Uniform Coinductive Proof Search for Horn Clauses

Henning Basold [1] Ekaterina Komendantskaya [2] Yue Li [2]

[1]ENS Lyon, France [2]Heriot-Watt University, UK

Motivation

Infinite proof-trees with respect to Horn-clause assumptions can be categorized according to whether or not infinite terms are involved, and additionally, whether or not the proof-tree is regular (A *regular* structure has a finite number of distinct sub-structures, otherwise it is *irregular*). Table 1 points to examples.

	Involving Infinite Terms	No Infinite Term
Regular Proof Tree	Example (A)	Example (B)
Irregular Proof Tree	Example (C)	Example (D)

Table 1: Categorising non-terminating proof construction with Horn clauses.

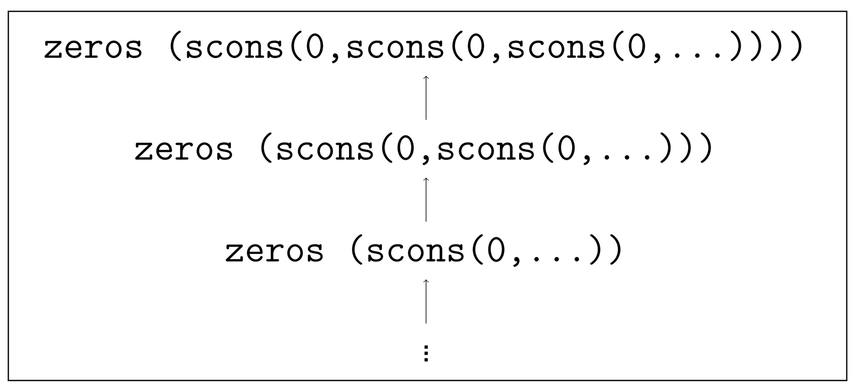
Sound logics have been proposed previously to capture infinite proof-trees: Example A and B can be captured using CoLP (Simon et al 2006) whilst Example B and D can be captured using PC-resolution (PC for Proof-relevant Co-recursive, Fu et al 2016). No logic exists to capture Example C. We propose such a logic that unifies CoLP and PC-resolution and additionally captures proof-trees like Example C. Our logic is like Cyclic Proof (Brotherston and Simpson 2011) but for coinduction rather than induction. Our approach to coinduction is different from that of μMALL (Baelde 2012), but is justifiable constructively by using Tarski's fixed-point theorem. We published the work as (Basold et al 2019).

Example (A)

The clause (1)

$$\texttt{zeros (X)} \implies \texttt{zeros (scons(0, X))} \qquad (1)$$

says that if `X` is a stream of zeros, then `scons(0,X)` is a stream of zeros. It gives rise to an infinite proof-tree (shown below) that involves the infinite term `scons(0,scons(0,scons(0,...)))` and is regular — all sub-trees are identical.

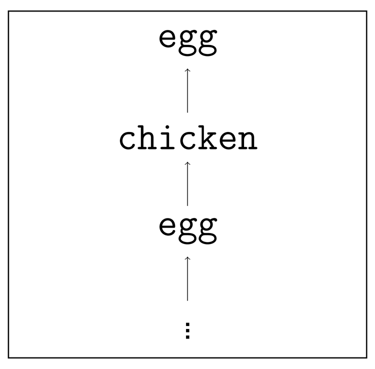


Example (B)

Clauses (2) and (3)

$$\texttt{chicken} \implies \texttt{egg} \qquad (2)$$
$$\texttt{egg} \implies \texttt{chicken} \qquad (3)$$

give rise to a proof-tree that has two distinct sub-trees (hence a regular tree) but does not involve any infinite term.

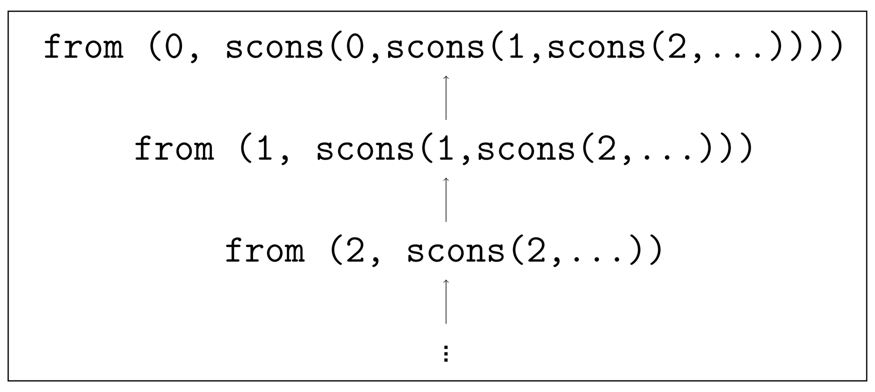


Example (C)

Clause (4)

$$\texttt{from (s (X), Y)} \implies \texttt{from (X, scons(X, Y))} \qquad (4)$$

says that if `Y` is the ordered stream of all natural numbers greater than or equal to `s (X)` (which denotes the successor of `X`, i.e., `X+1`), then `scons(X, Y)` is the ordered stream of all natural numbers greater than or equal to `X`. Using `1,2,...` to denote `s (0)`, `s (s (0))`, `...` we have the following proof-tree that both involves infinite terms and is irregular.

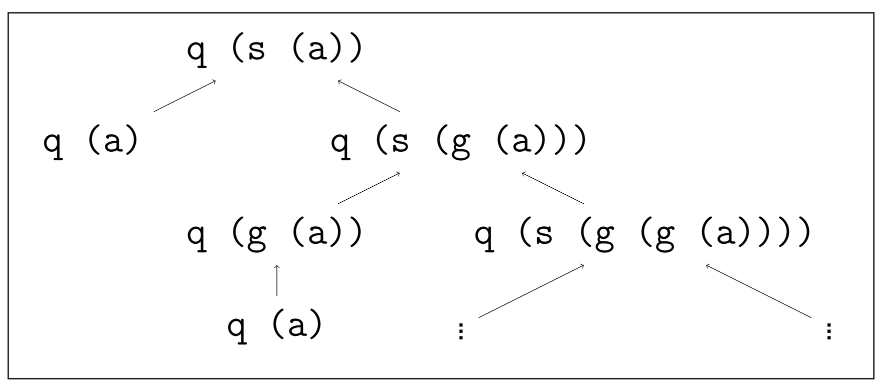


Example (D)

Clauses (5), (6) and (7)

$$\texttt{q (X)} \wedge \texttt{q (s (g (X)))} \implies \texttt{q (s (X))} \qquad (5)$$
$$\texttt{q (X)} \implies \texttt{q (g (X))} \qquad (6)$$
$$\texttt{q (a)} \qquad (7)$$

give an infinite proof-tree that has an infinite amount of distinct sub-trees (hence an irregular tree) but does not involve infinite terms.



Technical Background

We extend the standard notion of a first-order Horn clause to allow infinite terms that have a finite number of free variables, and call the resulting expression a *Horn$^\omega$ clause*.

Variable `X, Y, Z, ...`

Constant `a, b, c, f, g, h, ...`

The *arity* of a symbol is the number of its arguments. We assume that all variables have arity 0, whilst some constants, in particular, those denoted by `a, b, c` have arity 0.

μ-term (ν-term) refers to the smallest (respectively, largest) set T that satisfies the following rules:

- All variables are in T.
- All constants of arity 0 are in T.
- If g is a constant of arity $n > 0$ and $t_1, \ldots, t_n \in T$, then $g(t_1, \ldots, t_n) \in T$.

ω-term refers to those ν-terms that have a finite number of distinct variables.

Predicate `p, q, r, ...`

Atom If a predicate p has arity $n \geq 0$ and $t_1, \ldots, t_n$ are μ-terms (ω-terms) then $p(t_1, \ldots, t_n)$ is a μ-atom (respectively ω-atom).

Horn/Horn$^\omega$ Clause Let $A_0, A_1, \ldots, A_n$ $(n \geq 0)$ be ω-atoms (μ-atoms) then $A_1 \wedge \cdots \wedge A_n \implies A_0$ is a Horn$^\omega$clause (respectively, Horn clause).

Variables in a Horn$^\omega$ clause are regarded as being quantified by $\forall$. Standard least and greatest fixed-point semantics for Horn clauses can be extended for Horn$^\omega$ clauses.

Our Contribution: Coinductive Uniform Proof (CUP)

We extend Uniform Proof (Figure 1a) with a simple rule *co-fix* for coinduction (Figure 1c). The side conditions for using co-fix are formulated as auxiliary rules (Figure 1b).

$$\frac{c, \Sigma; P; \Delta \implies G\,[c/x] \quad c \notin \Sigma}{\Sigma; P; \Delta \implies \forall x.\, G}\,\forall R$$

$$\frac{\Sigma; P, D; \Delta \implies G}{\Sigma; P; \Delta \implies D \to G}\,\to R$$

$$\frac{\Sigma; P; \Delta \implies G_1 \quad \Sigma; P; \Delta \implies G_2}{\Sigma; P; \Delta \implies G_1 \wedge G_2}\,\wedge R$$

$$\frac{\Sigma; P; \Delta \overset{D}{\Rightarrow} A \quad D \in P \cup \Delta}{\Sigma; P; \Delta \implies A}\,\mathrm{DECIDE}$$

$$\frac{\Sigma; P; \Delta \xrightarrow{D[N/x]} A}{\Sigma; P; \Delta \xrightarrow{\forall x.\, D} A}\,\forall L$$

$$\frac{\Sigma; P; \Delta \overset{D}{\Rightarrow} A \quad \Sigma; P; \Delta \implies G}{\Sigma; P; \Delta \xrightarrow{G \to D} A}\,\to L$$

$$\frac{\Sigma; P; \Delta \overset{D_x}{\Rightarrow} A \quad x \in \{1, 2\}}{\Sigma; P; \Delta \xrightarrow{D_1 \wedge D_2} A}\,\wedge L$$

$$\frac{A \equiv A'}{\Sigma; P; \Delta \overset{A'}{\Rightarrow} A}\,\mathrm{INITIAL}$$

(a) The *Uniform Proof* Subsystem of CUP.

$$\frac{c, \Sigma; P; \Delta \implies \langle G\,[c/x] \rangle \quad c \notin \Sigma}{\Sigma; P; \Delta \implies \langle \forall x.\, G \rangle}\,\forall R\langle\rangle$$

$$\frac{\Sigma; P, D; \Delta \implies \langle G \rangle}{\Sigma; P; \Delta \implies \langle D \to G \rangle}\,\to R\langle\rangle$$

$$\frac{\Sigma; P; \Delta \implies \langle G_1 \rangle \quad \Sigma; P; \Delta \implies \langle G_2 \rangle}{\Sigma; P; \Delta \implies \langle G_1 \wedge G_2 \rangle}\,\wedge R\langle\rangle$$

$$\frac{\Sigma; P; \Delta \overset{D}{\Rightarrow} A \quad P \ni D \notin \Delta}{\Sigma; P; \Delta \implies \langle A \rangle}\,\mathrm{DECIDE}\langle\rangle$$

(b) The *Guarded Uniform Proof* Subsystem of CUP.

$$\frac{\Sigma; P; \varphi \implies \langle \varphi \rangle}{\Sigma; P \looparrowright \varphi}\,\mathrm{CO\text{-}FIX}$$

(c) The *Coinduction* Subsystem of CUP.

Figure 1: The *Coinductive Uniform Proof* (CUP) System.

Note that for typesetting reasons, full reference to all literatures cited in *this* section are listed at the end of this section in the block "Section Reference".

Uniform Proof (Miller et al 1991) is for goal directed search. The co-fix rule is inspired by (Giménez 1998) so it resembles the `cofix` tactic of Coq (Coq 2019). P is a finite set of Horn$^\omega$ assumptions. Δ is the (singleton) set of *coinductive hypothesis* (CH) that copies the initial goal φ — a conjunction of Horn$^\omega$ clauses. In a typical CUP session, the co-fix rule is used first and for *once*, and then Uniform Proof rules are used with care taken not to use the CH in an unsound way: this is helped by the auxiliary rules and the mark $\langle\cdot\rangle$. A goal provable in CUP is sound with respect to the greatest fixed-point model of P.

Section Reference

(Miller et al 1991) Dale Miller, Gopalan Nadathur, Frank Pfenning, Andre Scedrov, Uniform proofs as a foundation for logic programming, Annals of Pure and Applied Logic, Volume 51, Issues 12, 1991, Pages 125-157, ISSN 0168-0072

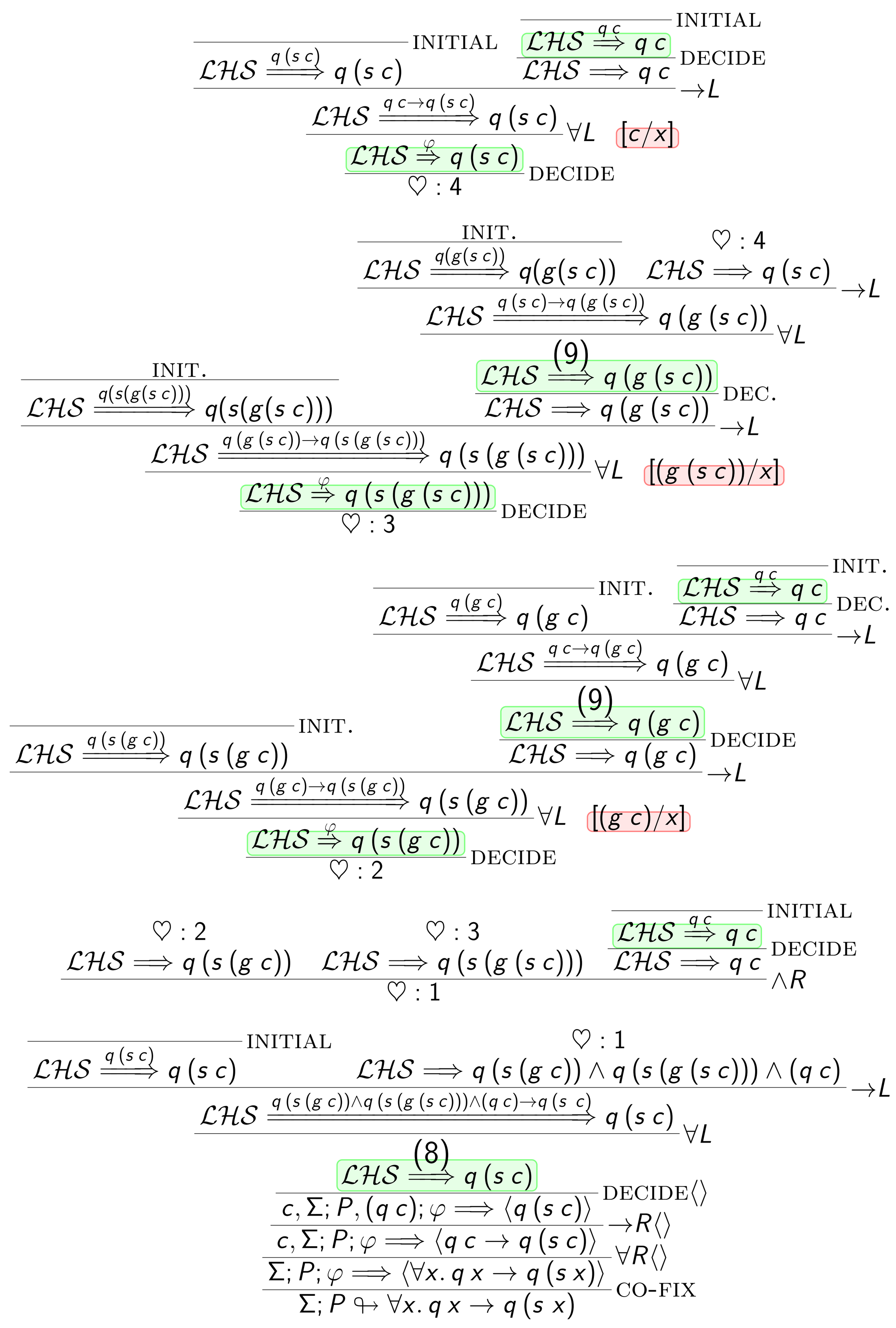
(Giménez 1998) Giménez E. (1998) Structural recursive definitions in type theory. In: Larsen K.G., Skyum S., Winskel G. (eds) Automata, Languages and Programming. ICALP 1998. Lecture Notes in Computer Science, vol 1443. Springer, Berlin, Heidelberg

(Coq 2019) `https://coq.inria.fr/`

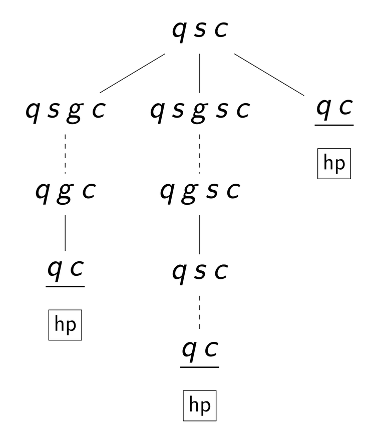
Why CUP works? Explained by an Example

$$\forall x.\, q\,(s\,(g\,x)) \wedge q\,(s\,(g\,(s\,x))) \wedge (q\,x) \implies q\,(s\,x) \qquad (8)$$
$$\forall x.\, q\,x \implies q\,(g\,x) \qquad (9)$$
$$q\,z \qquad (10)$$
$$\forall x.\, q\,x \implies q\,(s\,x) \qquad (11)$$

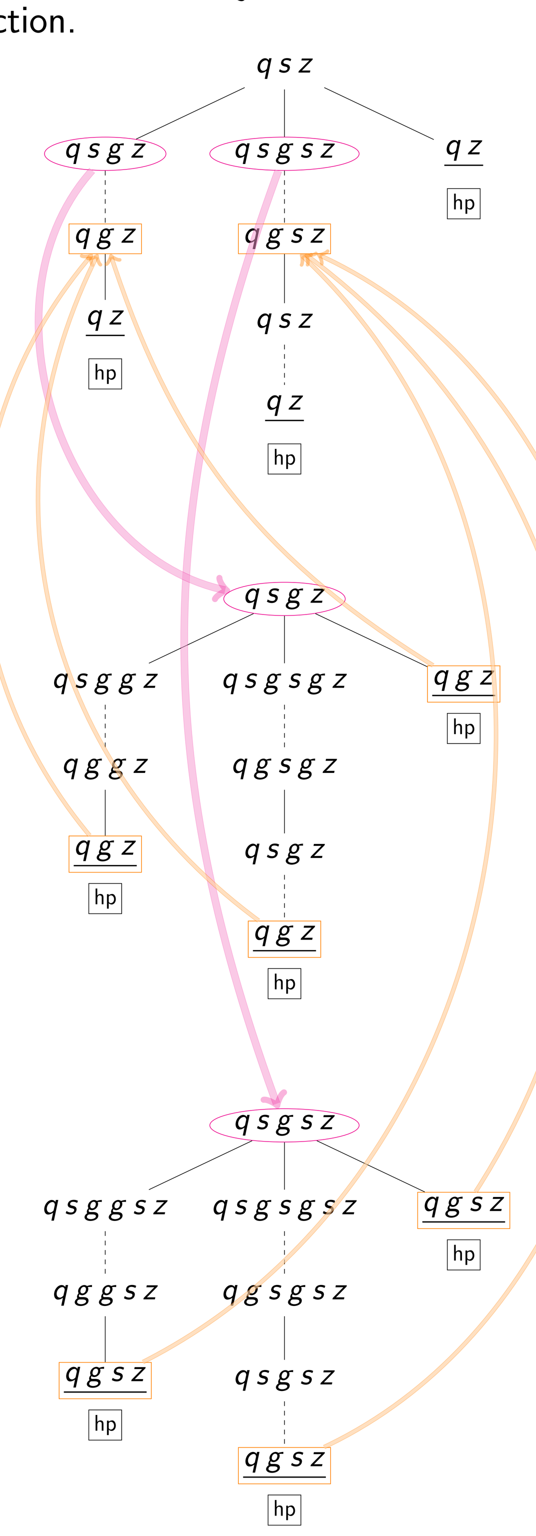
Let P contain (8), (9) and (10). We prove (11) in CUP. Note that in these clauses $\forall$ is written explicitly. Read bottom-up:



The diagram below captures the essence of the construction, and we call it a *simplified CUP-tree* (sCUP-tree). It is based on the principal nodes highlighted above in green.



Systematically instantiating the sCUP-tree by a system of substitutions built by composing those associated with using CH (highlighted above in red), we observe the following fragment from an infinite interconnection scheme that justifies the coinductive soundness of the CUP construction.



Limitation of CUP: consider `clock(s(Y), 0)` $\implies$ `clock(0, Y)` and `clock(X, s(Y))` $\implies$ `clock(s(X), Y)`.

Reference

(Simon et al 2006) Simon L., Mallya A., Bansal A., Gupta G. (2006) Coinductive Logic Programming. In: Etalle S., Truszczyski M. (eds) Logic Programming. ICLP 2006. Lecture Notes in Computer Science, vol 4079. Springer, Berlin, Heidelberg

(Fu et al 2016) Fu P., Komendantskaya E., Schrijvers T., Pond A. (2016) Proof Relevant Corecursive Resolution. In: Kiselyov O., King A. (eds) Functional and Logic Programming. FLOPS 2016. Lecture Notes in Computer Science, vol 9613. Springer, Cham

(Brotherston and Simpson 2011) Brotherston J., Simpson A. (2011) Sequent calculi for induction and infinite descent. In: Journal of Logic and Computation, Volume 21, Issue 6, Pages 1177-1216

(Baelde 2012) Baelde D. (2012) Least and Greatest Fixed Points in Linear Logic. In: ACM Trans. Comput. Logic 13, 1, Article 2, 44 pages.

(Basold et al 2019) Basold H., Komendantskaya E., Li Y. (2019) Coinduction in Uniform: Foundations for Corecursive Proof Search with Horn Clauses. In: Caires L. (eds) Programming Languages and Systems. ESOP 2019. Lecture Notes in Computer Science, vol 11423. Springer, Cham