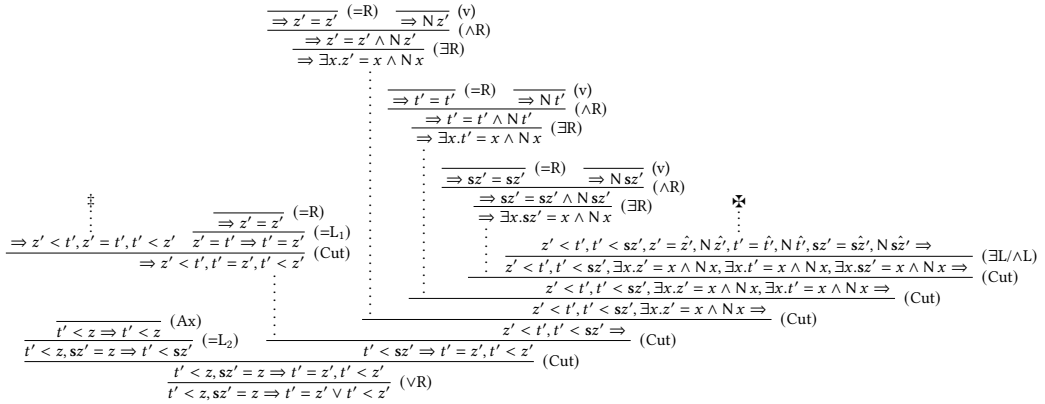
This allows us to construct a global translation of CA_G proofs into CRTC^{ω}_G+A proofs.

Lemma 6.8. *If* $\vdash_{\mathsf{CA}_G} \Gamma \Rightarrow \Delta$ *then* $\vdash_{\mathsf{CRTC}^{\omega}_G+\mathsf{A}} \Gamma \Rightarrow \Delta$.

Proof. Take a CA_G proof $\mathcal{P}$ of $\Gamma \Rightarrow \Delta$. Let X be the set of variables occurring free in the sequents appearing in $\mathcal{P}$. By Lemma 6.7(2), it follows that we can transform a CA_G pre-proof, via the $[\cdot]^*_X$-translation of each rule, into a CRTC^{ω}_G+A pre-proof with the same global structure. It remains to show that each such CRTC^{ω}_G+A pre-proof resulting from a CA_G proof is also a CRTC^{ω}_G+A proof. That is, it satisfies the CRTC^{ω}_G global trace condition. Consider an arbitrary infinite path in the CRTC^{ω}_G+A pre-proof. There are two cases to consider:



(a) Subderivation leading to $\mathsf{CRTC}^{\omega}_{G}$+A simulation of axiom $t' < 0 \Rightarrow$.



(b) Subderivation of key case-split in $\mathsf{CRTC}^{\omega}_{G}$+A simulation of CA_G progression point.

Fig. 5. Subcomponents of the derivation schema simulating a CA_G trace progression point in $\mathsf{CRTC}^{\omega}_{G}$+A.

- The infinite path ends up traversing an infinite path local to the $[\cdot]^*_X$-translation of an inference rule or CA_G axiom; in this case notice that each such infinite path has an infinitely progressing trace.
- The infinite path corresponds to an infinite path in the CA_G proof (possibly interspersed with finite traversals of the cycles local to the $[\cdot]^*_X$-translation of each rule instance). Since there is an infinitely progressing trace following the path in the CA_G proof, by the properties above there is also a corresponding infinitely progressing trace following the path in the $\mathsf{CRTC}^{\omega}_{G}$+A pre-proof. □

The following result also entails that β-translation of a sequent is provable in CA_G only if the sequent is provable in $\mathsf{CRTC}^{\omega}_{G}$.

Proposition 6.9. $\vdash_{\mathsf{CRTC}^{\omega}_{G}\text{+A}} \varphi \Leftrightarrow \varphi^{\beta}$.

Proof. This follows from Theorem 6.5(1) and Theorem 6.2. □

Lemma 6.10. *If* $\vdash_{\mathsf{CA}_G} \Gamma^{\beta} \Rightarrow \Delta^{\beta}$ *then* $\vdash_{\mathsf{CRTC}^{\omega}_{G}\text{+A}} \Gamma \Rightarrow \Delta$.

Proof. We first use Lemma 6.8 to derive $\Gamma^{\beta} \Rightarrow \Delta^{\beta}$ in $\mathsf{CRTC}^{\omega}_{G}$+A, and then combine this with derivations in $\mathsf{CRTC}^{\omega}_{G}$+A of $\varphi \Rightarrow \varphi^{\beta}$ (resp. $\varphi^{\beta} \Rightarrow \varphi$) for each $\varphi \in \Gamma$ (resp. $\varphi \in \Delta$), which exist by Proposition 6.9, with applications of cuts to derive $\Gamma \Rightarrow \Delta$. □

Inclusion of CRTC_G^ω+A in CA_G. We next show that from a CRTC_G^ω+A proof one can construct an analogous proof in CA_G which preserves cycles. Our construction introduces free variables that we are then able to trace in the cyclic CA_G proof. This is similar to the use of 'stage' variables to show the equivalence of the explicit and cyclic systems for LKID in [7]. There, LKID predicates $P(\vec{t})$ are translated into predicates $P'(\vec{t}, n)$ with equivalent inductive definitions and an extra parameter n comprising a stage variable. The equivalence is then derived by using the cycles in a proof to construct an explicit induction hypothesis over these stage variables.

As for the proof of Lemma 6.8 above, we define a local translation on proof rules that preserves CRTC_G^ω+A traces as CA_G traces. For this, we use a *parameterised* variant $\bar{\beta}[n]$ of the β-translation, which introduces its parameter as a free variable in the translation.

Definition 6.11 (Parameterised β-translation). The variant $\bar{\beta}$ of the β-translation takes a variable n as an additional parameter and is similarly defined inductively. For atomic formulas $\varphi_n^{\bar{\beta}} = \varphi$, the translation is homomorphic with respect to the first-order logical connectives, and for *RTC*-formulas $(RTC_{x,y}\,\varphi)(s,t)_n^{\bar{\beta}}$ is defined as follows:

$$s = t \vee \exists z, c \,.\, n = \mathbf{s}z \wedge B(c, 0, s) \wedge B(c, \mathbf{s}z, t) \wedge \forall u \leq z \,.\, \exists v, w \,.\, B(c, u, v) \wedge B(c, \mathbf{s}u, w) \wedge \varphi^{\beta}\left\{\frac{v}{x}, \frac{w}{y}\right\}$$

Note the use of the original β-translation for the body of the *RTC* formula φ. We define a translation $\hat{\beta}$ on sets of formulas, sequents, and inference rules as follows.

(i) For a set of formulas Γ, we define $\Gamma^{\hat{\beta}}$ to be the set of $\bar{\beta}$-translations of the formulas in Γ such that each translation of an *RTC* (sub-)formula uses a fresh (i.e. distinct) variable z for the parameter. If φ is a (sub-)formula in Γ, then we write $\varphi_z^{\bar{\beta}}$ to indicate that the variable z was used in the $\bar{\beta}$-translation of φ to produce $\Gamma^{\hat{\beta}}$.

(ii) For sequents, we define $(\Gamma \Rightarrow \Delta)^{\hat{\beta}}$ to be $\Gamma^{\hat{\beta}} \Rightarrow \Delta^{\beta}$ such that the variable parameters used by the $\hat{\beta}$-translation of the antecedent Γ are distinct from the free variables in the succedent Δ.

(iii) For an inference rule with premises $\Gamma_1 \Rightarrow \Delta_1, \ldots, \Gamma_n \Rightarrow \Delta_n$ and conclusion $\Gamma \Rightarrow \Delta$, we define its $\hat{\beta}$-translation as the inference rule with premises $(\Gamma_1 \Rightarrow \Delta_1)^{\hat{\beta}}, \ldots, (\Gamma_n \Rightarrow \Delta_n)^{\hat{\beta}}$ and conclusion $(\Gamma \Rightarrow \Delta)^{\hat{\beta}}$ such that *RTC* (sub-)formulas in the premises are $\bar{\beta}$-translated using the same variable parameter as the $\bar{\beta}$-translations of their immediate descendants in conclusion.

Remark 1. Note that for any sequence of formulas Σ, we can straightforwardly derive $\Gamma, \Sigma^{\beta} \Rightarrow \Delta$ from $\Gamma, \Sigma^{\hat{\beta}} \Rightarrow \Delta$ by first introducing existential quantifiers for the variable parameters in $\Sigma^{\hat{\beta}}$ and then eliminating the terms $n = \mathbf{s}z$ with cuts. We abbreviate such a derivation using the label $(\hat{\beta})$.

We now show that the $\hat{\beta}$-translation of each CRTC_G^ω+A inference rule can be derived in CA_G in such a way that there is a (progressing) CA_G trace simulating each (progressing) CRTC_G^ω+A trace pair present in the original rule. This is done by tracing the variables introduced as parameters in the $\hat{\beta}$-translation of the rule.

Lemma 6.12. *Let (r) be an instance of a CRTC_G^ω+A axiom or inference rule, with conclusion S and premises $S_1, \ldots, S_n$. Then the inference rule $(r)^{\hat{\beta}}$ is derivable in CA_G; moreover, if (τ, τ') is a (progressing) CRTC_G^ω trace pair for (S, S_i), with $\tau_n^{\bar{\beta}}$ in S and $(\tau')_m^{\bar{\beta}}$ in S_i, then there is a (progressing) CA_G trace following the path in the derived rule from the conclusion to the premise corresponding to S_i that starts with n in the conclusion and finishes with m in the premise.*

Proof. We here show the construction for Rule (4); the other rules are straightforward, and do not contain progressing CRTC_G^ω traces. Take an instance of Rule (4) with contexts Γ and Δ,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\varphi^\beta\left\{\frac{s}{x},\frac{t}{y}\right\} \Rightarrow \varphi^\beta\left\{\frac{s}{x},\frac{t}{y}\right\}}\,(\text{Ax})}{s = v, t = w, \varphi^\beta\left\{\frac{v}{x},\frac{w}{y}\right\} \Rightarrow \varphi^\beta\left\{\frac{s}{x},\frac{t}{y}\right\}}\,(=\text{L})}{B(c,0,s), B(c,\mathbf{s}0,t), B(c,0,v), B(c,\mathbf{s}0,w), \varphi^\beta\left\{\frac{v}{x},\frac{w}{y}\right\} \Rightarrow \varphi^\beta\left\{\frac{s}{x},\frac{t}{y}\right\}}\,(B)}{B(c,0,s), B(c,\mathbf{s}0,t), \forall u \le 0\,.\,\vartheta(u) \Rightarrow \varphi^\beta\left\{\frac{s}{x},\frac{t}{y}\right\}}\,(\le_0/\exists\text{L})}{z = 0, \Sigma(z) \Rightarrow \varphi^\beta\left\{\frac{s}{x},\frac{t}{y}\right\}}\,(\text{WL}/=\text{L})$$

$$\dfrac{\dfrac{\dfrac{}{z = 0, \Sigma(z) \Rightarrow s = s}\,(=\text{R})}{z = 0, \Sigma(z) \Rightarrow s = s \vee A(0,s,s)}\,(\vee\text{R})}{\vdots}$$

$$\dfrac{\dfrac{\dfrac{}{z = 0, \Sigma(z) \Rightarrow 0 < \mathbf{s}0}\,(\text{PA-Ax})}{z = 0, \Sigma(z) \Rightarrow 0 < n}\,(=\text{L})}{\vdots}$$

$$\dfrac{\vdots \qquad \vdots}{z = 0, \Sigma(z) \Rightarrow \exists z, m\,.\,m < n \wedge (s = z \vee A(m,s,z)) \wedge \varphi^\beta\left\{\frac{z}{x},\frac{t}{y}\right\}}\,(\exists\text{R}/\wedge\text{R})$$

(a) One step from s to t.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Pi(z') \Rightarrow \mathbf{s}z' = \mathbf{s}z'}\,(=\text{R}) \quad \dfrac{}{\Pi(z') \Rightarrow B(c,0,s)}\,(\text{Ax}) \quad \dfrac{}{\Pi(z') \Rightarrow B(c,\mathbf{s}z',v)}\,(\text{Ax}) \quad \dfrac{}{\Pi(z') \Rightarrow \forall u \le z'\,.\,\vartheta(u)}\,(\text{Ax})}{\Pi(z') \Rightarrow \exists z, c\,.\,\mathbf{s}z' = \mathbf{s}z \wedge B(c,0,s) \wedge B(c,\mathbf{s}z,v) \wedge \forall u \le z\,.\,\vartheta(u)}\,(\exists\text{R}/\wedge\text{R})}{\Pi(z') \Rightarrow s = v \vee A(\mathbf{s}z',s,v)}\,(\vee\text{R})}{\vdots}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Pi(z') \Rightarrow \mathbf{s}z' < \mathbf{s}\mathbf{s}z'}\,(\text{PA-Ax})}{\Pi(z') \Rightarrow \mathbf{s}z' < n}\,(=\text{L})}{\vdots} \qquad \dfrac{\dfrac{\dfrac{}{\varphi^\beta\left\{\frac{v}{x},\frac{t}{y}\right\} \Rightarrow \varphi^\beta\left\{\frac{v}{x},\frac{t}{y}\right\}}\,(\text{Ax})}{t = w, \varphi^\beta\left\{\frac{v}{x},\frac{w}{y}\right\} \Rightarrow \varphi^\beta\left\{\frac{v}{x},\frac{t}{y}\right\}}\,(=\text{L})}{\Pi(z') \Rightarrow \varphi^\beta\left\{\frac{v}{x},\frac{t}{y}\right\}}\,(B/WL)}{\Pi(z') \Rightarrow \exists z, m\,.\,m < n \wedge (s = z \vee A(m,s,z)) \wedge \varphi^\beta\left\{\frac{z}{x},\frac{t}{y}\right\}}\,(\exists\text{R}/\wedge\text{R})$$

$$\dfrac{\dfrac{\dfrac{n = \mathbf{s}\mathbf{s}z', B(c,0,s), B(c,\mathbf{s}\mathbf{s}z',t), \vartheta(\mathbf{s}z'), \forall u \le z'\,.\,\vartheta(u) \Rightarrow \psi}{n = \mathbf{s}\mathbf{s}z', B(c,0,s), B(c,\mathbf{s}\mathbf{s}z',t), \forall u \le \mathbf{s}z'\,.\,\vartheta(u) \Rightarrow \psi}\,(\le_\mathbf{s})}{\exists z'\,.\,z = \mathbf{s}z', \Sigma(z) \Rightarrow \psi}\,(\exists\text{L}/=\text{L})}{}$$

(b) Multi-step from s to t.

$$\dfrac{\dfrac{\dfrac{}{\Rightarrow z = 0 \vee \exists z'\,.\,z = \mathbf{s}z'}\,(\dagger) \qquad \dfrac{\dfrac{\vdots}{z = 0, \Sigma(z) \Rightarrow \psi}\,(a) \quad \dfrac{\vdots}{\exists z'\,.\,z = \mathbf{s}z', \Sigma(z) \Rightarrow \psi}\,(b)}{z = 0 \vee \exists z'\,.\,z = \mathbf{s}z', \Sigma(z) \Rightarrow \psi}\,(\vee\text{L})}{\Sigma(z) \Rightarrow \exists z, m\,.\,m < n \wedge (s = z \vee A(m,s,z)) \wedge \varphi^\beta\left\{\frac{z}{x},\frac{t}{y}\right\}}\,(\text{Cut})}{A(n,s,t) \Rightarrow \exists z, m\,.\,m < n \wedge (s = z \vee A(m,s,z)) \wedge \varphi^\beta\left\{\frac{z}{x},\frac{t}{y}\right\}}\,(\exists\text{L})$$

Fig. 6. The core subderivation of the simulation of Rule (4) in CA_G.

and principal formula $(RTC_{x,y}\,\varphi)(s,t)$ with immediate ancestor $(RTC_{x,y}\,\varphi)(s,z)$ in the right-hand premise. For terms r, s, and t, let $\vartheta(r)$ abbreviate $\exists v, w\,.\,B(c,r,v) \wedge B(c,\mathbf{s}r,w) \wedge \varphi^\beta\left\{\frac{v}{x},\frac{w}{y}\right\}$, and $A(r,s,t)$ abbreviate $\exists z, c\,.\,r = \mathbf{s}z \wedge B(c,0,s) \wedge B(c,\mathbf{s}z,t) \wedge \forall u \le z\,.\,\vartheta(u)$. Additionally, we define the following two abbreviations for sets of formulas:

$$\Sigma(r) = \{n = \mathbf{s}r, B(c,0,s), B(c,\mathbf{s}r,t), \forall u \le r\,.\,\vartheta(u)\}$$
$$\Pi(r) = \{n = \mathbf{s}\mathbf{s}r, B(c,0,s), B(c,\mathbf{s}\mathbf{s}r,t), B(c,\mathbf{s}r,v), B(c,\mathbf{s}\mathbf{s}r,w), \varphi^\beta\left\{\frac{v}{x},\frac{w}{y}\right\}, \forall u \le r\,.\,\vartheta(u)\}$$

Moreover, note the following.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A(n,s,t) \Rightarrow \exists z, m \,.\, m < n \wedge (s = z \vee A(m,s,z)) \wedge \varphi^{\beta}\left\{\frac{z}{x}, \frac{t}{y}\right\}}\ \text{Fig. 6}}{\Gamma^{\hat{\beta}}, (s = z \vee A(\boxed{m}, s, z)), \varphi^{\hat{\beta}}\left\{\frac{z}{x}, \frac{t}{y}\right\} \Rightarrow \Delta^{\beta}}\ (\hat{\beta})}{\Gamma^{\hat{\beta}}, (s = z \vee A(\boxed{m}, s, z)), \varphi^{\beta}\left\{\frac{z}{x}, \frac{t}{y}\right\} \Rightarrow \Delta^{\beta}}\ (\text{WL})}{\Gamma^{\hat{\beta}}, \boxed{m} < n, (s = z \vee A(m,s,z)), \varphi^{\beta}\left\{\frac{z}{x}, \frac{t}{y}\right\} \Rightarrow \Delta^{\beta}}\ (\exists\text{L}/\wedge\text{L})}{\Gamma^{\hat{\beta}}, \exists z, m \,.\, m < \boxed{n} \wedge (s = z \vee A(m,s,z)) \wedge \varphi^{\beta}\left\{\frac{z}{x}, \frac{t}{y}\right\} \Rightarrow \Delta^{\beta}}\ (\text{Cut})}{\Gamma^{\hat{\beta}}, A(\boxed{n}, s, t) \Rightarrow \Delta^{\beta}}$$

$$\dfrac{\dfrac{(\Gamma')^{\hat{\beta}}\left\{\frac{s}{u}, \frac{t}{w}\right\} \Rightarrow (\Delta')^{\beta}\left\{\frac{s}{u}, \frac{t}{w}\right\}}{\Gamma^{\hat{\beta}}, s = t \Rightarrow \Delta^{\beta}}\ (=\text{L})}{\Gamma^{\hat{\beta}}, s = t \vee A(\boxed{n}, s, t) \Rightarrow \Delta^{\beta}}\ (\vee\text{L})$$

Fig. 7. A derivation schema simulating Rule (4) in CA_G.

i) We can easily derive $\Rightarrow z = 0 \vee \exists z' \,.\, z = \mathbf{s}z'$ using standard first-order rules and the axioms of CA_G; we refer to this derivation using †.

ii) Technically, $\forall u \leq t \,.\, \gamma$ abbreviates the CA_G formula $\forall u \,.\, (u = t \vee u < t) \rightarrow \gamma$, and so we may straightforwardly derive both $\forall u \leq 0 \,.\, \gamma(u) \Rightarrow \gamma(0)$ and $\forall u \leq \mathbf{s}t \,.\, \gamma(u) \Rightarrow \gamma(\mathbf{s}t) \wedge \forall u \leq t \,.\, \gamma(u)$; for brevity, we refer to an instance of the (Cut) rule that applies these sequents using the labels $(\leq_0)$ and $(\leq_{\mathbf{s}})$, respectively.

iii) Recall that, since the formula B captures a β-function, we may derive $B(r,s,t), B(r,s,u) \Rightarrow t = u$; we abbreviate using the label (B) instances of (Cut) that apply an instance of this sequent.

Using these elements, Fig. 6 shows a derivation (in which we have abbreviated the consequent formula by ψ) of the sequent $A(n,s,t) \Rightarrow \exists z, m \,.\, m < n \wedge (s = z \vee A(m,s,z)) \wedge \varphi^{\beta}\left\{\frac{z}{x}, \frac{t}{y}\right\}$. Then, using Fig. 6 as a subderivation, we derive the $\hat{\beta}$-translation of Rule (4) in CA_G as shown in Fig. 7. Recall that the rule labelled $(\hat{\beta})$ abbreviates a derivation as described above in Remark 1. Note that this derivation admits non-progressing traces for all the $\bar{\beta}$-translation variable parameters in Γ. However the crucial feature is that there is a CA_G trace (indicated by the boxed variables in Fig. 7) from the variable parameter n in the conclusion to m in the right-hand premise, which progresses at the sequent containing the double-boxed variable m. Also, since the context $\Gamma^{\hat{\beta}}$ is preserved along the paths in the derivation from conclusion to premises, all non-progressing traces are simulated as well. □

Lemma 6.13. *If* $\vdash_{\mathsf{CRTC}^{\omega}_{G}\text{+A}} \Gamma \Rightarrow \Delta$ *then* $\vdash_{\mathsf{CA}_G} \Gamma^{\beta} \Rightarrow \Delta^{\beta}$.

Proof. By Remark 1, it suffices to show that if $\vdash_{\mathsf{CRTC}^{\omega}_{G}\text{+A}} \Gamma \Rightarrow \Delta$ then $\vdash_{\mathsf{CA}_G} (\Gamma \Rightarrow \Delta)^{\hat{\beta}}$. Using the derivations of $\hat{\beta}$-translations of the inference rules given in Lemma 6.12, from a $\mathsf{CRTC}^{\omega}_{G}$+A pre-proof we can build a CA_G pre-proof with the same global structure. For each bud in the resulting CA_G pre-proof, we first apply an instance of the substitution rule that substitutes each variable parameter of the $\bar{\beta}$-translated *RTC*-formulas in order to match those used in its companion. Notice that this is possible, since the parameter variable is unique for the $\bar{\beta}$-translation of each *RTC* sub-formula. We can then form a cycle in the CA_G pre-proof. Since the $\mathsf{CRTC}^{\omega}_{G}$ traces for each rule are simulated by the CA_G derived rules, for each trace following a (finite or infinite) path in the $\mathsf{CRTC}^{\omega}_{G}$+A pre-proof there is a trace following the corresponding path in the CA_G pre-proof containing a progression

point for each progression point in the $\mathsf{CRTC}^{\omega}_{G}$+A trace. From this it follows that if the $\mathsf{CRTC}^{\omega}_{G}$+A pre-proof satisfies the $\mathsf{CRTC}^{\omega}_{G}$ global trace condition, then its translation satifies the CA_G global trace condition. □

Theorem 6.14. $\vdash_{\mathsf{CRTC}^{\omega}_{G}+\mathsf{A}} \Gamma \Rightarrow \Delta$ *iff* $\vdash_{\mathsf{CA}_G} \Gamma^{\beta} \Rightarrow \Delta^{\beta}$.

Proof. By Lemmas 6.10 and 6.13. □

Equivalence of RTC_G+A **and** $\mathsf{CRTC}^{\omega}_{G}$+A**.** These results allow us to show an equivalence between the finitary and cyclic systems for TC with arithmetic.

Theorem 6.15. RTC_G+A *and* $\mathsf{CRTC}^{\omega}_{G}$+A *are equivalent.*

Proof. The fact that RTC_G+A $\subseteq$ $\mathsf{CRTC}^{\omega}_{G}$+A follows immediately from Theorem 6.2. For the converse, suppose $\Gamma \Rightarrow \Delta$ is provable in $\mathsf{CRTC}^{\omega}_{G}$+A. By Lemma 6.13 we get that $\vdash_{\mathsf{CA}_G} \Gamma^{\beta} \Rightarrow \Delta^{\beta}$. Using the equivalence between CA_G and PA_G, we obtain $\vdash_{\mathsf{PA}_G} \Gamma^{\beta} \Rightarrow \Delta^{\beta}$. Then we conclude using Theorem 6.5(2). □

Note that the result above can easily be extended to show that adding the same set of additional axioms to both RTC_G+A and $\mathsf{CRTC}^{\omega}_{G}$+A results in equivalent systems. Also note that in the systems with pairs, to embed arithmetics there is no need to explicitly include addition and its axioms. Thus, by only including the signature $\{0, \mathbf{s}\}$ and the corresponding axioms for it we can obtain that $\langle\mathsf{RTC}\rangle_G$+A and $\langle\mathsf{CRTC}\rangle^{\omega}_{G}$+A are equivalent.

In [7], the equivalence result of [34] was improved to show it holds for any set of inductive predicates containing the natural number predicate N. On the one hand, our result goes beyond that of [7] as it shows the equivalence for systems with a richer notion of inductive definition, due to the expressiveness of TC. On the other hand, TC does not support restricting the set of inductive predicates, i.e. the *RTC* operator may operate on any formula in the language. To obtain a finer result which corresponds to that of [7] we need to further explore the transformations between proofs in the two systems. This is left for future work.

6.2.2 The General Case. As mentioned, the general equivalence conjecture between LKID and CLKID^{ω} was refuted in [6], by providing a concrete example of a statement which is provable in the cyclic system but not in the explicit one. The statement, involving the 2-Hydra predicate described above in Section 2.2, is the following:

$$((\forall x.\mathbf{0} \neq \mathbf{s}x) \wedge (\forall x.\forall y.\mathbf{s}x = \mathbf{s}y \rightarrow x = y)) \rightarrow \forall x.\forall y.((\mathsf{N}(x) \wedge \mathsf{N}(y)) \rightarrow \mathsf{H}(x, y))$$

where N is a predicate defined by the induction scheme for natural numbers. That is, assuming that $\mathbf{0}$ is not a successor element and that the successor function is injective, then every pair of natural numbers is related by the 2-Hydra predicate. The non-provability of this statement in LKID (i.e. using explicit induction) follows from the existence of a Henkin model in which the statement does not hold, whose construction was given in [6].

However, a careful examination of this counter-example reveals that it only refutes a strong form of the conjecture, according to which both systems are based on the same set of productions. In fact, already in [6] it is shown that if the explicit system is extended by another inductive predicate, namely one expressing the standard ordering $\leq$ over natural numbers, then the 2-Hydra statement above becomes provable. Therefore, the less strict formulation of the question, namely whether for any proof in $\mathsf{CLKID}^{\omega}_{\phi}$ there is a proof in $\mathsf{LKID}_{\phi'}$ for some $\phi' \supseteq \phi$, has not yet been resolved. Notice that in TC the equivalence question is of this weaker variety, since the *RTC* operator 'generates' all inductive definitions at once. That is, there is no *a priori* restriction on the inductive predicates one is allowed to use. Indeed, the 2-Hydra counter-example from [6] can be expressed in $\mathcal{L}_{\mathsf{RTC}}$

(cf. Section 2.2) and proved in CRTC_G^ω. However this does not produce a counter-example to the equivalence of the two TC proof systems, since it is also provable in RTC_G due to the fact that $s \leq t$ is definable via the formula $(RTC_{w,u}\ \mathbf{s}w = u)(s,t)$.

Despite our best efforts, we have not yet managed to settle this question, which appears to be harder to resolve in the TC setting. One possible approach to solving it is the semantical one, i.e. exploiting the fact that the explicit system is known to be sound w.r.t. Henkin semantics. This is what was done in [6]. Thus, to show strict inclusion one could construct an alternative statement that is provable in CRTC_G^ω whilst also demonstrating a Henkin model for TC that is not a model of the statement. However, it has become apparent through communications with one of the authors of [6] that constructing a Henkin model appears to be much more difficult for TC than for LKID, due to its rich inductive power. In particular, it is not at all clear whether the structure that underpins the LKID counter-model for 2-Hydra admits a Henkin model for TC [5]. Alternatively, to prove equivalence, one could show that CRTC_G^ω is also sound w.r.t. Henkin semantics. Here, again, proving this does not seem to be straightforward.

In our setting, there is also the question of the inclusion of CRTC_G^ω in NCRTC_G^ω, which amounts to the question of whether overlapping cycles can be eliminated. Moreover, we can ask if NCRTC_G^ω is included in RTC_G, independently of whether this also holds for CRTC_G^ω. Again, the semantic approach described above may prove fruitful in answering these questions.

7 CONCLUSIONS AND FUTURE WORK

Transitive closure logic seems to offer a congenial framework for inductive reasoning. In this paper we have enhanced its proof theory by developing a natural infinitary proof system which is cut-free complete for its standard semantics. We further explored the restriction of this framework to cyclic proofs which provides the basis for an effective system for automating inductive reasoning and subsumes its explicit proof system. In particular, we syntactically identified a subset of cyclic proofs that is Henkin-complete.

As mentioned in the introduction, as well as throughout the paper, this research was motivated by other work on systems of inductive definitions, particularly the LKID framework of [11], its infinitary counterpart LKID^ω, and its cyclic subsystem CLKID^ω. In terms of the expressive power of the underlying logic, TC (assuming pairs) subsumes the inductive machinery underlying LKID. This is because for any inductive predicate P of LKID, there is an $\mathcal{L}_{\mathsf{RTC}}$ formula ψ such that for every standard admissible structure M for $\mathcal{L}_{\mathsf{RTC}}$, P has the same interpretation as ψ under M. This is due to Theorem 3 in [2] and the fact that the interpretation of P must necessarily be a recursively enumerable set. As for the converse inclusion, for any positive $\mathcal{L}_{\mathsf{RTC}}$ formula there is a production of a corresponding LKID inductive definition. This is due to the fact that since productions can be seen as Horn clauses they can capture disjunction and conjunction. However, the *RTC* operator can also be applied on complex formulas (whereas LKID productions only consider atomic predicates). This indicates that TC might be more expressive. It was noted in [11, p. 1180] that complex formulas may be handled by stratifying the theory of LKID, similar to [27], but the issue of relative expressiveness of the resulting theory is not addressed. While we strongly believe it is the case that TC is strictly more expressive than the logic of LKID, proving so is left for future work. Also left for future research is establishing the comparative status of the corresponding formal proof systems.

It is already clear that TC logic, as a framework, diverges from existing systems for inductive reasoning (e.g. LKID) in interesting, non-trivial ways. At this point, it is still unclear whether or not the added flexibility of transitive closure logic over that of LKID is sufficient for establishing an equivalence between RTC_G and CRTC_G^ω even in the absence of arithmetic. Thus an immediate open question that remains for future work is that of the (in)equivalence of the systems in the general case, as discussed in Section 6.2. However the question of general equivalence notwithstanding, the

uniformity provided by the transitive closure operator may offer a way to better study the more subtle relationship between implicit and explicit induction. That is, it can help in the investigation of the connections between cuts required in each system, or the relative complexity of proofs that each system admits.

In addition, several other questions and directions for further study naturally arise from the work of this paper. An obvious one would be to implement our cyclic proof system in order to investigate the practicalities of using TC logic to support automated inductive reasoning. In particular, we would like to further study the applications of the logic, and the proof systems we have defined for it, in various areas of computer science such as those described in Section 2.2. Furthermore, our preliminary investigations suggest we could enable *co*inductive reasoning in a variant of the formal system. Determining how naturally such a variant would capture styles of coinductive reasoning commonly found in the literature, and the extent of its expressivity, is left for future work.

REFERENCES

[1] Bahareh Afshari and Graham E. Leigh. Cut-free Completeness for Modal Mu-calculus. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20–23, 2017*, pages 1–12, 2017. doi:10.1109/LICS.2017.8005088.

[2] Arnon Avron. Transitive Closure and the Mechanization of Mathematics. In F. D. Kamareddine, editor, *Thirty Five Years of Automating Mathematics*, volume 28 of *Applied Logic Series*, pages 149–171. Springer, Netherlands, 2003. doi:10.1007/978-94-017-0253-9_7.

[3] David Baelde, Amina Doumane, and Alexis Saurin. Infinitary Proof Theory: the Multiplicative Additive Case. In *Proceedings of the 25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 – September 1, 2016, Marseille, France*, pages 42:1–42:17, 2016. doi:10.4230/LIPIcs.CSL.2016.42.

[4] Johan van Benthem. *The Logic of Time.* Synthese Library. Springer Netherlands, 2nd edition, 2012.

[5] Stefano Berardi. Personal Communication (10th April 2018).

[6] Stefano Berardi and Makoto Tatsuta. Classical System of Martin-Löf's Inductive Definitions Is Not Equivalent to Cyclic Proof System. In *Proceedings of the 20th International Conference on Foundations of Software Science and Computation Structures, FOSSACS 2017, Uppsala, Sweden, April 22–29, 2017*, pages 301–317, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg. doi:10.1007/978-3-662-54458-7_18.

[7] Stefano Berardi and Makoto Tatsuta. Equivalence of Inductive Definitions and Cyclic Proofs Under Arithmetic. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20–23, 2017*, pages 1–12, 2017. doi:10.1109/LICS.2017.8005114.

[8] Patrick Blackburn and Johan van Benthem. Modal Logic: A Semantic Perspective. In Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 1–84. Elsevier, 2007. doi:10.1016/S1570-2464(07)80004-8.

[9] James Brotherston. Formalised Inductive Reasoning in the Logic of Bunched Implications. In *Proceedings of Static Analysis, 14th International Symposium, SAS 2007, Kongens Lyngby, Denmark, August 22–24, 2007*, pages 87–103, 2007. doi:10.1007/978-3-540-74061-2_6.

[10] James Brotherston, Richard Bornat, and Cristiano Calcagno. Cyclic Proofs of Program Termination in Separation Logic. In *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7–12, 2008*, pages 101–112, 2008. doi:10.1145/1328438.1328453.

[11] James Brotherston and Alex Simpson. Sequent Calculi for Induction and Infinite Descent. *Journal of Logic and Computation*, 21(6):1177–1216, 2010. doi:10.1093/logcom/exq052.

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

[13] Liron Cohen. Ancestral Logic and Equivalent Systems. Master's thesis, Tel-Aviv University, Israel, 2010.

[14] Liron Cohen. Completeness for Ancestral Logic via a Computationally-Meaningful Semantics. In *Proceedings of the 26th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2017, Brasília, Brazil, September 25–28, 2017*, pages 247–260, 2017. doi:10.1007/978-3-319-66902-1_15.

[15] Liron Cohen and Arnon Avron. Ancestral Logic: A Proof Theoretical Study. In U. Kohlenbach et al., editor, *Logic, Language, Information, and Computation*, volume 8652 of *Lecture Notes in Computer Science*, pages 137–151. Springer, 2014. doi:10.1007/978-3-662-44145-9_10.

[16] Liron Cohen and Arnon Avron. The Middle Ground–Ancestral Logic. *Synthese*, pages 1–23, 2015. doi:10.1007/s11229-015-0784-3.

[17] Liron Cohen and Reuben N. S. Rowe. Uniform Inductive Reasoning in Transitive Closure Logic via Infinite Descent. In *Proceedings of the 27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4–7, 2018, Birmingham, UK*, pages 16:1–16:17, 2018. doi:10.4230/LIPIcs.CSL.2018.16.

[18] Stephen A. Cook and Robert A. Reckhow. The Relative Efficiency of Propositional Proof Systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979. doi:10.2307/2273702.

[19] Bruno Courcelle. Fundamental Properties of Infinite Trees. *Theoretical Computer Science*, 25:95–169, 1983. doi:10.1016/0304-3975(83)90059-2.

[20] Anupam Das and Damien Pous. A Cut-Free Cyclic Proof System for Kleene Algebra. In *Proceedings of the 26th International Conference Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2017, Brasília, Brazil, September 25–28, 2017*, pages 261–277, 2017. doi:10.1007/978-3-319-66902-1_16.

[21] Anupam Das and Damien Pous. Non-Wellfounded Proof Theory for (Kleene+Action)(Algebras+Lattices). In *Proceedings of the 27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4–7, 2018, Birmingham, UK*, pages 19:1–19:18, 2018. doi:10.4230/LIPIcs.CSL.2018.19.

[22] Amina Doumane. Constructive Completeness for the Linear-time μ-calculus. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20–23, 2017*, pages 1–12, 2017. doi:10.1109/LICS.2017.8005075.

[23] Gerhard Gentzen. Untersuchungen über das Logische Schließen. I. *Mathematische Zeitschrift*, 39(1):176–210, 1935. doi:10.1007/BF01201353.

[24] Leon Henkin. Completeness in the Theory of Types. *Journal of Symbolic Logic*, 15(2):81–91, 1950. doi:10.2307/2266967.

[25] Ryo Kashima and Keishi Okamoto. General Models and Completeness of First-order Modal μ-calculus. *Journal of Logic and Computation*, 18(4):497–507, 2008. doi:10.1093/logcom/exm077.

[26] Laurie Kirby and Jeff Paris. Accessible Independence Results for Peano Arithmetic. *Bulletin of the London Mathematical Society*, 14(4):285–293, 1982. doi:10.1112/blms/14.4.285.

[27] Per Martin-Löf. Hauptsatz for the Intuitionistic Theory of Iterated Inductive Definitions. In J. E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of Mathematics*, pages 179–216. Elsevier, 1971. doi:10.1016/S0049-237X(08)70847-4.

[28] Per Martin-Löf and Giovanni Sambin. *Intuitionistic Type Theory*, volume 9. Bibliopolis Napoli, 1984.

[29] Raymond McDowell and Dale Miller. Cut-elimination for a Logic with Definitions and Induction. *Theoretical Computer Science*, 232(1-2):91–119, 2000. doi:10.1016/S0304-3975(99)00171-1.

[30] Remi Nollet, Christine Tasson, and Alexis Saurin. The Complexity of Thread Criterion for Least and Greatest fixed points. In *Proceedings of the 27th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2019, London, UK, September 3–7, 2019*, 2019.

[31] Amir Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*, pages 46–57, 1977. doi:10.1109/SFCS.1977.32.

[32] Reuben N. S. Rowe and James Brotherston. Automatic Cyclic Termination Proofs for Recursive Procedures in Separation Logic. In *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2017, Paris, France, January 16–17, 2017*, pages 53–65, 2017. doi:10.1145/3018610.3018623.

[33] Luigi Santocanale. A Calculus of Circular Proofs and Its Categorical Semantics. In Mogens Nielsen and Uffe Engberg, editors, *Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures, FOSSACS 2002, Grenoble, France, April 8–12, 2002*, pages 357–371, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. doi:10.1007/3-540-45931-6_25.

[34] Alex Simpson. Cyclic Arithmetic Is Equivalent to Peano Arithmetic. In *Proceedings of the 20th International Conference on Foundations of Software Science and Computation Structures, FOSSACS 2017, Uppsala, Sweden, April 22–29, 2017*, pages 283–300, 2017. doi:10.1007/978-3-662-54458-7_17.

[35] Christoph Sprenger and Mads Dam. On the Structure of Inductive Reasoning: Circular and Tree-Shaped Proofs in the μCalculus. In Andrew D. Gordon, editor, *Proceedings of the 6th International Conference on Foundations of Software Science and Computation Structures, FOSSACS 2003, Warsaw, Poland, April 7–11, 2003*, pages 425–440, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. doi:10.1007/3-540-36576-1_27.

[36] Gaisi Takeuti. *Proof Theory*. Courier Dover Publications, 1987.

[37] Gadi Tellez and James Brotherston. Automatically Verifying Temporal Properties of Pointer Programs with Cyclic Proof. In *Proceedings of the 26th International Conference on Automated Deduction, CADE 26, Gothenburg, Sweden, August 6–11, 2017*, pages 491–508, 2017. doi:10.1007/978-3-319-63046-5_30.

[38] Alwen Tiu. *A Logical Framework For Reasoning About Logical Specifications*. PhD thesis, Penn. State University, 2004.