Uniform Inductive Reasoning in Transitive Closure Logic via Infinite Descent

Liron Cohen[1]
Dept. of Computer Science, Cornell University, NY, USA
lironcohen@cornell.edu

Reuben N. S. Rowe[2]
School of Computing, University of Kent, Canterbury, UK
r.n.s.rowe@kent.ac.uk
0000-0002-4271-9078

Abstract

Transitive closure logic is a known extension of first-order logic obtained by introducing a transitive closure operator. While other extensions of first-order logic with inductive definitions are a priori parametrized by a set of inductive definitions, the addition of the transitive closure operator uniformly captures all finitary inductive definitions. In this paper we present an *infinitary* proof system for transitive closure logic which is an *infinite descent*-style counterpart to the existing (explicit induction) proof system for the logic. We show that, as for similar systems for first-order logic with inductive definitions, our 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.

2012 ACM Subject Classification Theory of computation → Proof theory; Theory of computation → Automated reasoning

Keywords and phrases Induction, Transitive Closure, Infinitary Proof Systems, Cyclic Proof Systems, Soundness, Completeness, Standard Semantics, Henkin Semantics

Digital Object Identifier 10.4230/LIPIcs.CSL.2018.16

1 Introduction

A core technique in mathematical reasoning is that of *induction*. This is especially true in computer science, where it plays a central role in reasoning about recursive data and computations. Formal systems for mathematical reasoning usually capture the notion of inductive reasoning via one or more inference rules that express the general induction schemes, or principles, that hold for the elements being reasoned over.

Increasingly, we are concerned with not only being able to formalise as much mathematical reasoning as possible, but also with doing so in an effective way. In other words, we seek to be able to automate such reasoning. *Transitive closure* (TC) logic has been identified as a potential candidate for a minimal, 'most general' system for inductive reasoning, which is also very suitable for automation [1, 10, 11]. TC adds to first-order logic a single operator

[1] Supported by Fulbright Post-doctoral Scholar program, Weizmann Institute of Science National Postdoctoral Award program for Advancing Women in Science, and Eric and Wendy Schmidt Postdoctoral Award program for Women in Mathematical and Computing Sciences.
[2] Supported by EPSRC Grant No. EP/N028759/1.

for forming binary relations: specifically, the transitive closures of arbitrary formulas (more precisely, the transitive closure of the binary relation induced by a formula with respect to two distinct variables). In this work, for simplicity, we use a reflexive form of the operator; however the two forms are equivalent in the presence of equality. This modest addition affords enormous expressive power: namely 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. Thus, bespoke induction principles do not need to be added to, or embedded within, the logic; instead, all induction schemes are available within a single, unified language. In this respect, the transitive closure operator resembles the W-type [22], which also provides a single type constructor from which one can uniformly define a variety of inductive types.

TC logic is intermediate between first- and second-order logic. Furthermore, since the TC operator is a particular instance of a least fixed point operator, TC logic is also subsumed by fixed-point logics such as the μ-calculus [19]. However, despite its minimality TC logic retains enough expressivity to capture inductive reasoning, as well as to subsume arithmetics (see Section 4.2.1). Moreover, from a proof theoretical perspective the conciseness of the logic makes it of particular interest. The use of only one constructor of course comes with a price: namely, formalizations (mostly of non-linear induction schemes) may be somewhat complex. However, they generally do not require as complex an encoding as in arithmetics, since the TC operator can be applied on any formula and thus (depending on the underlying signature) more naturally encode induction on sets more complex than the natural numbers.

Since its expressiveness entails that TC logic subsumes arithmetics, by Gödel's result, any effective proof system for it must necessarily be incomplete for the standard semantics. Notwithstanding, a natural, effective proof system which is sound for TC logic was shown to be complete with respect to a generalized form of Henkin semantics [9]. In this paper, following similar developments in other formalizations for fixed point logics and inductive reasoning (see e.g. [4, 5, 6, 24, 27]), we present an *infinitary* proof theory for TC logic which, as far as we know, is the first system that is (cut-free) complete with respect to the standard semantics. More specifically, our system employs infinite-height, rather than infinite-width proofs (see Section 3.2). The soundness of such infinitary proof theories is underpinned by the principle of *infinite descent*: proofs are permitted to be infinite, non-well-founded trees, but subject to the restriction that every infinite path in the proof admits some infinite descent. The descent is witnessed by tracing terms or formulas for which we can give a correspondence with elements of a well-founded set. In particular, we can trace terms that denote elements of an inductively defined (well-founded) set. For this reason, such theories are considered systems of implicit induction, as opposed to those which employ explicit rules for applying induction principles. While a full infinitary proof theory is clearly not effective, in the aforementioned sense, such a system can be obtained by restricting consideration to only the *regular* infinite proofs. These are precisely those proofs that can be finitely represented as (possibly cyclic) graphs.

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

proof system for Kleene algebra [15], providing further evidence of its utility for automation.

In the setting of TC logic, we observe some further benefits over more traditional formal systems of inductive definitions and their infinitary proof theories (cf. LKID [6, 21]). TC (with a pairing function) has all first-order definable finitary inductive definitions immediately 'available' within the language of the logic: as with inductive hypotheses, one does not need to 'know' in advance which induction schemes will be required. Moreover, the use of a single transitive closure operator provides a uniform treatment of all induction schemes. That is, instead of having a proof system parameterized by a set of inductive predicates and rules for them (as is the case in LKID), TC offers a single proof system with a single rule scheme for induction. This has immediate advantages for developing the metatheory: the proofs of completeness for standard semantics and adequacy (i.e. subsumption of explicit induction) for the infinitary system presented in this paper are simpler and more straightforward. Moreover, it permits a cyclic subsystem, which also subsumes explicit induction, to be defined via a simple syntactic criterion that we call *normality*. The smaller search space of possible proofs further enhances the potential for automation. TC logic seems more expressive in other ways, too. For instance, the transitive closure operator may be applied to arbitrarily complex formulas, not only to collections of atomic formulas (cf. Horn clauses), as in e.g. [4, 6].

We show that the explicit and cyclic TC systems are equivalent under arithmetic, as is the case for LKID [3, 26]. However, there are cases in which the cyclic system for LKID is strictly more expressive than the explicit induction system [2]. To obtain a similar result for TC, the fact that all induction schemes are available poses a serious challenge. For one, the counter-example used in [2] does not serve to show this result holds for TC. If this strong inequivalence indeed holds also for TC, it must be witnessed by a more subtle and complex counter-example. Conversely, it may be that the explicit and cyclic systems do coincide for TC. In either case, this points towards fundamental aspects that require further investigation.

The rest of the paper is organised as follows. In Section 2 we reprise the definition of transitive closure logic and both its standard and Henkin-style semantics. Section 3 presents the existing explicit induction proof system for TC logic, and also our new infinitary proof system. We prove the latter sound and complete for the standard semantics, and also derive cut-admissibility. In Section 4 we compare the expressive power of the infinitary system (and its cyclic subsystem) with the explicit system. Section 5 concludes and examines the remaining open questions for our system as well as future work. Due to lack of space, proofs are omitted but can be found in an extended version [12].

2 Transitive Closure Logic and its Semantics

In this section we review the language of transitive closure logic, and two possible semantics for it: a standard one, and a Henkin-style one. For simplicity of presentation we assume (as is standard practice) a designated equality symbol in the language. We denote by $v[x_1 := a_n, \ldots, x_n := a_n]$ the variant of the assignment v which assigns a_i to x_i for each i, and by $\varphi\left\{\frac{t_1}{x_1}, \ldots, \frac{t_n}{x_n}\right\}$ the result of simultaneously substituting each t_i for the free occurrences of x_i in φ.

▶ **Definition 1** (The language $\mathcal{L}_{\mathsf{RTC}}$)**.** Let σ be a first-order signature with equality, whose terms are ranged over by s and t and predicates by P, and let x, y, z, etc. range over a countable set of variables. The language $\mathcal{L}_{\mathsf{RTC}}$ consists of the formulas defined by the grammar:

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

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

▶ **Definition 2** (Standard Semantics). Let $M = \langle D, I\rangle$ be a first-order structure (i.e. D is a non-empty domain and I an interpretation function), and v an assignment in M which we extend to terms in the obvious way. The satisfaction relation $\models$ between model-valuation pairs $\langle M, v\rangle$ and formulas is defined inductively on the structure of formulas by:

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

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

We next recall the concepts of frames and Henkin structures (see, e.g., [18]). A frame is a first-order structure together with some subset of the powerset of its domain (called its set of admissible subsets).

▶ **Definition 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 $\mathcal{D} \subseteq \wp(D)$.

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

▶ **Definition 4** (Frame Semantics). $\mathcal{L}_{\mathsf{RTC}}$ formulas are interpreted in frames as in Definition 2 above, except for:

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

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

▶ **Definition 5** (Henkin structures). A Henkin structure is a frame $M = \langle D, I, \mathcal{D}\rangle$ such that $\{a \in D \mid M, v[x := a] \models \varphi\} \in \mathcal{D}$ for every φ, and v in M.

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

It is worth noting that the inclusion of equality in the basic language is merely for notational convenience. This is because the *RTC* operator allows us, under both the standard and Henkin semantics, to actually *define* equality $s = t$ on terms as $(RTC_{x,y}\,\bot)(s,t)$.

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

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

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

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

$$\text{(}\exists\text{L):}\ \frac{\Gamma, \varphi \Rightarrow \Delta}{\Gamma, \exists x.\varphi \Rightarrow \Delta}\ x \notin \mathsf{fv}(\Gamma, \Delta) \quad \text{(}\forall\text{L):}\ \frac{\Gamma, \varphi\left\{\frac{t}{x}\right\} \Rightarrow \Delta}{\Gamma, \forall x.\varphi \Rightarrow \Delta} \quad \text{(Cut):}\ \frac{\Gamma \Rightarrow \varphi, \Delta \quad \Sigma, \varphi \Rightarrow \Pi}{\Gamma, \Sigma \Rightarrow \Delta, \Pi}$$

$$\text{(}\exists\text{R):}\ \frac{\Gamma \Rightarrow \varphi\left\{\frac{t}{x}\right\}, \Delta}{\Gamma \Rightarrow \exists x.\varphi, \Delta} \quad \text{(}\forall\text{R):}\ \frac{\Gamma \Rightarrow \varphi, \Delta}{\Gamma \Rightarrow \forall x.\varphi, \Delta}\ x \notin \mathsf{fv}(\Gamma, \Delta) \quad \text{(Subst):}\ \frac{\Gamma \Rightarrow \Delta}{\Gamma\left\{\frac{t_1}{x_1}, \ldots, \frac{t_n}{x_n}\right\} \Rightarrow \Delta\left\{\frac{t_1}{x_1}, \ldots, \frac{t_n}{x_n}\right\}}$$

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

3 Proof Systems for $\mathcal{L}_{\mathsf{RTC}}$

In this section, we define two proof systems for $\mathcal{L}_{\mathsf{RTC}}$. The first is a finitary proof system with an explicit induction rule for *RTC* formulas. The second is an infinitary proof system, in which *RTC* formulas are simply unfolded, and inductive arguments are represented via infinite descent-style constructions. We show the soundness and completeness of these proof systems, and also compare their provability relations.

Our systems for $\mathcal{L}_{\mathsf{RTC}}$ are extensions of $\mathcal{LK}_=$, the sequent calculus for classical first-order logic with equality [16, 28] whose proof rules we show in Fig. 1.[3] Sequents are expressions of the form $\Gamma \Rightarrow \Delta$, for finite sets of formulas Γ and Δ. We write Γ, Δ and Γ, φ as a shorthand for $\Gamma \cup \Delta$ and $\Gamma \cup \{\varphi\}$ respectively, and $\mathsf{fv}(\Gamma)$ for the set of free variables of the formulas in the set Γ. 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.

3.1 The Finitary Proof System

We briefly summarise the finitary proof system for $\mathcal{L}_{\mathsf{RTC}}$. For more details see [10, 11]. We write $\varphi(x_1, \ldots, x_n)$ to emphasise that the formula φ may contain $x_1, \ldots, x_n$ as free variables.

▶ **Definition 6.** The proof system RTC_G for $\mathcal{L}_{\mathsf{RTC}}$ is defined by adding to $\mathcal{LK}_=$ the following inference rules:

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

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

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

$$\frac{\Gamma, \psi(x), \varphi(x,y) \Rightarrow \Delta, \psi\left\{\frac{y}{x}\right\}}{\Gamma, \psi\left\{\frac{s}{x}\right\}, (RTC_{x,y}\,\varphi)(s,t) \Rightarrow \Delta, \psi\left\{\frac{t}{x}\right\}}\ x \notin \mathsf{fv}(\Gamma,\Delta) \text{ and } y \notin \mathsf{fv}(\Gamma,\Delta,\psi) \tag{3}$$

Rule (3) is a generalized induction principle. It states that if an extension of formula ψ is closed under the relation induced by φ, then it is also closed under the reflexive transitive closure of that relation. In the case of arithmetic this rule captures the induction rule of Peano's Arithmetics PA [11].

3.2 Infinitary Proof Systems

We now present our infinitary proof systems for $\mathcal{L}_{\mathsf{RTC}}$ which are based on the principle of infinite descent. This is in contrast to infinite-width proof systems based on a variant of the infinite branching ω-rule [25, 17]. Such systems have been widely investigated and known to be useful for attaining completeness (e.g. for arithmetics). Nonetheless, the infinite ω-rule renders them practically useless for automated reasoning. Since our motivation here is that of effectiveness and automation we opt for a finite system in which we allow infinite-height, non-well-founded proofs.

▶ **Definition 7.** The infinitary proof system $\mathsf{RTC}^{\omega}_{G}$ for $\mathcal{L}_{\mathsf{RTC}}$ is defined like RTC_{G}, but replacing Rule (3) by:

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

where z is fresh, i.e. z 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. [7, §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 15 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 8** (Pre-proofs)**.** An $\mathsf{RTC}^{\omega}_{G}$ *pre-proof* is a possibly infinite (i.e. non-well-founded) derivation tree formed using the inference rules. A *path* in a pre-proof is a possibly infinite sequence of sequents $s_0, s_1, \ldots (, s_n)$ such that s_0 is the root sequent of the proof, and s_{i+1} is a premise of s_i for each $i < n$.

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

▶ **Definition 9** (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 10** (Traces). A *trace* is a (possibly infinite) sequence of *RTC* formulas. We say that a trace $\tau_1, \tau_2, \ldots (, \tau_n)$ *follows* a path $s_1, s_2, \ldots (, s_m)$ in a pre-proof $\mathcal{P}$ if, for some $k \geq 0$, each consecutive pair of formulas (τ_i, τ_{i+1}) is a trace pair for (s_{i+k}, s_{i+k+1}). If (τ_i, τ_{i+1}) is a progressing pair then we say that the trace *progresses* at i, and we say that the trace is *infinitely progressing* if it progresses at infinitely many points.

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

▶ **Definition 11** (Infinite Proofs). A $\mathsf{RTC}^{\omega}_{G}$ *proof* is a pre-proof in which every infinite path is followed by some infinitely progressing trace.

Clearly, we cannot reason effectively about such infinite proofs in general. In order to do so we need to restrict our attention to those proof trees which are finitely representable. These are the *regular* infinite proof trees, which contain only finitely many *distinct* subtrees. They can be specified as systems of recursive equations or, alternatively, as cyclic *graphs* [14]. Note that a given regular infinite proof may have many different graph representations. One possible way of formalizing such proof graphs is as standard proof trees containing open nodes (called buds), to each of which is assigned a syntactically equal internal node of the proof (called a companion). Due to space limitation, we elide a formal definition of cyclic proof graphs (see, e.g., Sect. 7 in [6]) and rely on the reader's basic intuitions.

▶ **Definition 12** (Cyclic Proofs). The cyclic proof system $\mathsf{CRTC}^{\omega}_{G}$ for $\mathcal{L}_{\mathsf{RTC}}$ is the subsystem of $\mathsf{RTC}^{\omega}_{G}$ comprising of all and only the finite and *regular* infinite proofs (i.e. those proofs that can be represented as finite, possibly cyclic, graphs).

Note that it is decidable whether a cyclic pre-proof satisfies the global trace condition, using a construction involving an inclusion between Büchi automata (see, e.g., [4, 26]). However since this requires complementing Büchi automata (a PSPACE procedure), our system cannot be considered a proof system in the Cook-Reckhow sense [13]. Notwithstanding, checking the trace condition for cyclic proofs found in practice is not prohibitive [23, 29].

3.3 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 13** (Soundness and Completeness of RTC_G [9]). *RTC_G is sound for standard semantics, and also sound and complete for Henkin semantics.*

Note that the system RTC_G as presented here does not admit cut elimination. The culprit is the induction rule (3), which does not permute with cut. We may obtain admissibility of cut by using the following alternative formulation of the induction rule which, like the induction rule for LKID, incorporates a cut with the induction formula ψ.

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

For the system with this rule, a simple adaptation of the completeness proof in [9], in the spirit of the corresponding proof for LKID in [6], suffices to obtain cut-free completeness. However, the tradeoff is that the resulting cut-free system no longer has the sub-formula property. In contrast, cut-free proofs in RTC_G do satisfy the sub-formula property, for a generalized notion of a subformula that incorporates substitution instances (as in $\mathcal{LK}_=$).

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.

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 14** (Degree of RTC Formulas)**.** For $\phi \equiv (RTC_{x,y}\,\varphi)(s,t)$, define $\delta_\phi(M,v) = 0$ if $v(s) = v(t)$, and $\delta_\phi(M,v) = n$ if $v(s) \neq v(t)$ and $a_0, \ldots, a_n$ is a minimal-length sequence of elements in the domain of M 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 15** (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 $\delta_{\tau'}(M', v') < \delta_\tau(M, v)$.*

As is standard for infinite descent inference systems [4, 5, 6, 15, 23, 29], 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 16** (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)*

The soundness of the cyclic system is an immediate corollary, since each CRTC_G^ω proof is also a RTC_G^ω proof.

▶ **Corollary 17** (Soundness of CRTC_G^ω)**.** *If there is a CRTC_G^ω proof of $\Gamma \Rightarrow \Delta$, then $\Gamma \Rightarrow \Delta$ is valid (w.r.t. the standard semantics)*

Following a standard technique (as used in e.g. [6]), we can show cut-free completeness of RTC_G^ω with respect to the standard semantics.

▶ **Definition 18** (Schedule)**.** A *schedule element* E is defined as any of the following:

- a formula of the form $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \to \psi$;
- 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 $\langle s = t, x, \varphi, n, \Gamma, \Delta\rangle$ where s and t are terms, x is a variable, φ is a formula, $n \in \{1, 2\}$, and Γ and Δ are finite sequences of formulas.

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 19** (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.

We give the definition of T_{i+1} when E_i is an *RTC* schedule element, i.e. of the form $\langle (RTC_{x,y}\,\varphi)(s,t), r, z, \Gamma, \Delta\rangle$ (the other cases are similar). T_{i+1} is then 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:

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

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

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

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 20** (Limit Sequents). When a search tree T_ω is not an RTC^ω_G 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 the *untraceable branch* of T_ω. We then define $\Gamma_\omega = \bigcup_{i>0} \Gamma_i$ and $\Delta_\omega = \bigcup_{i>0} \Delta_i$, and call $\Gamma_\omega \Rightarrow \Delta_\omega$ the *limit sequent*.[4]

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 their purpose is not to play a role in syntactic proofs, but to induce counter-models as follows.

▶ **Definition 21** (Counter-interpretations). Assume a search tree T_ω which is not a RTC^ω_G proof with limit sequent $\Gamma_\omega \Rightarrow \Delta_\omega$. Let $\sim$ be the smallest congruence relation on terms such that $s \sim t$ whenever $s = t \in \Gamma_\omega$. Define a structure $M_\omega = \langle D, I\rangle$ as follows (where $[t]$ stands for the $\sim$-equivalence class of t):

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

[4] To be rigorous, we may pick e.g. the left-most open node or untraceable branch.

- For every k-ary relation symbol q: $I(q) = \{([t_1], \ldots, [t_k]) \mid q(t_1, \ldots, t_k) \in \Gamma_\omega\}$

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

Counter-interpretations $\langle M_\omega, v_\omega \rangle$ have the following property, meaning that M_ω is a counter-model for the corresponding sequent $\Gamma \Rightarrow \Delta$ if its search tree T_ω is not a proof.

▶ **Lemma 22.** *If $\psi \in \Gamma_\omega$ then $M_\omega, v_\omega \models \psi$; and if $\psi \in \Delta_\omega$ then $M_\omega, v_\omega \not\models \psi$.*

The completeness result therefore follows since, by construction, a sequent S is contained within its corresponding limit sequents. Thus, for any sequent S, if some search tree T_ω contracted for S is not an RTC^ω_G proof then it follows from Lemma 22 that S is not valid (M_ω is a counter model for it). Hence if S is valid, then T_ω is a recursive RTC^ω_G proof for it.

▶ **Theorem 23** (Completeness). RTC^ω_G *is complete for standard semantics.*

We obtain admissibility of cut as the search tree T_ω is cut-free.

▶ **Corollary 24** (Cut admissibility). *Cut is admissible in* RTC^ω_G.

3.4 $\mathcal{L}_{\mathsf{RTC}}$ with Pairs

To obtain the full inductive expressivity we must allow the formation of the transitive closure of not only binary relations, but any $2n$-ary relation. In [1] it was shown that taking such a RTC^n operator for every n (instead of just for $n = 1$) results in a more expressive logic, namely one that captures all finitary first-order definable inductive definitions and relations. Nonetheless, from a proof theoretical point of view having infinitely many such operators is suboptimal. Thus, we here instead incorporate the notion of ordered pairs and use it to encode such operators. For example, writing $\langle x, y \rangle$ for the application of the pairing function $\langle\rangle(x, y)$, the formula $(RTC^2_{x_1,x_2,y_1,y_2}\, \varphi)(s_1, s_2, t_1, t_2)$ can be encoded by:

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

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

The above proof systems are extended to capture ordered pairs as follows.

▶ **Definition 25.** For a signature containing at least one constant c, and a binary function symbol denoted by $\langle\rangle$, the proof systems $\langle\mathsf{RTC}\rangle_G$, $\langle\mathsf{RTC}\rangle^\omega_G$, and $\langle\mathsf{CRTC}\rangle^\omega_G$ are obtained from RTC_G, RTC^ω_G, CRTC^ω_G (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 13 and 23 can easily be extended to obtain the following results for languages with a pairing function. For completeness, the key observation is that the model of 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 26** (Soundness and Completeness of $\langle\mathsf{RTC}\rangle_G$ and $\langle\mathsf{RTC}\rangle^\omega_G$). *The proof systems $\langle\mathsf{RTC}\rangle_G$ and $\langle\mathsf{RTC}\rangle^\omega_G$ are both sound and complete for the admissible forms of Henkin and standard semantics, respectively.*

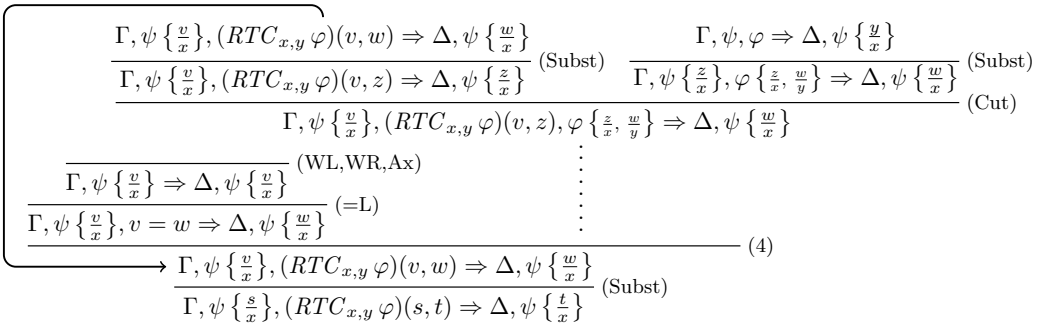

Figure 2 $\mathsf{CRTC}^{\omega}_{G}$ derivation simulating Rule (3). The variables v and w are fresh (i.e. do not occur free in Γ, Δ, φ, or ψ).

4 Relating the Finitary and Infinitary Proof Systems

This section discusses the relation between the explicit and the cyclic system for TC. In Section 4.1 we show that the former is contained in the latter. The converse direction, which is much more subtle, is discussed in Section 4.2.

4.1 Inclusion of RTC_G in $\mathsf{CRTC}^{\omega}_{G}$

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

▶ **Lemma 27.** *Rule (3) is derivable in* $\mathsf{CRTC}^{\omega}_{G}$.

This leads to the following result (an analogue to [6, Thm. 7.6]).

▶ **Theorem 28.** $\mathsf{CRTC}^{\omega}_{G} \supseteq \mathsf{RTC}_G$, *and is thus complete w.r.t. Henkin semantics.*

Lemma 27 is the TC counterpart of [6, Lemma 7.5]. It is interesting to note that the simulation of the explicit LKID induction rule in the cyclic LKID system is rather complex since each predicate has a slightly different explicit induction rule, which depends on the particular productions defining it. Thus, the construction for the cyclic LKID system must take into account the possible forms of arbitrary productions. In contrast, $\mathsf{CRTC}^{\omega}_{G}$ provides a single, uniform way to unfold an *RTC* formula: the construction given in Fig. 2 is the cyclic representation of the *RTC* operator semantics, with the variables v and w implicitly standing for arbitrary terms (that we subsequently substitute for).

This uniform syntactic translation of the explicit RTC_G induction rule into $\mathsf{CRTC}^{\omega}_{G}$ allows us to *syntactically* identify a proper subset of cyclic proofs which is also complete w.r.t. Henkin semantics.[5] 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 (i.e. not identical up to permutation) basic cycles *overlap* if they share any nodes in common, i.e. at

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

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

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

▶ **Theorem 30.** $\mathsf{NCRTC}^{\omega}_{G} \supseteq \mathsf{RTC}_G$.

Henkin-completeness of the normal cyclic system then follows from Theorem 30 and Theorem 13.

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

This section addresses the question of whether the cyclic system is equivalent to the explicit one, or strictly stronger. In [6] it was conjectured that for the system with inductive definitions, LKID and CLKID^{ω} are equivalent. Later, it was shown that they are indeed equivalent when containing arithmetics [3, 26]. We obtain a corresponding theorem in Section 4.2.1 for the TC systems. However, it was also shown in [2] that in the general case the cyclic system is stronger than the explicit one. We discuss the general case for TC and its subtleties in Section 4.2.2.

4.2.1 The Case of Arithmetics

Let $\mathcal{L}_{\mathsf{RTC}}$ be a language based on the signature $\{0, \mathsf{s}, +\}$. Let RTC_G+A and $\mathsf{CRTC}^{\omega}_{G}$+A be the systems for $\mathcal{L}_{\mathsf{RTC}}$ obtained by adding to RTC_G and $\mathsf{CRTC}^{\omega}_{G}$, respectively, the standard axioms of PA together with the RTC-characterization of the natural numbers, i.e.:

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

Note that we do not need to assume multiplication explicitly in the signature, nor do we need to add axioms for it, since multiplication is definable in $\mathcal{L}_{\mathsf{RTC}}$ and its standard axioms are derivable [1, 11].

Recall that we can express facts about sequences of numbers in PA by using a β-function such that for any finite sequence $k_0, k_1, ..., k_n$ there is some c such that for all $i \leq n$, $\beta(c, i) = k_i$. Accordingly, let B be a well-formed formula of the language of PA with three free variables which captures in PA a β-function. For each formula φ of the language of PA define $\varphi^{\beta} := \varphi$, and define $((RTC_{x,y}\,\varphi)(s,t))^{\beta}$ to be:

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

The following result, which was proven in [8, 11], establishes an equivalence between RTC_G+A and PA_G (a Gentzen-style system for PA). It is mainly based on the fact that in RTC_G+A all instances of PA_G induction rule are derivable.

▶ **Theorem 31** (cf. [11]). *The following hold:*
1. $\vdash_{\mathsf{RTC}_G+\mathsf{A}} \varphi \Leftrightarrow \varphi^\beta$.
2. $\vdash_{\mathsf{RTC}_G+\mathsf{A}} \Gamma \Rightarrow \Delta$ *iff* $\vdash_{\mathsf{PA}_G} \Gamma^\beta \Rightarrow \Delta^\beta$.

We show a similar equivalence holds between the cyclic system CRTC^ω_G and CA_G, a cyclic system for arithmetic shown to be equivalent to PA_G [26].

▶ **Theorem 32.** $\vdash_{\mathsf{CRTC}^\omega_G+\mathsf{A}} \Gamma \Rightarrow \Delta$ *iff* $\vdash_{\mathsf{CA}_G} \Gamma^\beta \Rightarrow \Delta^\beta$.

These results allow us to show an equivalence between the finitary and cyclic systems for TC with arithmetic.

▶ **Theorem 33.** $\mathsf{RTC}_G+\mathsf{A}$ *and* $\mathsf{CRTC}^\omega_G+\mathsf{A}$ *are equivalent.*

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

In [3], the equivalence result of [26] was improved to show it holds for any set of inductive predicates containing the natural number predicate N. On the one hand, our result goes beyond that of [3] as it shows the equivalence for systems with a richer notion of inductive definition, due to the expressiveness of TC. On the other hand, TC does not support restricting the set of inductive predicates, i.e. the *RTC* operator may operate on any formula in the language. To obtain a finer result which corresponds to that of [3] we need to further explore the transformations between proofs in the two systems. This is left for future work.

4.2.2 The General Case

As mentioned, the general equivalence conjecture between LKID and CLKID^ω was refuted in [2], by providing a concrete example of a statement which is provable in the cyclic system but not in the explicit one. The statement (called 2-Hydra) involves a predicate encoding a binary version of the 'hydra' induction scheme for natural numbers given in [20], and expresses that every pair of natural numbers is related by the predicate.[6] However, a careful examination of this counter-example reveals that it only refutes a strong form of the conjecture, according to which both systems are based on the same set of productions. In fact, already in [2] it is shown that if the explicit system is extended by another inductive predicate, namely one expressing the $\leq$ relation, then the 2-Hydra counter-example becomes provable. Therefore, the less strict formulation of the question, namely whether for any proof in $\mathsf{CLKID}^\omega_\phi$ there is a proof in $\mathsf{LKID}_{\phi'}$ for some $\phi' \supseteq \phi$, has not yet been resolved. Notice that in TC the equivalence question is of this weaker variety, since the *RTC* operator 'generates' all inductive definitions at once. That is, there is no *a priori* restriction on the inductive predicates one is allowed to use. Indeed, the 2-Hydra counter-example from [2] can be expressed in $\mathcal{L}_{\mathsf{RTC}}$ and proved in CRTC^ω_G. However, this does not produce a counter-example for TC since it is also provable in RTC_G, due to the fact that $s \leq t$ is definable via the *RTC* formula $(RTC_{w,u}\, \mathsf{s}\, w = u)(s, t)$.

Despite our best efforts, we have not yet managed to settle this question, which appears to be harder to resolve in the TC setting. One possible approach to solving it is the semantical

[6] In fact, the falsifying Henkin model constructed in [2] also satisfies the '0-axiom' $(\forall x.0 \neq \mathsf{s}\, x)$, and the '$\mathsf{s}$-axiom' $(\forall x, y.\, \mathsf{s}\, x = \mathsf{s}\, y \to x = y)$ stipulating injectivity of the successor function, and so the actual counter-example to equivalence is the sequent: $(0, \mathsf{s})$-axioms $\Rightarrow$ 2-Hydra.

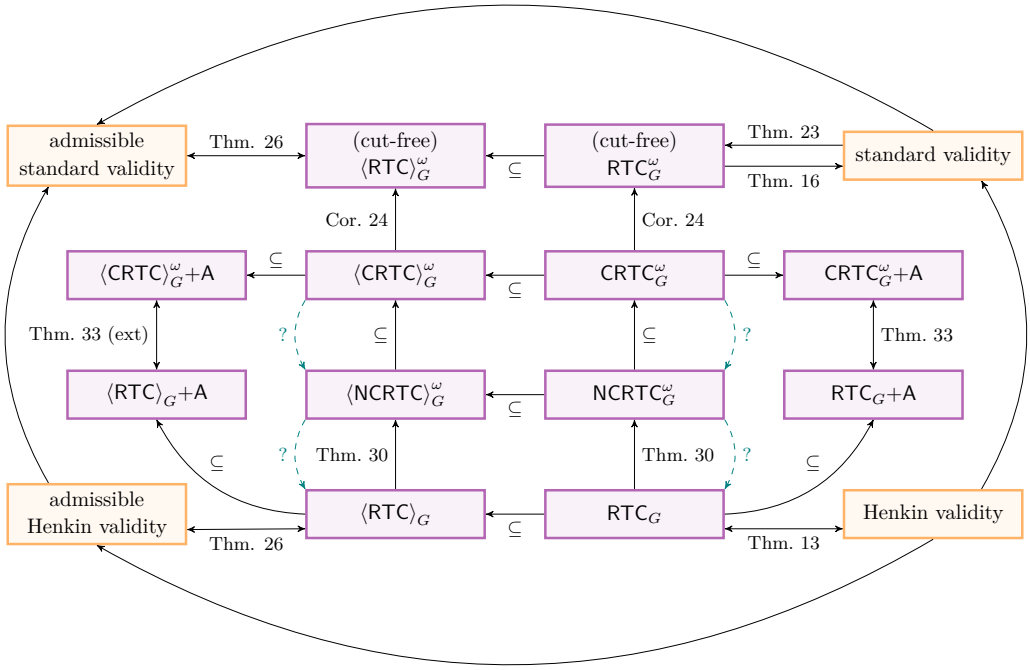

Figure 3 Diagrammatic Summary of our Results.

one, i.e. exploiting the fact that the explicit system is known to be sound w.r.t. Henkin semantics. This is what was done in [2]. Thus, to show strict inclusion one could construct an alternative statement that is provable in $\mathsf{CRTC}^{\omega}_{G}$ whilst also demonstrating a Henkin model for TC that is not a model of the statement. However, constructing a TC Henkin model appears to be non-trivial, due to its rich inductive power. In particular, it is not at all clear whether the structure that underpins the LKID counter-model for 2-Hydra admits a Henkin model for TC. Alternatively, to prove equivalence, one could show that $\mathsf{CRTC}^{\omega}_{G}$ is also sound w.r.t. Henkin semantics. Here, again, proving this does not seem to be straightforward.

In our setting, there is also the question of the inclusion of $\mathsf{CRTC}^{\omega}_{G}$ in $\mathsf{NCRTC}^{\omega}_{G}$, which amounts to the question of whether overlapping cycles can be eliminated. Moreover, we can ask if $\mathsf{NCRTC}^{\omega}_{G}$ is included in RTC_{G}, independently of whether this also holds for $\mathsf{CRTC}^{\omega}_{G}$. Again, the semantic approach described above may prove fruitful in answering these questions.

5 Conclusions and Future Work

We developed a natural infinitary proof system for transitive closure logic which is cut-free complete for the standard semantics and subsumes the explicit system. We further explored its restriction to cyclic proofs which provides the basis for an effective system for automating inductive reasoning. In particular, we syntactically identified a subset of cyclic proofs that is Henkin-complete. A summary of the proof systems we have studied in this paper, and their interrelationships, is shown in Figure 3. Where an edge between systems is labelled with an inclusion $\subseteq$, this signifies that a proof in the source system is already a proof in the destination system.

As mentioned in the introduction, as well as throughout the paper, this research was motivated by other work on systems of inductive definitions, particularly the LKID framework of [6], its infinitary counterpart LKID^{ω}, and its cyclic subsystem CLKID^{ω}. In terms of the expressive power of the underlying logic, TC (assuming pairs) subsumes the inductive

machinery underlying LKID. This is because for any inductive predicate P of LKID, there is an $\mathcal{L}_{\mathsf{RTC}}$ formula ψ such that for every standard admissible structure M for $\mathcal{L}_{\mathsf{RTC}}$, P has the same interpretation as ψ under M. This is due to Thm. 3 in [1] and the fact that the interpretation of P must necessarily be a recursively enumerable set. As for the converse inclusion, for any positive $\mathcal{L}_{\mathsf{RTC}}$ formula there is a production of a corresponding LKID inductive definition. However, the *RTC* operator can also be applied on complex formulas (whereas LKID productions only consider atomic predicates). This indicates that TC might be more expressive. It was noted in [6, p. 1180] that complex formulas may be handled by stratifying the theory of LKID, similar to [21], but the issue of relative expressiveness of the resulting theory is not addressed. While we strongly believe it is the case that TC is strictly more expressive than the logic of LKID, proving so is left for future work. Also left for future research is establishing the comparative status of the corresponding formal proof systems.

In addition to the open question of the (in)equivalence of RTC_G and CRTC^{ω}_G in the general case, discussed in Section 4.2, several other questions and directions for further study naturally arise from the work of this paper. An obvious one would be to implement our cyclic proof system in order to investigate the practicalities of using TC logic to support automated inductive reasoning. More theoretically it is already clear that TC logic, as a framework, diverges from existing systems for inductive reasoning (e.g. LKID) in interesting, non-trivial ways. The uniformity provided by the transitive closure operator may offer a way to better study the relationship between implicit and explicit induction, e.g. in the form of cuts required in each system, or the relative complexity of proofs that each system admits. Moreover, it seems likely that *co*inductive reasoning can also be incorporated into the formal system. Determining whether, and to what extent, these are indeed the case is left for future work.

References

1. Arnon Avron. Transitive Closure and the Mechanization of Mathematics. In F. D. Kamareddine, editor, *Thirty Five Years of Automating Mathematics*, volume 28 of *Applied Logic Series*, pages 149–171. Springer, Netherlands, 2003. doi:10.1007/978-94-017-0253-9_7.
2. Stefano Berardi and Makoto Tatsuta. Classical System of Martin-Löf's Inductive Definitions Is Not Equivalent to Cyclic Proof System. In *Proceedings of FOSSACS, Uppsala, Sweden, April 22–29, 2017*, pages 301–317, Berlin, Heidelberg, 2017. Springer. doi:10.1007/978-3-662-54458-7_18.
3. Stefano Berardi and Makoto Tatsuta. Equivalence of Inductive Definitions and Cyclic Proofs Under Arithmetic. In *Proceedings of LICS, Reykjavik, Iceland, June 20–23, 2017*, pages 1–12, 2017. doi:10.1109/LICS.2017.8005114.
4. James Brotherston. Formalised Inductive Reasoning in the Logic of Bunched Implications. In *Proceedings of SAS, Kongens Lyngby, Denmark, August 22–24, 2007*, pages 87–103, 2007. doi:10.1007/978-3-540-74061-2_6.
5. James Brotherston, Richard Bornat, and Cristiano Calcagno. Cyclic Proofs of Program Termination in Separation Logic. In *Proceedings of POPL, San Francisco, California, USA, January 7–12, 2008*, pages 101–112, 2008. doi:10.1145/1328438.1328453.
6. James Brotherston and Alex Simpson. Sequent Calculi for Induction and Infinite Descent. *Journal of Logic and Computation*, 21(6):1177–1216, 2010.
7. Samuel R. Buss. *Handbook of Proof Theory*. Studies in Logic and the Foundations of Mathematics. Elsevier Science, 1998.
8. Liron Cohen. Ancestral Logic and Equivalent Systems. Master's thesis, Tel-Aviv University, Israel, 2010.

9 Liron Cohen. Completeness for Ancestral Logic via a Computationally-Meaningful Semantics. In *Proceedings of TABLEAUX, Brasília, Brazil, September 25–28, 2017*, pages 247–260, 2017. doi:10.1007/978-3-319-66902-1_15.

10 Liron Cohen and Arnon Avron. Ancestral Logic: A Proof Theoretical Study. In U. Kohlenbach, editor, *Logic, Language, Information, and Computation*, volume 8652 of *Lecture Notes in Computer Science*, pages 137–151. Springer, 2014.

11 Liron Cohen and Arnon Avron. The Middle Ground–Ancestral Logic. *Synthese*, pages 1–23, 2015.

12 Liron Cohen and Reuben N. S. Rowe. Infinitary and Cyclic Proof Systems for Transitive Closure Logic. *CoRR*, abs/1802.00756, 2018. arXiv:1802.00756.

13 Stephen A. Cook and Robert A. Reckhow. The Relative Efficiency of Propositional Proof Systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.

14 Bruno Courcelle. Fundamental Properties of Infinite Trees. *Theor. Comput. Sci.*, 25:95–169, 1983. doi:10.1016/0304-3975(83)90059-2.

15 Anupam Das and Damien Pous. A Cut-Free Cyclic Proof System for Kleene Algebra. In *Proceedings of TABLEAUX, Brasília, Brazil, September 25–28, 2017*, pages 261–277, 2017. doi:10.1007/978-3-319-66902-1_16.

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

17 Jean-Yves Girard. *Proof Theory and Logical Complexity*, volume 1. Humanities Press, 1987.

18 Leon Henkin. Completeness in the Theory of Types. *Journal of Symbolic Logic*, 15(2):81–91, 1950.

19 Ryo Kashima and Keishi Okamoto. General Models and Completeness of First-order Modal μ-calculus. *Journal of Logic and Computation*, 18(4):497–507, 2008.

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

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

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

23 Reuben N. S. Rowe and James Brotherston. Automatic Cyclic Termination Proofs for Recursive Procedures in Separation Logic. In *Proceedings of CPP, Paris, France, January 16–17, 2017*, pages 53–65, 2017. doi:10.1145/3018610.3018623.

24 Luigi Santocanale. A Calculus of Circular Proofs and Its Categorical Semantics. In *Proceedings of FOSSACS, Grenoble, France, April 8–12, 2002*, pages 357–371. Springer Berlin Heidelberg, 2002. doi:10.1007/3-540-45931-6_25.

25 Kurt Schütte. Beweistheoretische Erfassung der unendlichen Induktion in der Zahlentheorie. *Mathematische Annalen*, 122:369–389, 1950/51.

26 Alex Simpson. Cyclic Arithmetic Is Equivalent to Peano Arithmetic. In *Proceedings of FOSSACS, Uppsala, Sweden, April 22–29, 2017*, pages 283–300, 2017. doi:10.1007/978-3-662-54458-7_17.

27 Christoph Sprenger and Mads Dam. On the Structure of Inductive Reasoning: Circular and Tree-Shaped Proofs in the μ-Calculus. In *Proceedings of FOSSACS, Warsaw, Poland, April 7–11, 2003*, pages 425–440. Springer Berlin Heidelberg, 2003. doi:10.1007/3-540-36576-1_27.

28 Gaisi Takeuti. *Proof Theory*. Courier Dover Publications, 1987.

29 Gadi Tellez and James Brotherston. Automatically Verifying Temporal Properties of Pointer Programs with Cyclic Proof. In *Proceedings of CADE, Gothenburg, Sweden, August 6–11, 2017*, pages 491–508, 2017. doi:10.1007/978-3-319-63046-5_30.