Transitive Closure Logic: Infinitary and Cyclic Proof Systems

Reuben N. S. Rowe[1] and Liron Cohen[2]

[1] School of Computing, University of Kent, Canterbury, UK
r.n.s.rowe@kent.ac.uk
[2] Dept. of Computer Science, Cornell University, NY, USA
lironcohen@cornell.edu

Abstract

We present a non-well-founded proof system for Transitive Closure (TC) logic, and also consider its subsystem of cyclic proofs. TC logic is an extension of first-order logic with an operator for forming the transitive closure of (relations induced by) arbitrary formulas, allowing it to capture all first-order definable finitary inductive definitions. While the existing, 'explicit' induction proof system is only sound for a Henkin-style semantics, the 'implicit' infinitary and cyclic systems are sound for the standard semantics. When including arithmetic, provability in the explicit and cyclic systems coincides. This mirrors a similar relationship between explicit induction and cyclic proof systems for Martin-Löf-style inductive definitions. Surprisingly, though, a construction that shows the inequivalence of these systems, with respect to particular sets of Martin-Löf productions, does not appear to work for TC logic, which has all inductive definitions available 'at once'.

1 TC Logic and its Semantics

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, 7, 8]. TC adds to first order logic a single operator for forming binary relations: specifically, the (reflexive and) transitive closures of arbitrary formulas (more precisely, the transitive closure of the binary relation induced by a formula with respect to two distinct variables).

Definition 1 (The language $\mathcal{L}_{\mathsf{RTC}}$)**.** *Let σ be some first-order signature, 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:*

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

Note that in the formula $(RTC_{x,y}\,\varphi)(s,t)$ free occurrences of x and y in φ are bound.

The semantics of $\mathcal{L}_{\mathsf{RTC}}$ are defined by extending those of standard first-order logic.

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 (extended to terms in the standard way). We define the satisfaction relation $\models$ for $\mathcal{L}_{\mathsf{RTC}}$ in the usual way for the standard first-order components, and for RTC formulas as:*

$$M, v \models (RTC_{x,y}\,\varphi)(s,t) \Leftrightarrow v(s) = v(t) \vee \exists a_0, \ldots, a_n \in D. v(s) = a_0 \wedge v(t) = a_n \wedge \forall i < n. M, v[x := a_i, y := a_{i+1}] \models \varphi$$

We also give a Henkin-style semantics for $\mathcal{L}_{\mathsf{RTC}}$, by considering *frames* $\langle D, I, \mathcal{D}\rangle$, which are first-order structures $\langle D, I\rangle$ together with some subset $\mathcal{D} \subseteq 2^D$ of the powerset of its domain (called its set of admissible subsets).

Definition 3 (Frame Semantics)**.** *We define the satisfaction relation* $\models_{\mathcal{H}}$ *for* $\mathcal{L}_{\mathsf{RTC}}$ *in the usual way for the standard first-order components, and for RTC formulas as follows, where* $M = \langle D, I, \mathcal{D}\rangle$ *is a frame:*

$$M, v \models_{\mathcal{H}} (RTC_{x,y}\,\varphi)(s,t) \Leftrightarrow \text{for every } A \in \mathcal{D}, \text{ if } v(s) \in A \text{ and } (\forall a, b \in D. a \in A \wedge M, v[x := a, y := b] \models_{\mathcal{H}} \varphi \to b \in A), \text{ then } v(t) \in A$$

Henkin *structures* are frames whose set of admissible subsets is closed under definability.

Definition 4 (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.

The modest addition of the *RTC* operator 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 the logic; instead, all induction schemes are available within a single, unified language.

TC logic is intermediate between first- and second-order logic. Furthermore, the *RTC* operator is a particular instance of a (least) fixed point operator. Thus TC logic is also subsumed by fixed-point logics such as the first-order μ-calculus [11]. However its appeal lies in the fact that despite its minimality it retains enough expressivity to capture inductive reasoning, as well as to subsume Peano arithmetic.

2 Proof Systems for TC

Since 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 the Henkin-semantics defined above [6]. This proof system, RTC_G, is obtained by adding the following rules to Gentzen's sequent calculus $\mathcal{LK}$ for classical logic with equality and substitution.

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

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

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

Rule (3) is a generalized induction principle. It states that if the extension of 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 (see [8]).

Following similar developments in other formalizations for fixed point logics and inductive reasoning (see e.g. [3, 4, 5, 10, 13, 14, 16, 17]), we have developed 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, whose soundness 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 *RTC* formulas.

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 $(RTC_{x,y}\,\varphi)(s,t)$.

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 $i < n$. We track *RTC* formulas through a pre-proof, allowing to formalize inductive arguments via infinite descent. If τ and τ' are *RTC* formulas occurring in the left-hand side of the conclusion s and a premise s', respectively, of an inference rule then (τ, τ') is said to be a *trace pair* for (s, s') if the rule is:

- the substitution rule, and $\tau = \tau'\theta$ where θ is the associated substitution;
- the equality rule, and $\tau' = \tau''\left\{\frac{t}{x}, \frac{s}{y}\right\}$ where $s = t$ is the equality introduced by the rule and τ'' is such that $\tau = \tau''\left\{\frac{s}{x}, \frac{t}{y}\right\}$;
- Rule (4) and either: 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*); or $\tau = \tau'$.
- any other rule, and $\tau = \tau'$.

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 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 5 (Infinite Proofs)**.** *An* $\mathsf{RTC}^{\omega}_{G}$ 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.

Definition 6 (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., [3, 15]).

Infinitary proof theories such as this 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 sometimes stronger than the goal one aims to prove. However, in implicit systems the inductive hypotheses may be encoded in the cycles of a proof, so cyclic proof systems seem better for automation. The TC framework takes us another step further: the inductive *schemes* themselves (i.e. inductive definitions) do not have to be chosen *a priori* since they are constructed using the *RTC* operator.

3 Results

The finitary and infinitary proof systems are sound and complete.

Theorem 7 (Soundness and Completeness)**.** RTC_G *is sound for standard semantics, and sound and complete for Henkin semantics.* RTC^ω_G *is sound and (cut-free) complete for standard semantics.*

Rule (3) is derivable in CRTC^ω_G, which leads to the following result.

Theorem 8. $\mathsf{CRTC}^\omega_G \supseteq \mathsf{RTC}_G$*, and is thus complete w.r.t. Henkin semantics.*

We can *syntactically* identify a proper subset of cyclic proofs which is also complete w.r.t. Henkin semantics. The criterion we use is based on the notion of *overlapping cycles*.

Theorem 9. *Let* NCRTC^ω_G *be the subsystem of* RTC^ω_G *comprising of all and only the cyclic proofs containing no overlapping cycles. Then,* $\mathsf{NCRTC}^\omega_G \supseteq \mathsf{RTC}_G$*.*

The restriction to non-overlapping proofs has a potential advantage for automation, since one has only to search for cycles in one single branch.

The cyclic and explicit induction systems are equivalent under arithmetic. We define the systems $\mathsf{RTC}_G{+}\mathsf{A}$ and $\mathsf{CRTC}^\omega_G{+}\mathsf{A}$ for $\mathcal{L}_{\mathsf{RTC}}$ by adding to RTC_G and CRTC^ω_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) $\Rightarrow x + \mathsf{s}\,y = \mathsf{s}\,(x + y)$ and (v) $\Rightarrow (RTC_{w,u}\,\mathsf{s}\,w = u)(0, x)$.

Theorem 10. $\mathsf{RTC}_G{+}\mathsf{A}$ *and* $\mathsf{CRTC}^\omega_G{+}\mathsf{A}$ *are equivalent.*

Our results largely mirror known results for finitary (LKID), infinitary (LKID^ω), and cyclic (CLKID^ω) proof systems for first-order logic with Martin-Löf style inductive definitions. The point at which they differ is in considering the equivalence of RTC_G and CRTC^ω_G in general.

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 [12], and expresses that every pair of natural numbers is related by the predicate. 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. 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.

A full technical report of this extended abstract is available [9].

Acknowledgements

Reuben Rowe was supported by EPSRC Grant No. EP/N028759/1. Liron Cohen was supported by: Fulbright Post-doctoral Scholar program; Weizmann Institute of Science National Postdoctoral Award program for Advancing Women in Science; Eric and Wendy Schmidt Postdoctoral Award program for Women in Mathematical and Computing Sciences.

References

[1] A. 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.

[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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

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

[9] Liron Cohen and Reuben N. S. Rowe. Infinitary and Cyclic Proof Systems for Transitive Closure Logic. *CoRR*, abs/1802.00756, 2018.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] Alex Simpson. Cyclic Arithmetic Is Equivalent to Peano Arithmetic. In *Proceedings of FOSSACS, Uppsala, Sweden, April 22–29, 2017*, pages 283–300, 2017.

[16] 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.

[17] 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.