

Non-well-founded Proof Theory of Transitive Closure Logic

LIRON COHEN, Cornell University, USA

REUBEN N. S. ROWE, University of Kent, UK

Supporting inductive reasoning is an essential component in 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 (April 2020), 30 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

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 finitary¹ 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 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,

¹We here mean finitary as opposed to transfinite.

50 while other extensions of first-order logic with inductive definitions are a priori parametrized by
51 a set of inductive definitions (see, e.g., [11, 27, 29, 38]), bespoke induction principles do not need
52 to be added to, or embedded within, transitive closure logic; instead, all induction schemes are
53 available within a single, unified language. In this respect, the transitive closure operator resembles
54 the W-type [28], which also provides a single type constructor from which one can uniformly
55 define a variety of inductive types. This conciseness of the logic makes it of particular interest from
56 an automation point of view. The use of only one constructor of course comes with a price: namely,
57 formalizations (mostly of non-linear induction schemes) may be somewhat complex. However, they
58 generally do not require as complex an encoding as in arithmetics, since the TC operator can be
59 applied on any formula and thus (depending on the underlying signature) more naturally encode
60 induction on sets more complex than the natural numbers.

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

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

86 These infinitary proof theories generally subsume systems of explicit induction in expressive
87 power, but also offer a number of advantages. Most notably, they can ameliorate the primary
88 challenge for inductive reasoning: finding an induction *invariant*. In explicit induction systems,
89 this must be provided *a priori*, and is often much stronger than the goal one is ultimately interested
90 in proving. However, in implicit systems the inductive arguments and hypotheses may be encoded
91 in the cycles of a proof, so cyclic proof systems seem better for automation. The cyclic approach
92 has also been used to provide an optimal cut-free complete proof system for Kleene algebra [20],
93 providing further evidence of its utility for automation.

94 In the setting of TC logic, we observe some further benefits over more traditional formal systems
95 of inductive definitions and their infinitary proof theories (cf. LKID [11, 27]). TC (with a pairing
96 function) has all first-order definable finitary inductive definitions immediately 'available' within
97 the language of the logic: as with inductive hypotheses, one does not need to 'know' in advance
98

99 which induction schemes will be required. Moreover, the use of a single transitive closure operator
 100 provides a uniform treatment of all induction schemes. That is, instead of having a proof system
 101 parameterized by a set of inductive predicates and rules for them (as is the case in LKID), TC offers
 102 a single proof system with a single rule scheme for induction. This has immediate advantages
 103 for developing the metatheory: the proofs of completeness for standard semantics and adequacy
 104 (i.e. subsumption of explicit induction) for the infinitary system presented in this paper are simpler
 105 and more straightforward. Moreover, it permits a further refinement of the cyclic system, which also
 106 subsumes explicit induction, to be defined via a simple structural criterion that we call *normality*.
 107 This restriction in the search space of possible proofs further enhances the potential for automation.
 108 TC logic seems more expressive in other ways, too. For instance, the transitive closure operator
 109 may be applied to arbitrarily complex formulas, and thus inductive definitions are not restricted to
 110 consist only of conjunctions of atomic formulas (i.e. Horn clauses) as in e.g. [9, 11]. Conversely,
 111 since the TC operator is a particular instance of a least fixed point operator, it is itself subsumed by
 112 fixed-point logics such as the μ -calculus [25].

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

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

132 2 TRANSITIVE CLOSURE LOGIC AND ITS EXPRESSIVENESS

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

137 For simplicity of presentation we assume (as is standard practice) a designated equality symbol
 138 in the language. We denote by $v[x_1 := a_n, \dots, x_n := a_n]$ the variant of the assignment v which
 139 assigns a_i to x_i for each i , and by $\varphi\{\frac{t_i}{x_i}, \dots, \frac{t_n}{x_n}\}$ the result of simultaneously substituting each t_i for
 140 the free occurrences of x_i in φ . Note also that we use an operator denoting the reflexive transitive
 141 closure; however the reflexive and non-reflexive forms are equivalent in the presence of equality.

143 2.1 The Syntax and Semantics

144 *Definition 2.1 (The language \mathcal{L}_{RTC}).* Let σ be a first-order signature with equality, whose terms
 145 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
 146

variables. The language \mathcal{L}_{RTC} consists of the formulas defined by the grammar:

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

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 $(\text{RTC}_{x,y} \varphi)(s, t)$ 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 $v(s) = v(t)$;
- $M, v \models P(t_1, \dots, t_n)$ if $(v(t_1), \dots, v(t_n)) \in I(P)$;
- $M, v \models \neg\varphi$ 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 $a \in D$;
- $M, v \models (\text{RTC}_{x,y} \varphi)(s, t)$ if $v(s) = v(t)$, or there exist $a_0, \dots, a_n \in D$ ($n > 0$) 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}_{RTC} formulas are interpreted in frames as in Definition 2.2 above, except for:

- $M, v \models (\text{RTC}_{x,y} \varphi)(s, t)$ 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.

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

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

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

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

$$210 \quad \mathbf{H}(t_1, t_2) \equiv \exists n_1, n_2 . (RTC_{x,y} \varphi_1 \vee \varphi_2 \vee \varphi_3)(\langle n_1, n_2 \rangle, \langle t_1, t_2 \rangle)$$

$$211 \quad \wedge ((n_1 = \mathbf{0} \wedge n_2 = \mathbf{0}) \vee (n_1 = \mathbf{s}(\mathbf{0}) \wedge n_2 = \mathbf{0}) \vee (\mathbf{N}(n_1) \wedge n_2 = \mathbf{s}(\mathbf{0})))$$

212 where the formulas φ_1 – φ_3 are defined as follows:

$$214 \quad \varphi_1 \equiv \exists z_1 . \exists z_2 . x = \langle z_1, z_2 \rangle \wedge y = \langle \mathbf{s}(z_1), \mathbf{s}(z_2) \rangle$$

$$215 \quad \varphi_2 \equiv \exists z . x = \langle z, \mathbf{s}(z) \rangle \wedge y = \langle \mathbf{0}, \mathbf{s}(z) \rangle$$

$$216 \quad \varphi_3 \equiv \exists z . x = \langle z, \mathbf{s}(z) \rangle \wedge y = \langle \mathbf{s}(z), \mathbf{0} \rangle$$

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

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

- 226 • extending the arity of each predicate symbol to capture how its interpretation changes over
 227 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);
- 228 • introducing a fresh unary function $\mathbf{next}(s)$ to denote the state immediately following s ; and
- 229 • assuming a term constant \mathbf{s}_{init} to indicate the initial state.

230 The formula $(RTC_{x,y} y = \mathbf{next}(x))(s, s')$ then expresses that state s' occurs *after* state s , which we
 231 will abbreviate as $s \leq s'$. We can then define a translation $T[s]$ of LTL formulas, with respect
 232 to a ‘state’ variable s . For atomic formulas, we define $T[s](\mathbf{q}(t_1, \dots, t_n)) = \mathbf{q}(t_1, \dots, t_n, s)$. For
 233 standard first-order logical connectives, the translation is defined straightforwardly by induction.
 234 The translation of the temporal operators \mathbf{X} (next), \mathbf{F} (finally, or eventually), \mathbf{G} (globally, or always),
 235 and \mathbf{U} (until), are as follows:

$$237 \quad T[s](\mathbf{X} \phi) = \exists s' . s' = \mathbf{next}(s) \wedge T[s'](\phi)$$

$$238 \quad T[s](\mathbf{F} \phi) = \exists s' . s \leq s' \wedge T[s'](\phi)$$

$$239 \quad T[s](\mathbf{G} \phi) = \forall s' . s \leq s' \rightarrow T[s'](\phi)$$

$$240 \quad T[s](\phi \mathbf{U} \psi) = \exists s' . s \leq s' \wedge T[s'](\psi) \wedge \forall s'' . (s \leq s'' \wedge s'' \leq s') \rightarrow T[s''](\phi)$$

241 Finally, an LTL formula ψ is interpreted as $T[\mathbf{s}_{\text{init}}](\psi)$. This is essentially equivalent to the standard
 242 translation of modal logic into first-order logic given in, e.g., [4, 8] which takes the temporal
 243

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}_{RTC}

We briefly summarise a variation of the finitary proof system for \mathcal{L}_{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}_{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 $\text{fv}(\Gamma)$ 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 $\varphi(x_1, \dots, x_n)$ to emphasise that the formula φ may contain x_1, \dots, x_n as free variables.

3.1 The Proof System RTC_G

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

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

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

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

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].

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

$$\begin{array}{l}
295 \\
296 \\
297 \\
298 \\
299 \\
300 \\
301 \\
302 \\
303 \\
304 \\
305 \\
306 \\
307 \\
308 \\
309 \\
310 \\
311 \\
312 \\
313 \\
314 \\
315 \\
316 \\
317 \\
318 \\
319 \\
320 \\
321 \\
322 \\
323 \\
324 \\
325 \\
326 \\
327 \\
328 \\
329 \\
330 \\
331 \\
332 \\
333 \\
334 \\
335 \\
336 \\
337 \\
338 \\
339 \\
340 \\
341 \\
342 \\
343
\end{array}$$

$$\begin{array}{l}
\text{(Axiom): } \frac{}{\varphi \Rightarrow \varphi} \quad \text{(WL): } \frac{\Gamma \Rightarrow \Delta}{\Gamma, \varphi \Rightarrow \Delta} \quad \text{(WR): } \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \varphi} \\
\text{(=L}_1\text{): } \frac{\Gamma \Rightarrow \varphi\{\frac{s}{x}\}, \Delta}{\Gamma, s = t \Rightarrow \varphi\{\frac{t}{x}\}, \Delta} \quad \text{(=L}_2\text{): } \frac{\Gamma \Rightarrow \varphi\{\frac{t}{x}\}, \Delta}{\Gamma, s = t \Rightarrow \varphi\{\frac{s}{x}\}, \Delta} \quad \text{(=R): } \frac{}{\Rightarrow t = t} \\
\text{(Cut): } \frac{\Gamma \Rightarrow \varphi, \Delta \quad \Sigma, \varphi \Rightarrow \Pi}{\Gamma, \Sigma \Rightarrow \Delta, \Pi} \quad \text{(Subst): } \frac{\Gamma \Rightarrow \Delta}{\Gamma\{\frac{t_1}{x_1}, \dots, \frac{t_n}{x_n}\} \Rightarrow \Delta\{\frac{t_1}{x_1}, \dots, \frac{t_n}{x_n}\}} \\
\text{(}\forall\text{L): } \frac{\Gamma, \varphi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \varphi \vee \psi \Rightarrow \Delta} \quad \text{(}\wedge\text{L): } \frac{\Gamma, \varphi, \psi \Rightarrow \Delta}{\Gamma, \varphi \wedge \psi \Rightarrow \Delta} \quad \text{(}\rightarrow\text{L): } \frac{\Gamma \Rightarrow \varphi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \varphi \rightarrow \psi \Rightarrow \Delta} \\
\text{(}\forall\text{R): } \frac{\Gamma \Rightarrow \varphi, \psi, \Delta}{\Gamma \Rightarrow \varphi \vee \psi, \Delta} \quad \text{(}\wedge\text{R): } \frac{\Gamma \Rightarrow \varphi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \varphi \wedge \psi, \Delta} \quad \text{(}\rightarrow\text{R): } \frac{\Gamma, \varphi \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \varphi \rightarrow \psi, \Delta} \\
\text{(}\exists\text{L): } \frac{\Gamma, \varphi \Rightarrow \Delta}{\Gamma, \exists x. \varphi \Rightarrow \Delta} \quad x \notin \text{fv}(\Gamma, \Delta) \quad \text{(}\forall\text{L): } \frac{\Gamma, \varphi\{\frac{t}{x}\} \Rightarrow \Delta}{\Gamma, \forall x. \varphi \Rightarrow \Delta} \quad \text{(}\neg\text{L): } \frac{\Gamma \Rightarrow \varphi, \Delta}{\Gamma, \neg \varphi \Rightarrow \Delta} \\
\text{(}\exists\text{R): } \frac{\Gamma \Rightarrow \varphi\{\frac{t}{x}\}, \Delta}{\Gamma \Rightarrow \exists x. \varphi, \Delta} \quad \text{(}\forall\text{R): } \frac{\Gamma \Rightarrow \varphi, \Delta}{\Gamma \Rightarrow \forall x. \varphi, \Delta} \quad x \notin \text{fv}(\Gamma, \Delta) \quad \text{(}\neg\text{R): } \frac{\Gamma, \varphi \Rightarrow \Delta}{\Gamma \Rightarrow \neg \varphi, \Delta}
\end{array}$$

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

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:

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

where $x \notin \text{fv}(\Gamma, \Delta)$ and $y \notin \text{fv}(\Gamma, \Delta, \psi)$. 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}_{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}_{RTC} is defined like RTC_G , but replacing Rule (3) by the following case-split rule:

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

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 $s_0, s_1, \dots, (s_n)$ 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 $\tau_1, \tau_2, \dots, (\tau_n)$ follows a path $s_1, s_2, \dots, (s_m)$ 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 $\phi \equiv (\text{RTC}_{x,y} \varphi)(s, t)$, 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, \dots, 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 $M, v \models (\text{RTC}_{x,y} \varphi)(s, s)$ for all M and v .

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

- For Rule (4), since $M, v \models (\text{RTC}_{x,y} \varphi)(s, t)$ 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 $a_0, \dots, a_n \in D$ ($n > 0$) 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, \dots, a_n of minimal length, and thus surmise $\delta_{(\text{RTC}_{x,y} \varphi)(s,t)}(M, v) = n$. Since z is fresh, it follows that $M', v' \models \varphi\{\frac{z}{x}, \frac{t}{y}\}$ 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 $M, v' \models (\text{RTC}_{x,y} \varphi)(s, z)$; otherwise this is witnessed by the sequence a_0, \dots, a_{n-1} . Thus we also have that $\delta_{(\text{RTC}_{x,y} \varphi)(s,z)}(M', v') = n - 1$. 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' . \square

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 RTC_G^ω proof ensures soundness via a standard infinite descent-style construction.

THEOREM 4.8 (SOUNDNESS OF RTC_G^ω). *If there is a RTC_G^ω 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 RTC_G^ω 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. \square

Following a standard technique (as used in e.g. [11]), we can show cut-free completeness of RTC_G^ω 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:

(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:

³Note 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.

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

(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:

$$\frac{\Gamma, s = t \Rightarrow \Delta, \varphi\left\{\frac{t}{x}\right\}, \varphi\left\{\frac{s}{x}\right\}}{\Gamma, s = t \Rightarrow \Delta, \varphi\left\{\frac{t}{x}\right\}} \quad (=L_1) \qquad \frac{\Gamma, s = t \Rightarrow \Delta, \varphi\left\{\frac{s}{x}\right\}, \varphi\left\{\frac{t}{x}\right\}}{\Gamma, s = t \Rightarrow \Delta, \varphi\left\{\frac{s}{x}\right\}} \quad (=L_2)$$

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]. \square

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_G^ω if and only if $\Gamma_\omega \Rightarrow F\left\{\frac{u}{x}\right\}$ is cut-free provable in RTC_G^ω .*

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

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

544 ($t = u \in \Gamma_\omega$): Assume $\Gamma_\omega \Rightarrow F\{\frac{t}{x}\}$ is cut-free provable, then we can apply the ($=L_1$) rule to derive
 545 (without cut) $\Gamma_\omega, t = u \Rightarrow F\{\frac{u}{x}\}$; however notice that $\Gamma_\omega, t = u$ is simply Γ_ω since $t = u \in \Gamma_\omega$
 546 already. The converse direction is symmetric, using rule ($=L_2$).

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

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

549 ($t_1 \sim u_1 \wedge \dots \wedge t_n \sim u_n \Rightarrow f(t_1, \dots, t_n) \sim f(u_1, \dots, u_n)$): Consider the formula F ; clearly there
 550 exist formulas G_1, \dots, G_n and some variable y such that $G_i\{\frac{t_i}{y}\} \equiv F\{\frac{f(u_1, \dots, u_{i-1}, t_i, \dots, t_n)}{x}\}$ for each
 551 $i \leq n$. By induction, each sequent $\Gamma_\omega \Rightarrow G_i\{\frac{t_i}{y}\}$ is cut-free provable if and only if so too is
 552 $\Gamma_\omega \Rightarrow G_i\{\frac{u_i}{y}\}$. The result then follows since $F\{\frac{f(t_1, \dots, t_n)}{x}\} \equiv G_1\{\frac{t_1}{y}\}$ and $F\{\frac{f(u_1, \dots, u_n)}{x}\} \equiv G_n\{\frac{u_n}{y}\}$,
 553 and also $G_i\{\frac{u_i}{y}\} \equiv G_{i+1}\{\frac{t_{i+1}}{y}\}$ for each $i < n$. \square

554 We define the counter-interpretation as follows.
 555

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

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

561 We also define a valuation ρ_ω for M_ω by $\rho_\omega(x) = [x]$ for all variables x .
 562

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

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

566 PROOF. By well-founded induction using the lexicographic ordering of the number of binders
 567 (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 .
 568

569 For ψ atomic (i.e. of the form $q(t_1, \dots, t_k)$), if $\psi \in \Gamma_\omega$ then it follows immediately by Definition 4.15
 570 that $M_\omega, \rho_\omega \models q(t_1, \dots, t_k)$. If, on the other hand, $\psi \in \Delta_\omega$ then assume for contradiction that indeed
 571 $M_\omega, \rho_\omega \models q(t_1, \dots, t_k)$. It then follows from Definition 4.15 that there are terms u_1, \dots, u_k such that
 572 $q(u_1, \dots, u_k) \in \Gamma_\omega$ and $u_i \sim t_i$ for each $i \leq k$. Notice that then we can prove $\Gamma_\omega \Rightarrow q(u_1, \dots, u_k)$
 573 axiomatically, and so it follows by (k applications of) Lemma 4.14 that $\Gamma_\omega \Rightarrow q(t_1, \dots, t_k)$ is
 574 cut-free provable. However, since $q(t_1, \dots, t_k) \in \Delta_\omega$, this would mean that the limit sequent
 575 $\Gamma_\omega \Rightarrow \Delta_\omega$ is cut-free provable, which contradicts Lemma 4.12. Thus we conclude that in fact
 576 $M_\omega, \rho_\omega \not\models q(t_1, \dots, t_k)$.

577 For ψ an equality formula $s = t$, if $\psi \in \Gamma_\omega$ then we have immediately by Definition 4.15 that
 578 $\rho_\omega(s) = \rho_\omega(t)$ and thus that $M_\omega, \rho_\omega \models s = t$ by Definition 2.2. If, on the other hand, $s = t \in \Delta_\omega$,
 579 suppose for contradiction that indeed $M_\omega, \rho_\omega \models s = t$. It then follows from Definition 4.15 that
 580 $s \sim t$. Since we may derive $\Gamma_\omega \Rightarrow s = s$ axiomatically, it thus follows from Lemma 4.14 that there
 581 is a cut-free proof of $\Gamma_\omega \Rightarrow s = t$. However, since $s = t \in \Delta_\omega$ this would mean that the limit
 582 sequent $\Gamma_\omega \Rightarrow \Delta_\omega$ is cut-free provable, which contradicts Lemma 4.12. We thus conclude that in
 583 fact $M_\omega, \rho_\omega \not\models s = t$.
 584

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

587 In case $\psi = (\text{RTC}_{x,y} \varphi)(s, t)$, we reason as follows.
 588

• 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 $\varphi\{\frac{z_1}{x}, \frac{t}{y}\} \in \Gamma_\omega$. 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, \dots, z_n ($n > 0$) with $\varphi\{\frac{z_1}{x}, \frac{t}{y}\} \in \Gamma_\omega$, $\varphi\{\frac{z_{i+1}}{x}, \frac{z_i}{y}\} \in \Gamma_\omega$, 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 $M_\omega, \rho_\omega \models \varphi\{\frac{z_1}{x}, \frac{t}{y}\}$, and $M_\omega, \rho_\omega \models \varphi\{\frac{z_{i+1}}{x}, \frac{z_i}{y}\}$ 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 $a_0, \dots, a_n \in D$ ($n > 0$):

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 $\varphi\{\frac{s}{x}, \frac{t}{y}\} \in \Delta_\omega$ 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 $M_\omega, \rho_\omega \not\models \varphi\{\frac{s}{x}, \frac{t}{y}\}$ 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 $\varphi\{\frac{r}{x}, \frac{t}{y}\} \in \Delta_\omega$, 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 $M_\omega, \rho_\omega \not\models \varphi\{\frac{r}{x}, \frac{t}{y}\}$ 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 $a_0, \dots, a_n \in D$ ($n > 0$) 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. \square

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

640 THEOREM 4.17 (COMPLETENESS). RTC_G^ω is complete for standard semantics.

641
 642 PROOF. Now given any sequent S , if some search tree T_ω contracted for S is not an RTC_G^ω proof
 643 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,
 644 then T_ω is a recursive RTC_G^ω proof for it. \square
 645

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

647 COROLLARY 4.18 (CUT ADMISSIBILITY). Cut is admissible in RTC_G^ω .

649 4.3 The Proof System CRTC_G^ω

650 In general, one cannot reason effectively about infinite proofs as in RTC_G^ω . In order to do so we need
 651 to restrict our attention to those proof trees which are finitely representable. These are the *regular*
 652 infinite proof trees, which contain only finitely many *distinct* subtrees. They can be specified as
 653 systems of recursive equations or, alternatively, as *cyclic graphs* [19]. Note that a given regular
 654 infinite proof may have many different graph representations. One possible way of formalizing
 655 such proof graphs is as standard proof trees containing open nodes (called buds), to each of which
 656 is assigned a syntactically equal internal node of the proof (called a companion).
 657

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

661 *Definition 4.19 (Cyclic Proofs).* A CRTC_G^ω pre-proof is a pair $\langle P, f \rangle$, where P is a finite derivation
 662 tree formed using the inference rules of RTC_G^ω and f is a function assigning a companion to every
 663 bud node in P . The graph associated with a pre-proof is the one induced by P by identifying each
 664 bud node with its companion. A CRTC_G^ω proof is then a CRTC_G^ω pre-proof whose graph satisfies
 665 the global trace condition, i.e., every infinite path is followed by some infinitely progressing trace.
 666

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

675 Since every CRTC_G^ω proof is also a RTC_G^ω proof, soundness of CRTC_G^ω is an immediate corollary
 676 of Theorem 4.8.

677 COROLLARY 4.20 (SOUNDNESS OF CRTC_G^ω). If there is a CRTC_G^ω proof of $\Gamma \Rightarrow \Delta$, then $\Gamma \Rightarrow \Delta$ is
 678 valid (w.r.t. the standard semantics).
 679

680 Notice that the construction of Definition 4.10 does not necessarily produce CRTC_G^ω pre-proofs,
 681 and so we do not obtain a completeness result using this technique. Indeed, since one may encode
 682 arithmetic in TC and the set of CRTC_G^ω proofs is recursively enumerable, CRTC_G^ω cannot be complete
 683 w.r.t. the standard semantics. In Section 6 below we show that CRTC_G^ω subsumes RTC_G , which
 684 entails its Henkin-completeness. However, the question of whether CRTC_G^ω is *sound* w.r.t. Henkin
 685 semantics remains open.
 686

5 VARIANTS OF \mathcal{L}_{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}_{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 $\langle \cdot \rangle(x, y)$, the $2n$ -ary RTC -formula $(RTC_{x_1, x_2, y_1, y_2}^2 \varphi)(s_1, s_2, t_1, t_2)$ 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 $\langle \cdot \rangle$, the proof systems $\langle RTC \rangle_G$, $\langle RTC \rangle_G^\omega$, and $\langle CRTC \rangle_G^\omega$ are obtained from RTC_G , RTC_G^ω , $CRTC_G^\omega$ (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 $\langle RTC \rangle_G$ AND $\langle RTC \rangle_G^\omega$). *The proof systems $\langle RTC \rangle_G$ and $\langle RTC \rangle_G^\omega$ are both sound and complete for the admissible forms of Henkin and standard semantics, respectively.*

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

For the sake of simplicity of presentation we have thus far included a designated equality symbol in the language \mathcal{L}_{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 \perp , whose interpretation is defined such that $M, v \models \perp$ 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:

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

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

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

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

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

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}_{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 $(RTC_{x,y} \perp)(t, t)$, which is provable using Rule (1). In the finitary variation, Rule ($=L_1$) is derivable as follows:

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

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

⁴Note that the inclusion of \perp is itself also only a notational convenience since it may be encoded via any contradiction.

⁵Here, Γ', v, w, s and t refer to the instantiations of these same meta-variables appearing in the schema of Rule (6).

$$\begin{array}{c}
785 \\
786 \\
787 \\
788 \\
789 \\
790
\end{array}
\frac{\frac{\overline{\Gamma, \psi\{\frac{s}{x}\} \Rightarrow \Delta, \psi\{\frac{s}{x}\}} \quad (\text{Ax+WL+WR})}{\Gamma \Rightarrow \Delta, \psi\{\frac{s}{x}\}, \neg\psi\{\frac{s}{x}\}} \quad (\neg\text{R}) \quad \frac{\overline{\Gamma, \psi, \perp \Rightarrow \Delta, \psi\{\frac{t}{x}\}} \quad (\perp\text{+WL+WR})}{\Gamma, \neg\psi\{\frac{s}{x}\}, (\text{RTC}_{x,y \perp})(s, t) \Rightarrow \Delta, \neg\psi\{\frac{t}{x}\}} \quad (3)}{\Gamma, (\text{RTC}_{x,y \perp})(s, t) \Rightarrow \Delta, \psi\{\frac{s}{x}\}, \neg\psi\{\frac{t}{x}\}} \quad (\text{Cut}) \quad \frac{\Gamma \Rightarrow \Delta, \psi\{\frac{t}{x}\}}{\Gamma, \neg\psi\{\frac{t}{x}\} \Rightarrow \Delta} \quad (\neg\text{L})}{\Gamma, (\text{RTC}_{x,y \perp})(s, t) \Rightarrow \Delta, \psi\{\frac{s}{x}\}} \quad (\text{Cut})$$

791 In the infinitary variations, both Rule (=L₁) and Rule (=L₂) are simply instances of Rule (6). For
792 example, Rule (=L₁) is derivable as follows:

$$\begin{array}{c}
793 \\
794 \\
795
\end{array}
\frac{\Gamma \Rightarrow \Delta, \psi\{\frac{s}{x}\} \quad \overline{\Gamma, (\text{RTC}_{x,y \perp})(s, z), \perp \Rightarrow \Delta, \psi\{\frac{t}{x}\}} \quad (\perp\text{+WL+WR})}{\Gamma, (\text{RTC}_{x,y \perp})(s, t) \Rightarrow \Delta, \psi\{\frac{t}{x}\}} \quad (6)$$

796 where the only occurrences of t in the context of the conclusion that are swapped for s in the
797 left-hand premise are those in the formula ψ (and no occurrences of s in the conclusion are swapped
798 for t). Rule (=L₂) is derived symmetrically.

800 **5.2.3 Completeness.** The adaptation of the completeness proof for the finitary variation is
801 standard. Completeness of the infinitary system follows from the fact that, in addition to the
802 equality rules derived above, we can also derive the original form of the case-split rule as follows.

$$\begin{array}{c}
803 \\
804 \\
805 \\
806 \\
807 \\
808
\end{array}
\frac{\frac{\Gamma, (\text{RTC}_{x,y \perp})(s, t) \Rightarrow \Delta \quad \Gamma, (\text{RTC}_{x,y \perp})(s, s), (\text{RTC}_{x,y \perp})(s, z), \varphi\{\frac{z}{x}, \frac{t}{y}\} \Rightarrow \Delta}{\Gamma, (\text{RTC}_{x,y \perp})(s, t) \Rightarrow \Delta, \varphi\{\frac{z}{x}, \frac{t}{y}\} \Rightarrow \Delta} \quad (\text{WL})}{\frac{\frac{\Gamma, (\text{RTC}_{x,y \perp})(s, t) \Rightarrow \Delta \quad \Gamma, (\text{RTC}_{x,y \perp})(s, s), (\text{RTC}_{x,y \perp})(s, z), \varphi\{\frac{z}{x}, \frac{t}{y}\} \Rightarrow \Delta}{\Gamma, (\text{RTC}_{x,y \perp})(s, t) \Rightarrow \Delta, \varphi\{\frac{z}{x}, \frac{t}{y}\} \Rightarrow \Delta} \quad (6)}{\Rightarrow (\text{RTC}_{x,y \perp})(s, s)} \quad (1) \quad \frac{\Gamma, (\text{RTC}_{x,y \perp})(s, s), (\text{RTC}_{x,y \perp})(s, z), \varphi\{\frac{z}{x}, \frac{t}{y}\} \Rightarrow \Delta}{\Gamma, (\text{RTC}_{x,y \perp})(s, t) \Rightarrow \Delta} \quad (\text{Cut})}$$

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

814 6 RELATING THE FINITARY AND INFINITARY PROOF SYSTEMS

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

818 6.1 Inclusion of RTC_G in CRTC_G^ω

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

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

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

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

824 PROOF. Let \mathcal{P} be a proof in RTC_G and \mathcal{P}' be the corresponding pre-proof in CRTC_G^ω obtained
825 by replacing each instance of Rule (3) by the corresponding instance of the proof schema given in
826 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
827 the subproofs that simulate Rule (3), any infinite path in \mathcal{P}' must eventually end up traversing one
828 of these cycles infinitely often. Therefore, it suffices to show that there is an infinitely progressing
829 of these cycles infinitely often. Therefore, it suffices to show that there is an infinitely progressing
830 of these cycles infinitely often. Therefore, it suffices to show that there is an infinitely progressing
831 of these cycles infinitely often. Therefore, it suffices to show that there is an infinitely progressing
832 of these cycles infinitely often. Therefore, it suffices to show that there is an infinitely progressing
833 of these cycles infinitely often.

$$\begin{array}{c}
\frac{\Gamma, \psi\left\{\frac{v}{x}\right\}, (RTC_{x,y} \varphi)(v, w) \Rightarrow \Delta, \psi\left\{\frac{w}{x}\right\}}{\Gamma, \psi\left\{\frac{v}{x}\right\}, (RTC_{x,y} \varphi)(v, z) \Rightarrow \Delta, \psi\left\{\frac{z}{x}\right\}} \text{ (Subst)} \quad \frac{\Gamma, \psi, \varphi \Rightarrow \Delta, \psi\left\{\frac{y}{x}\right\}}{\Gamma, \psi\left\{\frac{z}{x}\right\}, \varphi\left\{\frac{z}{x}, \frac{w}{y}\right\} \Rightarrow \Delta, \psi\left\{\frac{w}{x}\right\}} \text{ (Subst)} \\
\hline
\Gamma, \psi\left\{\frac{v}{x}\right\}, (RTC_{x,y} \varphi)(v, z), \varphi\left\{\frac{z}{x}, \frac{w}{y}\right\} \Rightarrow \Delta, \psi\left\{\frac{w}{x}\right\} \text{ (Cut)} \\
\hline
\frac{\Gamma, \psi\left\{\frac{v}{x}\right\} \Rightarrow \Delta, \psi\left\{\frac{v}{x}\right\}}{\Gamma, \psi\left\{\frac{v}{x}\right\}, v = w \Rightarrow \Delta, \psi\left\{\frac{w}{x}\right\}} \text{ (=L}_1\text{)} \quad \begin{array}{c} \vdots \\ \vdots \\ \vdots \end{array} \\
\hline
\Gamma, \psi\left\{\frac{v}{x}\right\}, (RTC_{x,y} \varphi)(v, w) \Rightarrow \Delta, \psi\left\{\frac{w}{x}\right\} \text{ (4)} \\
\hline
\Gamma, \psi\left\{\frac{s}{x}\right\}, (RTC_{x,y} \varphi)(s, t) \Rightarrow \Delta, \psi\left\{\frac{t}{x}\right\} \text{ (Subst)}
\end{array}$$

Fig. 2. $CRTC_G^\omega$ 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). \square

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, $CRTC_G^\omega$ 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 being reasoned over.

This uniform syntactic translation of the explicit RTC_G induction rule into $CRTC_G^\omega$ 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 $NCRTC_G^\omega$ is the subsystem of RTC_G^ω 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 $CRTC_G^\omega$, using the construction shown in Figure 2, results in a proof with no overlapping cycles.

THEOREM 6.4. $NCRTC_G^\omega \supseteq RTC_G$.

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

6.2 Inclusions of $CRTC_G^\omega$ 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

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

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

887
 888 **6.2.1 The Case of Arithmetics.** Here we show equivalence of the systems RTC_G+A and $\text{CRTC}_G^\omega+A$
 889 for languages \mathcal{L}_{RTC} based on the signature $\{0, s, +\}$, obtained by adding to RTC_G and CRTC_G^ω ,
 890 respectively, the standard axioms of Peano arithmetic (PA) together with the *RTC* characterization
 891 of the natural numbers⁷, i.e.:

- 892 i) $sx = 0 \Rightarrow$
 893 ii) $sx = sy \Rightarrow x = y$
 894 iii) $\Rightarrow x + 0 = x$
 895 iv) $\Rightarrow x + sy = s(x + y)$
 896 v) $\Rightarrow (\text{RTC}_{w,u} sw = u)(0, x)$

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

901 Furthermore, recall that we can express facts about sequences of numbers in PA by using a
 902 β -function such that for any finite sequence k_0, k_1, \dots, k_n there is some c such that for all $i \leq n$,
 903 $\beta(c, i) = k_i$. Accordingly, let B be a well-formed formula of the language of PA with three free
 904 variables which captures in PA a β -function. The β -translation of a formula φ in \mathcal{L}_{RTC} is defined
 905 inductively. For atomic formulas $\varphi^\beta = \varphi$, the translation is homomorphic with respect to the
 906 first-order logical connectives, and for *RTC*-formulas $((\text{RTC}_{x,y} \varphi)(s, t))^\beta$ is defined as:

$$907 \quad s = t \vee \left(\exists z, c . B(c, 0, s) \wedge B(c, sz, t) \wedge \left(\forall u \leq z . \exists v, w . B(c, u, v) \wedge B(c, su, w) \wedge \varphi^\beta \left\{ \frac{v}{x}, \frac{w}{y} \right\} \right) \right).$$

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

913
 914 **THEOREM 6.5** (cf. [16]). *The following hold.*

- 915 (1) $\vdash_{\text{RTC}_G+A} \varphi \Leftrightarrow \varphi^\beta$.
 916 (2) $\vdash_{\text{RTC}_G+A} \Gamma \Rightarrow \Delta$ iff $\vdash_{\text{PA}_G} \Gamma^\beta \Rightarrow \Delta^\beta$.

917
 918 Secondly we show that, also modulo translation via the β -function, $\text{CRTC}_G^\omega+A$ is equivalent
 919 to Simpson's Gentzen-style cyclic system CA_G for arithmetic. Simpson's result [34] that CA_G is
 920 equivalent to PA_G provides the final link in the chain. This is summarised in Fig. 3 below.

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

$$926 \quad \text{(Eq): } \frac{\Gamma \left\{ \frac{u}{x}, \frac{t}{y} \right\} \Rightarrow \Delta \left\{ \frac{u}{x}, \frac{t}{y} \right\}}{\Gamma \left\{ \frac{t}{x}, \frac{u}{y} \right\}, t = u \Rightarrow \Delta \left\{ \frac{t}{x}, \frac{u}{y} \right\}} \quad \text{(Ind): } \frac{\Gamma, t = sx \Rightarrow \Delta}{\Gamma, 0 < t \Rightarrow \Delta} \quad x \text{ is fresh}$$

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

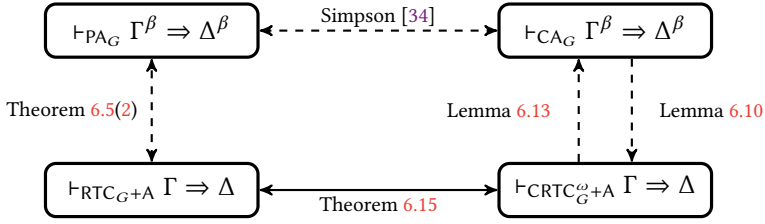


Fig. 3. The structure of the equivalence proof for RTC_{G+A} and $CRTCC_{G+A}^{\omega}$.

and the following axiom schemas:

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

Traces of CA_G consist of *terms*, rather than formulas as in $CRTCC_G^{\omega}$. 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 $t = t''\{\frac{u}{x}, \frac{v}{y}\}$ and $t' = t''\{\frac{v}{x}, \frac{u}{y}\}$ 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 $t_0, t_1, \dots, (t_n)$ follows a path $S_0, S_1, \dots, (S_m)$ 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 $CRTCC_{G+A}^{\omega}$. We give a translation of CA_G proofs into $CRTCC_{G+A}^{\omega}$ proofs. Since $CRTCC_G^{\omega}$ 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 $CRTCC_{G+A}^{\omega}$ as part of a proof of the equivalence of CA_G and $CRTCC_{G+A}^{\omega}$, 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}_{RTC} as $s \neq t \wedge (RTC_{x,y} sx = y)(s, t)$, 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 $CRTCC_G^{\omega}$ into their defining formulas in \mathcal{L}_{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 $(RTC_{x,y} sx = y)(0, \hat{t}) \in \Gamma_X^*$. 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 $CRTC_G^\omega + A$ proof of $\Gamma_X^* \Rightarrow \Delta$. Thence we may obtain a $CRTC_G^\omega + A$ proof of $\Gamma \Rightarrow \Delta$ by cutting all the added formulas $(RTC_{x,y} sx = y)(0, \hat{t})$ 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 $CRTC_G^\omega + A$ in such a way that (progressing) CA_G trace pairs are simulated by (progressing) $CRTC_G^\omega$ traces.

LEMMA 6.7. *Let S and S_1, \dots, 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 $CRTC_G^\omega + A$*

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

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 $CRTC_G^\omega$ trace starting with the formula $(RTC_{x,y} sx = y)(0, \hat{t})$ and ending with the formula $(RTC_{x,y} sx = y)(0, \hat{t}')$. Furthermore, all infinite paths in the derived inference rule are followed by infinitely progressing ($CRTC_G^\omega$) 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) $CRTC_G^\omega$ trace starting with the formula $(RTC_{x,y} sx = y)(0, \hat{t})$ and ending with the formula $(RTC_{x,y} sx = y)(0, \hat{t}')$.

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 $(RTC_{x,y} sx = y)(0, \hat{t})$ 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 $(RTC_{x,y} sx = y)(0, \hat{t})$ 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 $CRTC_G^\omega$ 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 $(RTC_{x,y} sx = y)(0, \hat{t})$ and $(RTC_{x,y} sx = y)(0, \hat{t}')$. The schema for these subderivations is shown in Figs. 4 and 5, where (for a term u) the notation Nu abbreviates the

1030 Fig. 5a
 1031 \vdots
 1032 $\frac{t' < z, t' = \hat{t}', N \hat{t}', 0 = z \Rightarrow}{\Gamma, t' < t, t' < z, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, 0 = z \Rightarrow \Delta}$ (WL)
 1033 \vdots
 1034 $t' < z, sz' = z \Rightarrow t' = z' \vee t' < z'$
 1035 \vdots
 1036 $\frac{\Gamma, t' < t, t' = z \vee t' < z, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}{\Gamma, t' < t, t' = z' \vee t' < z', t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz'} \Rightarrow \Delta}$ (Subst)
 1037 \vdots
 1038 $\frac{\Gamma, t' < t, t' < z, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz'}, sz' = z \Rightarrow \Delta}{\Gamma, t' < t, t' < z, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}$ (Cut)
 1039 \vdots
 1040 $\frac{\Gamma, t' < t, t' < z, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}{\Gamma, t' < t, t' < z, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}$ (4)
 1041 \vdots
 1042 Fig. 5a
 1043 $\frac{\frac{0 = \hat{t} \Rightarrow 0 = \hat{t}}{t = \hat{t}, 0 = \hat{t} \Rightarrow 0 = t}$ (Ax) $\frac{t' < z, t' = \hat{t}', N \hat{t}', 0 = z \Rightarrow}{t' < t, t' = \hat{t}', N \hat{t}', 0 = t \Rightarrow}$ (Subst)
 1044 $\frac{t' < t, t' = \hat{t}', N \hat{t}', t = \hat{t}, 0 = \hat{t} \Rightarrow}{\Gamma, t' < t, t' = \hat{t}', N \hat{t}', t = \hat{t}, 0 = \hat{t} \Rightarrow \Delta}$ (Cut)
 1045 \vdots
 1046 $\frac{t' < t, t' = \hat{t}', N \hat{t}', t = \hat{t}, 0 = \hat{t} \Rightarrow}{\Gamma, t' < t, t' = \hat{t}', N \hat{t}', t = \hat{t}, 0 = \hat{t} \Rightarrow \Delta}$ (WL/WR)
 1047 \vdots
 1048 Fig. 5b
 1049 $\frac{\frac{sz = \hat{t} \Rightarrow sz = \hat{t}}{t = \hat{t}, sz = \hat{t} \Rightarrow sz = t}$ (Ax) $\frac{t' < z, sz' = z \Rightarrow t' = z' \vee t' < z'}{t' < t, sz = t \Rightarrow t' = z \vee t' < z}$ (Subst)
 1050 \vdots
 1051 $\frac{t' < t, t = \hat{t}, sz = \hat{t} \Rightarrow t' = z \vee t' < z}{t' < t, t = \hat{t}, sz = \hat{t} \Rightarrow t' = z \vee t' < z}$ (Cut)
 1052 \vdots
 1053 $\frac{(*) \quad \Gamma, t' < t, t' = \hat{t}', t = \hat{t}, N \hat{t}, \boxed{N \hat{t}'} \Rightarrow \Delta}{\Gamma, t' < t, t' = z, t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}$ (Subst)
 1054 \vdots
 1055 $\frac{\Gamma, t' < t, t' = z, t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}{\Gamma, t' < t, t' = z, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}$ (WL)
 1056 \vdots
 1057 $\frac{\Gamma, t' < t, t' = z \vee t' < z, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}{\Gamma, t' < t, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}$ (vL)
 1058 \vdots
 1059 $\frac{\Gamma, t' < t, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}{\Gamma, t' < t, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}$ (Cut)
 1060 \vdots
 1061 $\frac{\Gamma, t' < t, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}{\Gamma, t' < t, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}$ (4)
 1062 \vdots
 1063 $\frac{\Gamma, t' < t, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}{\Gamma, t' < t, t' = \hat{t}', N \hat{t}', t = \hat{t}, N \hat{t}, \boxed{Nz} \Rightarrow \Delta}$ (4)

Fig. 4. A derivation schema simulating a CA_G trace progression point in $CRTC_G^\omega + A$.

1061 RTC -formula ($RTC_{x,y} sx = y$)($0, u$) and the symbols \dagger , \ddagger and \boxtimes (in Fig. 5) denote straightforward
 1062 $CRTC_G^\omega$ proofs of the $*$ -translations of the appropriate axioms. Fig. 4 presents the core of the
 1063 derivation, and Fig. 5 presents auxiliary subderivations. The CA_G progression point from t to t' is
 1064 simulated by the $CRTC_G^\omega$ traces consisting of the boxed formulas in Fig. 4, following paths from
 1065 the conclusion to the premise (marked with $*$). These traces progress at the point indicated by the
 1066 doubly boxed formula. Thus, in the resulting $CRTC_G^\omega$ derivation, each CA_G trace pair (consisting
 1067 of terms) is simulated by $CRTC_G^\omega$ traces (consisting of formulas). \square

1068 This allows us to construct a global translation of CA_G proofs into $CRTC_G^\omega + A$ proofs.

1069 LEMMA 6.8. *If $\vdash_{CA_G} \Gamma \Rightarrow \Delta$ then $\vdash_{CRTC_G^\omega + A} \Gamma \Rightarrow \Delta$.*

1070 PROOF. Take a CA_G proof \mathcal{P} of $\Gamma \Rightarrow \Delta$. Let X be the set of variables occurring free in the
 1071 sequents appearing in \mathcal{P} . By Lemma 6.7(2), it follows that we can transform a CA_G pre-proof, via
 1072 the $[\cdot]_X^*$ -translation of each rule, into a $CRTC_G^\omega + A$ pre-proof with the same global structure. It
 1073 remains to show that each such $CRTC_G^\omega + A$ pre-proof resulting from a CA_G proof is also a $CRTC_G^\omega + A$
 1074 proof. That is, it satisfies the $CRTC_G^\omega$ global trace condition. Consider an arbitrary infinite path in
 1075 the $CRTC_G^\omega + A$ pre-proof. There are two cases to consider:
 1076
 1077

Inclusion of $\text{CRTC}_G^\omega + A$ in CA_G . We next show that from a $\text{CRTC}_G^\omega + A$ 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 $\text{CRTC}_G^\omega + A$ 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 ($\text{RTC}_{x,y} \varphi$)(s, t) $_n^{\bar{\beta}}$ is defined as follows:

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

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 $\varphi_z^{\hat{\beta}}$ 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 $\Gamma^{\hat{\beta}} \Rightarrow \Delta^{\hat{\beta}}$ 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 $\Gamma_1 \Rightarrow \Delta_1, \dots, \Gamma_n \Rightarrow \Delta_n$ and conclusion $\Gamma \Rightarrow \Delta$, we define its $\hat{\beta}$ -translation as the inference rule with premises $(\Gamma_1 \Rightarrow \Delta_1)^{\hat{\beta}}, \dots, (\Gamma_n \Rightarrow \Delta_n)^{\hat{\beta}}$ 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 $\Gamma, \Sigma^{\hat{\beta}} \Rightarrow \Delta$ 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 $n = sz$ with cuts. We abbreviate such a derivation using the label $(\hat{\beta})$.

We now show that the $\hat{\beta}$ -translation of each $\text{CRTC}_G^\omega + A$ inference rule can be derived in CA_G in such a way that there is a (progressing) CA_G trace simulating each (progressing) $\text{CRTC}_G^\omega + A$ 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 $\text{CRTC}_G^\omega + A$ axiom or inference rule, with conclusion S and premises S_1, \dots, 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 $\tau_n^{\hat{\beta}}$ in S and $(\tau')_m^{\hat{\beta}}$ 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 ,

$$\begin{array}{c}
1177 \\
1178 \\
1179 \\
1180 \\
1181 \\
1182 \\
1183 \\
1184 \\
1185 \\
1186 \\
1187 \\
1188 \\
1189 \\
1190 \\
1191 \\
1192 \\
1193 \\
1194 \\
1195 \\
1196 \\
1197 \\
1198 \\
1199 \\
1200 \\
1201 \\
1202 \\
1203 \\
1204 \\
1205 \\
1206 \\
1207 \\
1208 \\
1209 \\
1210 \\
1211 \\
1212 \\
1213 \\
1214 \\
1215 \\
1216 \\
1217 \\
1218 \\
1219 \\
1220 \\
1221 \\
1222 \\
1223 \\
1224 \\
1225
\end{array}$$

$$\begin{array}{c}
\frac{}{\varphi^\beta \left\{ \frac{s}{x}, \frac{t}{y} \right\} \Rightarrow \varphi^\beta \left\{ \frac{s}{x}, \frac{t}{y} \right\}} \text{(Ax)} \\
\frac{}{s = v, t = w, \varphi^\beta \left\{ \frac{v}{x}, \frac{w}{y} \right\} \Rightarrow \varphi^\beta \left\{ \frac{s}{x}, \frac{t}{y} \right\}} \text{(=L)} \\
\frac{B(c, 0, s), B(c, s0, t), B(c, 0, v), B(c, s0, w), \varphi^\beta \left\{ \frac{v}{x}, \frac{w}{y} \right\} \Rightarrow \varphi^\beta \left\{ \frac{s}{x}, \frac{t}{y} \right\}}{B(c, 0, s), B(c, s0, t), \forall u \leq 0. \vartheta(u) \Rightarrow \varphi^\beta \left\{ \frac{s}{x}, \frac{t}{y} \right\}} \text{(B)} \\
\frac{}{z = 0, \Sigma(z) \Rightarrow \varphi^\beta \left\{ \frac{s}{x}, \frac{t}{y} \right\}} \text{(}\leq_0/\exists\text{L)} \\
\frac{}{z = 0, \Sigma(z) \Rightarrow \varphi^\beta \left\{ \frac{s}{x}, \frac{t}{y} \right\}} \text{(WL/=L)} \\
\frac{}{z = 0, \Sigma(z) \Rightarrow s = s} \text{(=R)} \\
\frac{}{z = 0, \Sigma(z) \Rightarrow s = s \vee A(0, s, s)} \text{(}\vee\text{R)} \\
\frac{}{z = 0, \Sigma(z) \Rightarrow 0 < s0} \text{(PA-Ax)} \\
\frac{}{z = 0, \Sigma(z) \Rightarrow 0 < n} \text{(=L)} \\
\frac{}{z = 0, \Sigma(z) \Rightarrow \exists z, m. m < n \wedge (s = z \vee A(m, s, z)) \wedge \varphi^\beta \left\{ \frac{z}{x}, \frac{t}{y} \right\}} \text{(}\exists\text{R}/\wedge\text{R)} \\
\text{(a) One step from } s \text{ to } t.
\end{array}$$

$$\begin{array}{c}
1194 \\
1195 \\
1196 \\
1197 \\
1198 \\
1199 \\
1200 \\
1201 \\
1202 \\
1203 \\
1204 \\
1205 \\
1206 \\
1207 \\
1208 \\
1209 \\
1210 \\
1211 \\
1212 \\
1213 \\
1214 \\
1215 \\
1216 \\
1217 \\
1218 \\
1219 \\
1220 \\
1221 \\
1222 \\
1223 \\
1224 \\
1225
\end{array}$$

$$\begin{array}{c}
\frac{\frac{}{\Pi(z') \Rightarrow sz' = sz'} \text{(=R)} \quad \frac{}{\Pi(z') \Rightarrow B(c, 0, s)} \text{(Ax)} \quad \frac{}{\Pi(z') \Rightarrow B(c, sz', v)} \text{(Ax)} \quad \frac{}{\Pi(z') \Rightarrow \forall u \leq z'. \vartheta(u)} \text{(Ax)}}{\frac{}{\Pi(z') \Rightarrow \exists z, c. sz' = sz \wedge B(c, 0, s) \wedge B(c, sz, v) \wedge \forall u \leq z. \vartheta(u)} \text{(}\vee\text{R)}} \text{(}\exists\text{R}/\wedge\text{R)} \\
\frac{}{\Pi(z') \Rightarrow s = v \vee A(sz', s, v)} \\
\frac{}{\varphi^\beta \left\{ \frac{v}{x}, \frac{t}{y} \right\} \Rightarrow \varphi^\beta \left\{ \frac{v}{x}, \frac{t}{y} \right\}} \text{(Ax)} \\
\frac{}{t = w, \varphi^\beta \left\{ \frac{v}{x}, \frac{w}{y} \right\} \Rightarrow \varphi^\beta \left\{ \frac{v}{x}, \frac{t}{y} \right\}} \text{(=L)} \\
\frac{}{\Pi(z') \Rightarrow sz' < ssz'} \text{(PA-Ax)} \\
\frac{}{\Pi(z') \Rightarrow sz' < n} \text{(=L)} \\
\frac{}{\Pi(z') \Rightarrow \varphi^\beta \left\{ \frac{v}{x}, \frac{t}{y} \right\}} \text{(B/WL)} \\
\frac{}{\Pi(z') \Rightarrow \exists z, m. m < n \wedge (s = z \vee A(m, s, z)) \wedge \varphi^\beta \left\{ \frac{z}{x}, \frac{t}{y} \right\}} \text{(}\exists\text{R}/\wedge\text{R)} \\
\frac{}{n = ssz', B(c, 0, s), B(c, ssz', t), \vartheta(ssz'), \forall u \leq z'. \vartheta(u) \Rightarrow \psi} \text{(}\exists\text{L}/\wedge\text{L)} \\
\frac{}{n = ssz', B(c, 0, s), B(c, ssz', t), \forall u \leq z'. \vartheta(u) \Rightarrow \psi} \text{(}\leq_s)} \\
\frac{}{\exists z'. z = sz', \Sigma(z) \Rightarrow \psi} \text{(}\exists\text{L/=L)} \\
\text{(b) Multi-step from } s \text{ to } t.
\end{array}$$

$$\begin{array}{c}
\frac{}{\vdots} \\
\frac{}{z = 0, \Sigma(z) \Rightarrow \psi} \quad \frac{}{\exists z'. z = sz', \Sigma(z) \Rightarrow \psi} \\
\frac{}{\Rightarrow z = 0 \vee \exists z'. z = sz'} \quad \frac{}{z = 0 \vee \exists z'. z = sz', \Sigma(z) \Rightarrow \psi} \text{(}\vee\text{L)} \\
\frac{}{\Sigma(z) \Rightarrow \exists z, m. m < n \wedge (s = z \vee A(m, s, z)) \wedge \varphi^\beta \left\{ \frac{z}{x}, \frac{t}{y} \right\}} \text{(}\exists\text{L)} \\
\frac{}{A(n, s, t) \Rightarrow \exists z, m. m < n \wedge (s = z \vee A(m, s, z)) \wedge \varphi^\beta \left\{ \frac{z}{x}, \frac{t}{y} \right\}} \text{(Cut)}
\end{array}$$

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 $\exists v, w. B(c, r, v) \wedge B(c, sr, w) \wedge \varphi^\beta \left\{ \frac{v}{x}, \frac{w}{y} \right\}$, and $A(r, s, t)$ abbreviate $\exists z, c. r = sz \wedge B(c, 0, s) \wedge B(c, sz, t) \wedge \forall u \leq z. \vartheta(u)$. Additionally, we define the following two abbreviations for sets of formulas:

$$\Sigma(r) = \{n = sr, B(c, 0, s), B(c, sr, t), \forall u \leq r. \vartheta(u)\}$$

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

Moreover, note the following.

1226 Fig. 6
 1227 \vdots
 1228 $A(n, s, t) \Rightarrow \exists z, m . m < n \wedge (s = z \vee A(m, s, z)) \wedge \varphi^\beta \left\{ \frac{z}{x}, \frac{t}{y} \right\}$
 1229 \vdots
 1230 $\frac{\Gamma^{\hat{\beta}}, (s = z \vee A(\overline{m}, s, z)), \varphi^{\hat{\beta}} \left\{ \frac{z}{x}, \frac{t}{y} \right\} \Rightarrow \Delta^\beta}{\Gamma^{\hat{\beta}}, (s = z \vee A(\overline{m}, s, z)), \varphi^\beta \left\{ \frac{z}{x}, \frac{t}{y} \right\} \Rightarrow \Delta^\beta} (\hat{\beta})$
 1231 \vdots
 1232 $\frac{\Gamma^{\hat{\beta}}, \overline{m} < n, (s = z \vee A(m, s, z)), \varphi^\beta \left\{ \frac{z}{x}, \frac{t}{y} \right\} \Rightarrow \Delta^\beta}{\Gamma^{\hat{\beta}}, \exists z, m . m < \overline{n} \wedge (s = z \vee A(m, s, z)) \wedge \varphi^\beta \left\{ \frac{z}{x}, \frac{t}{y} \right\} \Rightarrow \Delta^\beta} (\text{WL})$
 1233 \vdots
 1234 $\frac{\Gamma^{\hat{\beta}}, \exists z, m . m < \overline{n} \wedge (s = z \vee A(m, s, z)) \wedge \varphi^\beta \left\{ \frac{z}{x}, \frac{t}{y} \right\} \Rightarrow \Delta^\beta}{\Gamma^{\hat{\beta}}, A(\overline{n}), s, t \Rightarrow \Delta^\beta} (\exists\text{L}/\wedge\text{L})$
 1235 \vdots
 1236 $\frac{\Gamma^{\hat{\beta}}, A(\overline{n}), s, t \Rightarrow \Delta^\beta}{\Gamma^{\hat{\beta}}, s = t \vee A(\overline{n}), s, t \Rightarrow \Delta^\beta} (\text{Cut})$
 1237 $\frac{(\Gamma')^{\hat{\beta}} \left\{ \frac{s}{u}, \frac{t}{w} \right\} \Rightarrow (\Delta')^\beta \left\{ \frac{s}{u}, \frac{t}{w} \right\}}{\Gamma^{\hat{\beta}}, s = t \Rightarrow \Delta^\beta} (=L)$
 1238 \vdots
 1239 $\frac{\Gamma^{\hat{\beta}}, s = t \Rightarrow \Delta^\beta}{\Gamma^{\hat{\beta}}, s = t \vee A(\overline{n}), s, t \Rightarrow \Delta^\beta} (\vee\text{L})$
 1240 \vdots

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

1241 i) We can easily derive $\Rightarrow z = 0 \vee \exists z' . z = sz'$ using standard first-order rules and the axioms
 1242 of CA_G ; we refer to this derivation using \dagger .

1243 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
 1244 straightforwardly derive both $\forall u \leq 0 . \gamma(u) \Rightarrow \gamma(0)$ and $\forall u \leq st . \gamma(u) \Rightarrow \gamma(st) \wedge \forall u \leq t . \gamma(u)$; for
 1245 brevity, we refer to an instance of the (Cut) rule that applies these sequents using the labels (\leq_0)
 1246 and (\leq_s) , respectively.

1247 iii) Recall that, since the formula B captures a β -function, we may derive $B(r, s, t), B(r, s, u) \Rightarrow$
 1248 $t = u$; we abbreviate using the label (B) instances of (Cut) that apply an instance of this sequent.

1249 Using these elements, Fig. 6 shows a derivation (in which we have abbreviated the consequent
 1250 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,
 1251 using Fig. 6 as a subderivation, we derive the $\hat{\beta}$ -translation of Rule (4) in CA_G as shown in Fig. 7.
 1252 Recall that the rule labelled $(\hat{\beta})$ abbreviates a derivation as described above in Remark 1. Note that
 1253 this derivation admits non-progressing traces for all the $\hat{\beta}$ -translation variable parameters in Γ .
 1254 However the crucial feature is that there is a CA_G trace (indicated by the boxed variables in Fig. 7)
 1255 from the variable parameter n in the conclusion to m in the right-hand premise, which progresses at
 1256 the sequent containing the double-boxed variable m . Also, since the context $\Gamma^{\hat{\beta}}$ is preserved along
 1257 the paths in the derivation from conclusion to premises, all non-progressing traces are simulated
 1258 as well. \square

1259 LEMMA 6.13. *If $\vdash_{\text{CRTC}_G^{\omega+A}} \Gamma \Rightarrow \Delta$ then $\vdash_{\text{CA}_G} \Gamma^\beta \Rightarrow \Delta^\beta$.*

1260 PROOF. By Remark 1, it suffices to show that if $\vdash_{\text{CRTC}_G^{\omega+A}} \Gamma \Rightarrow \Delta$ then $\vdash_{\text{CA}_G} (\Gamma \Rightarrow \Delta)^\beta$. Using the
 1261 derivations of $\hat{\beta}$ -translations of the inference rules given in Lemma 6.12, from a $\text{CRTC}_G^{\omega+A}$ pre-proof
 1262 we can build a CA_G pre-proof with the same global structure. For each bud in the resulting CA_G
 1263 pre-proof, we first apply an instance of the substitution rule that substitutes each variable parameter
 1264 of the $\hat{\beta}$ -translated RTC -formulas in order to match those used in its companion. Notice that this is
 1265 possible, since the parameter variable is unique for the $\hat{\beta}$ -translation of each RTC sub-formula. We
 1266 can then form a cycle in the CA_G pre-proof. Since the CRTC_G^{ω} traces for each rule are simulated by
 1267 the CA_G derived rules, for each trace following a (finite or infinite) path in the $\text{CRTC}_G^{\omega+A}$ pre-proof
 1268 there is a trace following the corresponding path in the CA_G pre-proof containing a progression

1275 point for each progression point in the $\text{CRTC}_G^\omega + A$ trace. From this it follows that if the $\text{CRTC}_G^\omega + A$
 1276 pre-proof satisfies the CRTC_G^ω global trace condition, then its translation satisfies the CA_G global
 1277 trace condition. \square

1278

1279 **THEOREM 6.14.** $\vdash_{\text{CRTC}_G^\omega + A} \Gamma \Rightarrow \Delta$ iff $\vdash_{\text{CA}_G} \Gamma^\beta \Rightarrow \Delta^\beta$.

1280

1281 **PROOF.** By Lemmas 6.10 and 6.13. \square

1282 **Equivalence of $\text{RTC}_G + A$ and $\text{CRTC}_G^\omega + A$.** These results allow us to show an equivalence
 1283 between the finitary and cyclic systems for TC with arithmetic.

1284

1285 **THEOREM 6.15.** $\text{RTC}_G + A$ and $\text{CRTC}_G^\omega + A$ are equivalent.

1286

1287 **PROOF.** The fact that $\text{RTC}_G + A \subseteq \text{CRTC}_G^\omega + A$ follows immediately from Theorem 6.2. For the
 1288 converse, suppose $\Gamma \Rightarrow \Delta$ is provable in $\text{CRTC}_G^\omega + A$. By Lemma 6.13 we get that $\vdash_{\text{CA}_G} \Gamma^\beta \Rightarrow \Delta^\beta$.
 1289 Using the equivalence between CA_G and PA_G , we obtain $\vdash_{\text{PA}_G} \Gamma^\beta \Rightarrow \Delta^\beta$. Then we conclude using
 1290 Theorem 6.5(2). \square

1290

1291 Note that the result above can easily be extended to show that adding the same set of additional
 1292 axioms to both $\text{RTC}_G + A$ and $\text{CRTC}_G^\omega + A$ results in equivalent systems. Also note that in the systems
 1293 with pairs, to embed arithmetics there is no need to explicitly include addition and its axioms.
 1294 Thus, by only including the signature $\{0, s\}$ and the corresponding axioms for it we can obtain that
 1295 $\langle \text{RTC} \rangle_G + A$ and $\langle \text{CRTC} \rangle_G^\omega + A$ are equivalent.

1296

1297 In [7], the equivalence result of [34] was improved to show it holds for any set of inductive
 1298 predicates containing the natural number predicate N . On the one hand, our result goes beyond
 1299 that of [7] as it shows the equivalence for systems with a richer notion of inductive definition, due
 1300 to the expressiveness of TC. On the other hand, TC does not support restricting the set of inductive
 1301 predicates, i.e. the RTC operator may operate on any formula in the language. To obtain a finer
 1302 result which corresponds to that of [7] we need to further explore the transformations between
 1303 proofs in the two systems. This is left for future work.

1303

1304 **6.2.2 The General Case.** As mentioned, the general equivalence conjecture between LKID and
 1305 CLKID^ω was refuted in [6], by providing a concrete example of a statement which is provable
 1306 in the cyclic system but not in the explicit one. The statement, involving the 2-Hydra predicate
 1307 described above in Section 2.2, is the following:

1307

$$1308 ((\forall x. 0 \neq sx) \wedge (\forall x. \forall y. sx = sy \rightarrow x = y)) \rightarrow \forall x. \forall y. ((N(x) \wedge N(y)) \rightarrow H(x, y))$$

1309

1310 where N is a predicate defined by the induction scheme for natural numbers. That is, assuming
 1311 that 0 is not a successor element and that the successor function is injective, then every pair of
 1312 natural numbers is related by the 2-Hydra predicate. The non-provability of this statement in LKID
 1313 (i.e. using explicit induction) follows from the existence of a Henkin model in which the statement
 1314 does not hold, whose construction was given in [6].

1314

1315 However, a careful examination of this counter-example reveals that it only refutes a strong form
 1316 of the conjecture, according to which both systems are based on the same set of productions. In
 1317 fact, already in [6] it is shown that if the explicit system is extended by another inductive predicate,
 1318 namely one expressing the standard ordering \leq over natural numbers, then the 2-Hydra statement
 1319 above becomes provable. Therefore, the less strict formulation of the question, namely whether for
 1320 any proof in CLKID_ϕ^ω there is a proof in $\text{LKID}_{\phi'}$ for some $\phi' \supseteq \phi$, has not yet been resolved. Notice
 1321 that in TC the equivalence question is of this weaker variety, since the RTC operator ‘generates’ all
 1322 inductive definitions at once. That is, there is no *a priori* restriction on the inductive predicates
 1323 one is allowed to use. Indeed, the 2-Hydra counter-example from [6] can be expressed in \mathcal{L}_{RTC}

1323

1324 (cf. Section 2.2) and proved in CRTC_G^ω . However this does not produce a counter-example to the
 1325 equivalence of the two TC proof systems, since it is also provable in RTC_G due to the fact that $s \leq t$
 1326 is definable via the formula $(\text{RTC}_{w,u} \text{sw} = u)(s, t)$.

1327 Despite our best efforts, we have not yet managed to settle this question, which appears to
 1328 be harder to resolve in the TC setting. One possible approach to solving it is the semantical one,
 1329 i.e. exploiting the fact that the explicit system is known to be sound w.r.t. Henkin semantics. This is
 1330 what was done in [6]. Thus, to show strict inclusion one could construct an alternative statement
 1331 that is provable in CRTC_G^ω whilst also demonstrating a Henkin model for TC that is not a model of
 1332 the statement. However, it has become apparent through communications with one of the authors
 1333 of [6] that constructing a Henkin model appears to be much more difficult for TC than for LKID, due
 1334 to its rich inductive power. In particular, it is not at all clear whether the structure that underpins
 1335 the LKID counter-model for 2-Hydra admits a Henkin model for TC [5]. Alternatively, to prove
 1336 equivalence, one could show that CRTC_G^ω is also sound w.r.t. Henkin semantics. Here, again, proving
 1337 this does not seem to be straightforward.

1338 In our setting, there is also the question of the inclusion of CRTC_G^ω in NCRTC_G^ω , which amounts
 1339 to the question of whether overlapping cycles can be eliminated. Moreover, we can ask if NCRTC_G^ω
 1340 is included in RTC_G , independently of whether this also holds for CRTC_G^ω . Again, the semantic
 1341 approach described above may prove fruitful in answering these questions.

1342

1343

7 CONCLUSIONS AND FUTURE WORK

1344 Transitive closure logic seems to offer a congenial framework for inductive reasoning. In this paper
 1345 we have enhanced its proof theory by developing a natural infinitary proof system which is cut-free
 1346 complete for its standard semantics. We further explored the restriction of this framework to cyclic
 1347 proofs which provides the basis for an effective system for automating inductive reasoning and
 1348 subsumes its explicit proof system. In particular, we syntactically identified a subset of cyclic proofs
 1349 that is Henkin-complete.

1350 As mentioned in the introduction, as well as throughout the paper, this research was motivated
 1351 by other work on systems of inductive definitions, particularly the LKID framework of [11], its
 1352 infinitary counterpart LKID^ω , and its cyclic subsystem CLKID^ω . In terms of the expressive power
 1353 of the underlying logic, TC (assuming pairs) subsumes the inductive machinery underlying LKID.
 1354 This is because for any inductive predicate P of LKID, there is an \mathcal{L}_{RTC} formula ψ such that for
 1355 every standard admissible structure M for \mathcal{L}_{RTC} , P has the same interpretation as ψ under M . This
 1356 is due to Theorem 3 in [2] and the fact that the interpretation of P must necessarily be a recursively
 1357 enumerable set. As for the converse inclusion, for any positive \mathcal{L}_{RTC} formula there is a production of
 1358 a corresponding LKID inductive definition. This is due to the fact that since productions can be seen
 1359 as Horn clauses they can capture disjunction and conjunction. However, the RTC operator can also
 1360 be applied on complex formulas (whereas LKID productions only consider atomic predicates). This
 1361 indicates that TC might be more expressive. It was noted in [11, p. 1180] that complex formulas may
 1362 be handled by stratifying the theory of LKID, similar to [27], but the issue of relative expressiveness
 1363 of the resulting theory is not addressed. While we strongly believe it is the case that TC is strictly
 1364 more expressive than the logic of LKID, proving so is left for future work. Also left for future
 1365 research is establishing the comparative status of the corresponding formal proof systems.

1366 It is already clear that TC logic, as a framework, diverges from existing systems for inductive
 1367 reasoning (e.g. LKID) in interesting, non-trivial ways. At this point, it is still unclear whether or
 1368 not the added flexibility of transitive closure logic over that of LKID is sufficient for establishing an
 1369 equivalence between RTC_G and CRTC_G^ω even in the absence of arithmetic. Thus an immediate open
 1370 question that remains for future work is that of the (in)equivalence of the systems in the general
 1371 case, as discussed in Section 6.2. However the question of general equivalence notwithstanding, the
 1372

1373 uniformity provided by the transitive closure operator may offer a way to better study the more
 1374 subtle relationship between implicit and explicit induction. That is, it can help in the investigation
 1375 of the connections between cuts required in each system, or the relative complexity of proofs that
 1376 each system admits.

1377 In addition, several other questions and directions for further study naturally arise from the work
 1378 of this paper. An obvious one would be to implement our cyclic proof system in order to investigate
 1379 the practicalities of using TC logic to support automated inductive reasoning. In particular, we
 1380 would like to further study the applications of the logic, and the proof systems we have defined for
 1381 it, in various areas of computer science such as those described in Section 2.2. Furthermore, our
 1382 preliminary investigations suggest we could enable coinductive reasoning in a variant of the formal
 1383 system. Determining how naturally such a variant would capture styles of coinductive reasoning
 1384 commonly found in the literature, and the extent of its expressivity, is left for future work.
 1385

1386 REFERENCES

- 1387 [1] Bahareh Afshari and Graham E. Leigh. Cut-free Completeness for Modal Mu-calculus. In *Proceedings of the 32nd*
 1388 *Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20–23, 2017*, pages 1–12,
 1389 2017. doi:[10.1109/LICS.2017.8005088](https://doi.org/10.1109/LICS.2017.8005088).
- 1390 [2] Arnon Avron. Transitive Closure and the Mechanization of Mathematics. In F. D. Kamareddine, editor, *Thirty Five*
 1391 *Years of Automating Mathematics*, volume 28 of *Applied Logic Series*, pages 149–171. Springer, Netherlands, 2003.
 1392 doi:[10.1007/978-94-017-0253-9_7](https://doi.org/10.1007/978-94-017-0253-9_7).
- 1393 [3] David Baelde, Amina Doumane, and Alexis Saurin. Infinitary Proof Theory: the Multiplicative Additive Case. In
 1394 *Proceedings of the 25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 – September 1, 2016,*
 1395 *Marseille, France*, pages 42:1–42:17, 2016. doi:[10.4230/LIPIcs.CSL.2016.42](https://doi.org/10.4230/LIPIcs.CSL.2016.42).
- 1396 [4] Johan van Benthem. *The Logic of Time*. Synthese Library. Springer Netherlands, 2nd edition, 2012.
- 1397 [5] Stefano Berardi. Personal Communication (10th April 2018).
- 1398 [6] Stefano Berardi and Makoto Tatsuta. Classical System of Martin-Löf’s Inductive Definitions Is Not Equivalent to Cyclic
 1399 Proof System. In *Proceedings of the 20th International Conference on Foundations of Software Science and Computation*
 1400 *Structures, FOSSACS 2017, Uppsala, Sweden, April 22–29, 2017*, pages 301–317, Berlin, Heidelberg, 2017. Springer Berlin
 1401 Heidelberg. doi:[10.1007/978-3-662-54458-7_18](https://doi.org/10.1007/978-3-662-54458-7_18).
- 1402 [7] Stefano Berardi and Makoto Tatsuta. Equivalence of Inductive Definitions and Cyclic Proofs Under Arithmetic. In
 1403 *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June*
 1404 *20–23, 2017*, pages 1–12, 2017. doi:[10.1109/LICS.2017.8005114](https://doi.org/10.1109/LICS.2017.8005114).
- 1405 [8] Patrick Blackburn and Johan van Benthem. Modal Logic: A Semantical Perspective. In Patrick Blackburn, Johan van
 1406 Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*,
 1407 pages 1–84. Elsevier, 2007. doi:[10.1016/S1570-2464\(07\)80004-8](https://doi.org/10.1016/S1570-2464(07)80004-8).
- 1408 [9] James Brotherston. Formalised Inductive Reasoning in the Logic of Bunched Implications. In *Proceedings of Static*
 1409 *Analysis, 14th International Symposium, SAS 2007, Kongens Lyngby, Denmark, August 22–24, 2007*, pages 87–103, 2007.
 1410 doi:[10.1007/978-3-540-74061-2_6](https://doi.org/10.1007/978-3-540-74061-2_6).
- 1411 [10] James Brotherston, Richard Bornat, and Cristiano Calcagno. Cyclic Proofs of Program Termination in Separation
 1412 Logic. In *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL*
 1413 *2008, San Francisco, California, USA, January 7–12, 2008*, pages 101–112, 2008. doi:[10.1145/1328438.1328453](https://doi.org/10.1145/1328438.1328453).
- 1414 [11] James Brotherston and Alex Simpson. Sequent Calculi for Induction and Infinite Descent. *Journal of Logic and*
 1415 *Computation*, 21(6):1177–1216, 2010. doi:[10.1093/logcom/exq052](https://doi.org/10.1093/logcom/exq052).
- 1416 [12] Samuel R. Buss. *Handbook of Proof Theory*. Studies in Logic and the Foundations of Mathematics. Elsevier Science,
 1417 1998.
- 1418 [13] Liron Cohen. Ancestral Logic and Equivalent Systems. Master’s thesis, Tel-Aviv University, Israel, 2010.
- 1419 [14] Liron Cohen. Completeness for Ancestral Logic via a Computationally-Meaningful Semantics. In *Proceedings of the*
 1420 *26th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2017,*
 1421 *Brasilia, Brazil, September 25–28, 2017*, pages 247–260, 2017. doi:[10.1007/978-3-319-66902-1_15](https://doi.org/10.1007/978-3-319-66902-1_15).
- 1422 [15] Liron Cohen and Arnon Avron. Ancestral Logic: A Proof Theoretical Study. In U. Kohlenbach et al., editor, *Logic,*
 1423 *Language, Information, and Computation*, volume 8652 of *Lecture Notes in Computer Science*, pages 137–151. Springer,
 1424 2014. doi:[10.1007/978-3-662-44145-9_10](https://doi.org/10.1007/978-3-662-44145-9_10).
- 1425 [16] Liron Cohen and Arnon Avron. The Middle Ground–Ancestral Logic. *Synthese*, pages 1–23, 2015.
 1426 doi:[10.1007/s11229-015-0784-3](https://doi.org/10.1007/s11229-015-0784-3).

- 1422 [17] Liron Cohen and Reuben N. S. Rowe. Uniform Inductive Reasoning in Transitive Closure Logic via Infinite Descent. In
 1423 *Proceedings of the 27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4–7, 2018, Birmingham,*
 1424 *UK*, pages 16:1–16:17, 2018. doi:10.4230/LIPICs.CSL.2018.16.
- 1425 [18] Stephen A. Cook and Robert A. Reckhow. The Relative Efficiency of Propositional Proof Systems. *The Journal of*
 1426 *Symbolic Logic*, 44(1):36–50, 1979. doi:10.2307/2273702.
- 1427 [19] Bruno Courcelle. Fundamental Properties of Infinite Trees. *Theoretical Computer Science*, 25:95–169, 1983.
 1428 doi:10.1016/0304-3975(83)90059-2.
- 1429 [20] Anupam Das and Damien Pous. A Cut-Free Cyclic Proof System for Kleene Algebra. In *Proceedings of the 26th*
 1430 *International Conference Automated Reasoning with Analytic Tableaux and Related Methods, TABLEUX 2017, Brasilia,*
 1431 *Brazil, September 25–28, 2017*, pages 261–277, 2017. doi:10.1007/978-3-319-66902-1_16.
- 1432 [21] Anupam Das and Damien Pous. Non-Wellfounded Proof Theory for (Kleene+Action)(Algebras+Lattices). In *Proceedings*
 1433 *of the 27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4–7, 2018, Birmingham, UK*, pages
 1434 19:1–19:18, 2018. doi:10.4230/LIPICs.CSL.2018.19.
- 1435 [22] Amina Doumane. Constructive Completeness for the Linear-time μ -calculus. In *Proceedings of the 32nd Annual*
 1436 *ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20–23, 2017*, pages 1–12, 2017.
 1437 doi:10.1109/LICS.2017.8005075.
- 1438 [23] Gerhard Gentzen. Untersuchungen über das Logische Schließen. I. *Mathematische Zeitschrift*, 39(1):176–210, 1935.
 1439 doi:10.1007/BF01201353.
- 1440 [24] Leon Henkin. Completeness in the Theory of Types. *Journal of Symbolic Logic*, 15(2):81–91, 1950. doi:10.2307/2266967.
- 1441 [25] Ryo Kashima and Keishi Okamoto. General Models and Completeness of First-order Modal μ -calculus. *Journal of*
 1442 *Logic and Computation*, 18(4):497–507, 2008. doi:10.1093/logcom/exm077.
- 1443 [26] Laurie Kirby and Jeff Paris. Accessible Independence Results for Peano Arithmetic. *Bulletin of the London Mathematical*
 1444 *Society*, 14(4):285–293, 1982. doi:10.1112/blms/14.4.285.
- 1445 [27] Per Martin-Löf. Hauptsatz for the Intuitionistic Theory of Iterated Inductive Definitions. In J. E. Fenstad, editor,
 1446 *Proceedings of the Second Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of*
 1447 *Mathematics*, pages 179–216. Elsevier, 1971. doi:10.1016/S0049-237X(08)70847-4.
- 1448 [28] Per Martin-Löf and Giovanni Sambin. *Intuitionistic Type Theory*, volume 9. Bibliopolis Napoli, 1984.
- 1449 [29] Raymond McDowell and Dale Miller. Cut-elimination for a Logic with Definitions and Induction. *Theoretical Computer*
 1450 *Science*, 232(1-2):91–119, 2000. doi:10.1016/S0304-3975(99)00171-1.
- 1451 [30] Remi Nolllet, Christine Tasson, and Alexis Saurin. The Complexity of Thread Criterion for Least and Greatest fixed
 1452 points. In *Proceedings of the 27th International Conference on Automated Reasoning with Analytic Tableaux and Related*
 1453 *Methods, TABLEUX 2019, London, UK, September 3–7, 2019*, 2019.
- 1454 [31] Amir Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th Annual Symposium on Foundations of*
 1455 *Computer Science (SFCS 1977)*, pages 46–57, 1977. doi:10.1109/SFCS.1977.32.
- 1456 [32] Reuben N. S. Rowe and James Brotherston. Automatic Cyclic Termination Proofs for Recursive Procedures in Separation
 1457 Logic. In *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2017, Paris, France,*
 1458 *January 16–17, 2017*, pages 53–65, 2017. doi:10.1145/3018610.3018623.
- 1459 [33] Luigi Santocanale. A Calculus of Circular Proofs and Its Categorical Semantics. In Mogens Nielsen and Uffe Engberg,
 1460 editors, *Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures,*
 1461 *FOSSACS 2002, Grenoble, France, April 8–12, 2002*, pages 357–371, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
 1462 doi:10.1007/3-540-45931-6_25.
- 1463 [34] Alex Simpson. Cyclic Arithmetic Is Equivalent to Peano Arithmetic. In *Proceedings of the 20th International Conference*
 1464 *on Foundations of Software Science and Computation Structures, FOSSACS 2017, Uppsala, Sweden, April 22–29, 2017*, pages
 1465 283–300, 2017. doi:10.1007/978-3-662-54458-7_17.
- 1466 [35] Christoph Sprenger and Mads Dam. On the Structure of Inductive Reasoning: Circular and Tree-Shaped Proofs in the
 1467 μ Calculus. In Andrew D. Gordon, editor, *Proceedings of the 6th International Conference on Foundations of Software*
 1468 *Science and Computation Structures, FOSSACS 2003, Warsaw, Poland, April 7–11, 2003*, pages 425–440, Berlin, Heidelberg,
 1469 2003. Springer Berlin Heidelberg. doi:10.1007/3-540-36576-1_27.
- 1470 [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.