Realizability in Cyclic Proof: Extracting Ordering Information for Infinite Descent

Reuben N. S. Rowe[1] and James Brotherston[2]

[1] School of Computing, University of Kent, Canterbury, UK
[2] Dept. of Computer Science, University College London, UK

Abstract. In program verification, measures for proving the termination of programs are typically constructed using (notions of size for) the data manipulated by the program. Such data are often described by means of logical formulas. For example, the cyclic proof technique makes use of semantic approximations of inductively defined predicates to construct Fermat-style infinite descent arguments. However, logical formulas must often incorporate explicit size information (e.g. a list length parameter) in order to support inter-procedural analysis.
In this paper, we show that information relating the sizes of inductively defined data can be automatically extracted from *cyclic* proofs of logical entailments. We characterise this information in terms of a graph-theoretic condition on proofs, and show that this condition can be encoded as a containment between weighted automata. We also show that under certain conditions this containment falls within known decidability results. Our results can be viewed as a form of *realizability* for cyclic proof theory.

Keywords: Approximation semantics · Cyclic proof · Entailment · Inductive predicates · Infinite Descent · Realizability · Sequent calculus · Weighted automata

1 Introduction

In program verification, it is well known that proving *termination* of a particular program depends on identifying a well-founded measure that decreases monotonically during the program's execution. Thus, since the measure cannot decrease infinitely often, no execution of the program can be infinite. In practice, termination measures are typically derived from the data manipulated by the program itself (cf. *size-change termination* [14]), and in particular from notions of the *size* of its data structures.

For example, consider the following code, which "shuffles" a linked list with head pointer `x`, using an auxiliary list reversal procedure `rev`:

```
proc shuffle(*x) { if (x != nil) { y := [x]; rev(y); shuffle(y); } }
```

where the syntax `[x]` denotes pointer dereferencing. The termination of the `shuffle(x)` procedure can be deduced by taking as termination measure the length of the list from `x`. The call to `rev` and the recursive call to `shuffle` both

take place on the pointer `y` to the tail of the list. However, we also crucially rely upon the fact that the reversal procedure `rev` *does not increase* the size of the list. In a Hoare-style verification, this information is needed when we employ the sequential composition rule:

$$\frac{\{P\}\,\texttt{rev(y)}\,\{Q\} \quad \{Q\}\,\texttt{shuffle(y)}\,\{R\}}{\{P\}\,\texttt{rev(y);shuffle(y)}\,\{R\}}$$

Here, the information that `rev` maintains the size of the list must be reflected in the relationship between its pre- and postconditions P and Q (which are logical formulas). Typically, this must be done by endowing these formulas with explicit size information; e.g., we could write an *inductive predicate* $\mathsf{list}(y, n)$ representing linked lists in memory, with an explicit length parameter (cf. [3]).

In this paper, we show that this kind of information, relating the sizes of inductively defined data, can often be extracted automatically from *cyclic proofs* of logical entailments. Cyclic proofs can be seen as formalising proof by regular infinite descent [7]; they are derivation trees with "backlinks" from (some) leaves to interior nodes, subject to a global soundness condition ensuring that all infinite paths correspond to sound infinite descent arguments. Cyclic proof systems have been developed for a wide variety of settings ranging from pure logic [4, 5] to Hoare-style logics for program termination [6, 17] and other temporal properties [9]; the common denominator is the presence of logical data defined using fixed points. The soundness of cyclic proofs relies on infinite descent over the semantic *approximations* of these fixed points, which can be seen as capturing a notion of size for the corresponding data. Suitable entailments for which to construct these cyclic proofs may be formulated by procedures for verifying the correctness of (fragments of) programs. For example, a procedure to verify the Hoare triple $\{\mathsf{list}(y)\}\,\texttt{rev(y)}\,\{\mathsf{list}(y)\}$ might result in the entailment $y \mapsto x * \mathsf{list}(x) \vdash \mathsf{list}(y)$ of *separation logic* [11, 16]. Such entailments are commonly referred to as verification conditions, since they must be discharged independently.

Relationships between the sizes of inductive data are reflected by *inclusions* between the approximations of the fixed point semantics. To infer these inclusions, we formulate a novel condition on the structure of cyclic entailment proofs (Def. 8) which is sufficient to extract this information (Thm 2). This condition is equivalent to an inclusion between weighted automata that can be constructed from the cyclic proofs (Thm 3), and, when the cyclic proof is suitably structurally well-behaved, this inclusion becomes decidable (Thm 4). For simplicity, we present our results for the well-known cyclic proof system CLKID$^\omega$ for first order logic with inductive definitions [4, 7]. However, we stress that our results are not limited to this setting: in a separate technical report we formulate and prove our results for a general, abstract notion of cyclic proof [18]. Consequently our results also hold, e.g. for separation logic with inductive predicates [5, 6], and so can be deployed in our cyclic proof framework for proving program termination based on this logic [17].

The remainder of this paper is structured as follows. First, Section 2 gives an introductory example motivating our new structural condition for extracting size

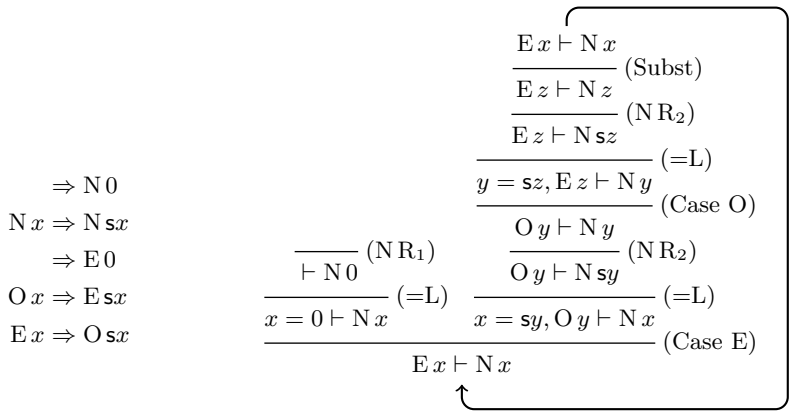


Fig. 1. Inductive definitions of N, E, and O, and cyclic proof of $\mathrm{E}\,x \vdash \mathrm{N}\,x$.

relationships from cyclic proofs. Section 3 then reprises the basics of first-order logic with inductive predicates and its cyclic proof system CLKID$^\omega$ from [4, 7]. In Section 4 we formulate our structural condition on cyclic proofs and prove its soundness. In Section 5 we show how this condition can be encoded as an inclusion between weighted automata and formulate further graph-theoretic conditions on cyclic proofs under which this is decidable. Section 6 concludes.

For space reasons, we elide the detailed proofs of the results in this paper, but they can be found in our longer technical report [18].

2 Motivating Example

Figure 1 gives inductive definitions of predicates N, E and O (intended to capture the properties of being a natural number, even number and odd number respectively) and a cyclic proof of the sequent $\mathrm{E}\,x \vdash \mathrm{N}\,x$. Note that E and O are mutually defined. The $(\mathrm{N}\,\mathrm{R}_i)$ rules indicate a right-unfolding of the N predicate, and the (Case E) and (Case O) rules a left unfolding (or *case analysis*) on the predicates E and O respectively. This cyclic proof is sound since its only infinite path contains an infinite, unbroken "trace" of the E and O predicates in the antecedent of each sequent that "progresses" infinitely often as these predicates are unfolded.

This condition ensures that the proof is valid because it can be related to *approximations* of the semantics $[\![\cdot]\!]$ of the predicates, which form an ordinal-indexed chain $[\![\cdot]\!]_0 \leq [\![\cdot]\!]_1 \leq \ldots \leq [\![\cdot]\!]_\alpha \leq \ldots \leq [\![\cdot]\!]$. If $\mathrm{E}\,x \vdash \mathrm{N}\,x$ is invalid then, by local soundness of the rules, so is every sequent on the infinite path in the proof. The trace along this path then corresponds to a non-increasing subsequence of the ordinals in this chain, which strictly decreases when the trace progresses. Since the trace progresses infinitely often, we obtain an infinitely decreasing chain of ordinals, which is a contradiction.

Interestingly, it turns out that, by examining the structure of this cyclic proof more closely and also considering the (right) unfoldings of N, we can deduce that the α^{th} approximation of E is also included in the α^{th} approximation of N, i.e., $[\![\mathrm{E}\, x]\!]_\alpha \subseteq [\![\mathrm{N}\, x]\!]_\alpha$. Intuitively, this is because on every maximally finite path in the proof along which N is unfolded, the mutually defined E and O are together unfolded at least as often as N. Thus when x is included in some approximation of E, it is already included in the corresponding approximation of N. Later, in Section 4, we will formalise this intuition as an additional syntactic, trace-based condition on cyclic proofs. The upshot is that we may form "traces", as described above, between instances of $\mathrm{E}\, t$ and $\mathrm{N}\, t$ (for any term t) in the antecedent of sequents, even though they are not related by their inductive definitions.

3 Cyclic Proofs for First Order Logic

In this section we summarise a variant of CLKID$^\omega$, a cyclic proof system for first order logic with inductive predicates [4, 7].

3.1 First Order Logic with Inductive Definitions

We assume the standard syntax and semantics of first order logic. For simplicity, we take models to be valuations of term variables to objects in the semantic domain. A *sequent* $\Gamma \vdash \Delta$ comprises two sequences of formulas: an *antecedent* Γ and a *consequent* Δ. For a sequent $S = \Gamma \vdash \Delta$, we write $m \models S$ to mean that the model m satisfies at least one formula in Δ whenever it satisfies all fomulas in Γ. Conversely, we write $m \not\models S$ to mean that m satisfies all fomulas in Γ and *no* formula in Δ. A sequent S is *valid* when $m \models S$ for all models m.

We give the semantics of predicate symbols in the signature by means of sets of inductive *productions*, in the style of Martin-Löf [15].

Definition 1 (Inductive Definition Set). *An* inductive definition set Φ *is a finite set of productions, each of the form* $\mathrm{P}_1\, \boldsymbol{t_1}, \ldots, \mathrm{P}_j\, \boldsymbol{t_j} \Rightarrow \mathrm{P}_0\, \boldsymbol{t_0}$*, consisting of a finite set of predicate formulas called premises and a predicate formula called the conclusion. We say that* $\mathrm{P}_1\, \boldsymbol{t_1}, \ldots, \mathrm{P}_j\, \boldsymbol{t_j} \Rightarrow \mathrm{P}_0\, \boldsymbol{t_0}$ *is a production* for P_0.

Predicate *interpretations* X are functions from predicate formulas to sets of models. We write $[\![\mathrm{P}\, \boldsymbol{t}]\!]_X$ to denote $X(\mathrm{P}\, \boldsymbol{t})$. An inductive definition set Φ induces a *characteristic operator* φ_Φ on predicate interpretations, which applies (substitution instances of) the productions in Φ, as follows (where θ is a substition of terms for variables):

$$\varphi_\Phi(X)(\mathrm{P}\, \boldsymbol{t}\theta) = \{m \mid \mathrm{P}_1\, \boldsymbol{t_1}, \ldots, \mathrm{P}_j\, \boldsymbol{t_j} \Rightarrow \mathrm{P}\, \boldsymbol{t} \in \Phi \\ \text{and } m \in [\![\mathrm{P}_i\, \boldsymbol{t_i}\theta]\!]_X \text{ for all } i \in \{1, \ldots, j\}\}$$

We define a partial ordering $\leq$ on the set of predicate interpretations $\mathcal{I}$ by $X \leq X' \Leftrightarrow \forall F.\ X(F) \subseteq X'(F)$. One can note that $(\mathcal{I}, \leq)$ is a complete lattice

and the least element, denoted by $X_\bot$, maps all predicate formulas to the empty set. Moreover, characteristic operators are *monotone* with respect to $\leq$, thus admitting the following (standard) construction that builds a canonical interpretation via a process of *approximation* [1, 7]:

Definition 2 (Interpretation of Inductive Definitions). *We interpret an inductive definition set Φ as the least prefixed point of its characteristic operator, $\mu X.\varphi_\Phi(X)$. This least prefixed point, denoted by $[\![\cdot]\!]^\Phi$, can be approached iteratively being the supremum of the (ordinal-indexed) chain $X_\bot \leq \varphi_\Phi(X_\bot) \leq \varphi_\Phi(\varphi_\Phi(X_\bot)) \leq \ldots \leq \varphi^\alpha_\Phi(X_\bot) \leq \ldots$; each $\varphi^\alpha_\Phi(X_\bot)$ is an* approximation *of $[\![\cdot]\!]^\Phi$ and is denoted by $[\![\cdot]\!]^\Phi_\alpha$. When the specific inductive rule set is not of immediate relevance we leave it implicit, writing $[\![\cdot]\!]$ and $[\![\cdot]\!]_\alpha$.*

3.2 The Cyclic Proof System

The proof system is essentially Gentzen's sequent calculus, LK, in which derivations are permitted to contain cycles. The full proof system is given in Appendix A. To the standard proof rules of LK with equality and substitution we add introduction rules for the inductive predicate symbols, derived from their productions. Each predicate P has a single left introduction rule, (Case P), which performs a case split over the full set of productions for P, and every i^{th} production for P induces a distinct right introduction rule ($\mathrm{P}\,\mathrm{R}_i$). Furthermore, we remove the right introduction rules for implication and negation since they invalidate the soundness of our realizability condition (specifically, not all instances of these rules satisfy Property 1, in Section 4 below). Although this system is actually quite weak, we believe these particular rules do not play a crucial role in deriving entailments between inductive predicates in general. Note we do not need them in our examples.

We view a cyclic derivation (or *pre-proof*) as a directed graph; each sequent is a node of the graph, and edges go from conclusion to premise. To track sequences of decreasing approximations, we use the notion of a trace in a pre-proof $\mathcal{P}$.

Definition 3 (Traces).

(i) *A* trace value *is a predicate formula (e.g.* $\mathrm{E}\,x$*).*

(ii) *A left-hand (resp. right-hand)* trace *is a possibly infinite sequence $\boldsymbol{\tau}$ of trace values in which those of each successive pair, (τ_i, τ_{i+1}), occur in the antecedents (resp. consequents) of successive nodes in $\mathcal{P}$, and either:*

- *a)* $\tau_i = \tau_{i+1}$*;*
- *b)* *τ_i and τ_{i+1} occur as part of the conclusion and premise of a substitution rule and τ_i is the result of applying the rule's substitution to τ_{i+1}; or*
- *c)* *τ_i and τ_{i+1} occur as part of the conclusion and premise of a (Case* P*) or (*$\mathrm{P}\,R_i$*) rule, with τ_i of the form* $\mathrm{P}\,\boldsymbol{t}$ *and τ_{i+1} derived from the body of the production for* P *associated with the premise of the rule (i.e. τ_{i+1} is derived from the* unfolding *of τ_i).*

We call each pair (τ_i, τ_{i+1}) a trace pair*. In the case that (c) holds, we say the trace* progresses *at point i and call (τ_i, τ_{i+1}) a* progressing *trace pair.*

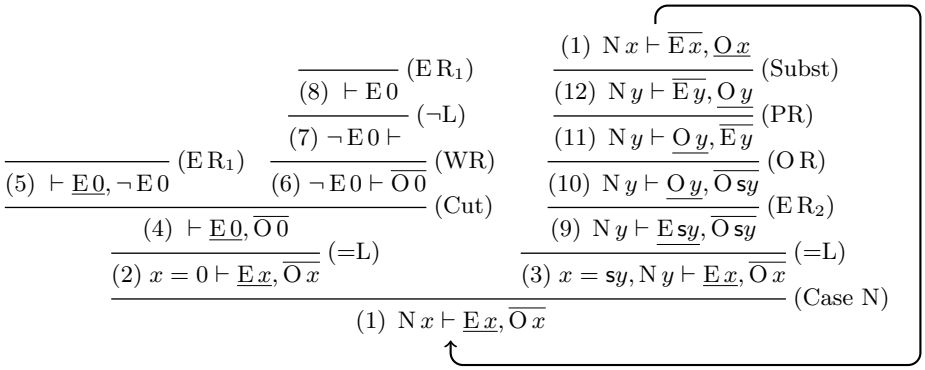


Fig. 2. A cyclic proof of the entailment $\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x$; each node is numbered uniquely, and the consequent trace pairs are indicated using under- and overlines.

(iii) For finite traces $\boldsymbol{\tau}$, we write $|\,\boldsymbol{\tau}\,|$ for the length of the trace and denote by $\mathsf{prog}(\boldsymbol{\tau})$ the number of progression points in $\boldsymbol{\tau}$, which we call the sum *of $\boldsymbol{\tau}$.*

(iv) For an inference rule $r = \langle S_0, (S_1, \ldots, S_n)\rangle$ with trace values τ and τ' occurring in the conclusion S_0 and j^{th} premise S_j, respectively, we write $\delta^{\mathcal{A}}_{(r,j)}(\tau, \tau')$ (resp. $\delta^{\mathcal{C}}_{(r,j)}(\tau, \tau')$) if (τ, τ') forms a left-hand (resp. right-hand) trace. We call δ the trace pair relation *(see Appendix B).*

When the meaning is clear from the context, we may sometimes simply write $\delta_r(\tau, \tau')$. In an abuse of notation we write $\delta_r(\tau, \tau') = 1$ to indicate that (τ, τ') is a progressing trace pair, and $\delta_r(\tau, \tau') = 0$ otherwise. When τ occurs in the conclusion of rule r, but there are no j and τ' such that $\delta_{(r,j)}(\tau, \tau')$ is defined, then we say τ is *terminal* for r.

Example 1. In Fig. 2 we show a cyclic proof of $\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x$, i.e. that every natural number is either even or odd. Each $\mathrm{N}\,t$ in an antecedent is related to the $\mathrm{N}\,t'$ in its premise(s); the trace pair relation for the consequent trace values is more complex, and we indicate it visually using under- and overlines.

A pre-proof is valid if it satisfies the following condition on traces.

Definition 4 (Global Soundness). *A pre-proof is* globally sound *when every infinite path has some tail that is followed by a left-hand trace which progresses infinitely often; when this holds we say that it is a (cyclic) proof.*

The global soundness of a pre-proof can be checked using Büchi automata.

Proposition 1 ([7, Prop. 7.4]). *It is decidable if a pre-proof is globally sound.*

Example 2. The pre-proof in Fig. 1 has only one infinite path (along the cycle), and there is a left-hand trace along this path formed by the alternating occurrences of the E and O predicates in the antecedent of each sequent. This progresses at two points around each cycle on traversing the (Case) rules and therefore the

pre-proof is globally sound. A similar argument shows the pre-proof in Fig. 2 is also globally sound: the (unique) infinite left-hand trace progresses once each time around the loop.

We may think of models as *realizers* of trace values. We define a trace *realization* function to specify which models realize trace values and how *quickly* they realize them.

Definition 5 (Trace Realization Function). *The* trace realization function Θ *maps models to the* least *approximations of trace values in which they appear:*

$$\Theta(\tau, m) \stackrel{def}{=} \mathsf{min}\,(\{\alpha \mid m \in [\![\tau]\!]_\alpha\})$$

The value assigned by Θ corresponds to the ordinal position of this approximation in the chain constructed in Definition 2. Notice that a model may not necessarily satisfy a given predicate formula, so Θ is partial *and we write $\Theta(\tau, m)\downarrow$ to indicate that Θ is defined on (τ, m).*

The global soundness condition ensures the validity of cyclic proofs because the trace realization function enables us to relate traces to descending chains of approximations. If a cyclic proof were to contain invalid sequents then the trace realization function could be used to derive an infinite descending chain of ordinals and hence a contradiction.

Theorem 1 ([7, Prop. 5.8]). *If $\Gamma \vdash \Delta$ has a cyclic proof then it is valid.*

4 Extracting Semantic Inclusions from Cyclic Proofs

We are aiming to deduce inclusions between the semantic approximations of predicates (viz. trace values), e.g. that whenever there is a model $m \in [\![\mathrm{E}\, x]\!]_\alpha$ then also $m \in [\![\mathrm{N}\, x]\!]_\alpha$ (cf. Fig. 1). We can express this using the trace realization function as $\Theta(\mathrm{N}\, x, m) \leq \Theta(\mathrm{E}\, x, m)$, since predicate approximations increase monotonically. We will deduce such relationships from sequents $\Gamma[\tau] \vdash \Delta[\tau']$ in cyclic proofs (where $\Gamma[\tau]$ indicates that the trace value τ occurs in Γ), and so in general we deduce such orderings within a context, Γ. Thus we will write $\Gamma : \tau' \leq \tau$ to mean:

$$\text{for all models } m, \text{ if } m \models \Gamma \text{ and } \Theta(\tau', m)\downarrow \text{ then } \Theta(\tau', m) \leq \Theta(\tau, m),$$

where $m \models \Gamma$ denotes that m satisfies all the formulas in Γ. We formulate an additional trace condition for cyclic proofs (Def. 8, below) and show that the existence of a proof satisfying this extra condition is sufficient to guarantee this ordering. We say that such a proof *realizes* the ordering, and so refer to the new trace condition as the *realizability condition*.

This realizability condition will express that for every right-hand trace of a certain kind, we can find a left-hand trace which 'matches' it in a sense that we will make precise below. We specify the kinds of right-hand traces of interest using the following concepts.

Definition 6 (Maximal Right-hand Traces). *A* finite *right-hand trace* $\boldsymbol{\tau}$ *(*$\mid \boldsymbol{\tau} \mid = n$*) following a path in a cyclic proof is called* maximal *when it cannot be extended any further, i.e. there is no trace value* τ' *and premise of the final node in the trace for which* $\delta_r(\boldsymbol{\tau}_n, \tau')$ *is defined (where* r *is the rule used to derive the final node). If the final node in the trace is derived using an axiom, then we say the trace is* partially *maximal; otherwise it is called* fully *maximal.*

Fully maximal traces are ones whose final trace value is introduced by an inference rule, e.g. weakening, as in node (6) of the proof in Fig. 2.

Definition 7 (Groundedness and Polarity). *We call a trace value derivable using a base production (i.e. a production without premises)* ground*, e.g.* $\mathrm{N}\,0$ *or* $\mathrm{E}\,0$*. A* grounded *trace is one whose final trace value is ground. When the antecedent of a sequent contains the negation of a ground predicate instance, we say that it is* negative*. A* positive *sequent is one with no such negated predicate. A negative (resp. positive) trace is one whose final sequent is negative (resp. positive).*

For example, in Fig. 2 the right-hand trace $(1, \mathrm{E}\,x)$, $(2, \mathrm{E}\,x)$, $(4, \mathrm{E}\,0)$, $(5, \mathrm{E}\,0)$ is grounded, but $(1, \mathrm{O}\,x)$, $(2, \mathrm{O}\,x)$, $(4, \mathrm{O}\,0)$, $(6, \mathrm{O}\,0)$ is not. Moreover, the latter trace is negative. Note that, by definition, all models m must satisfy ground predicate instances τ and $\Theta(\tau, m) = 1$. Thus no models may satisfy the antecedent of a negative sequent. This means that we can exclude negative traces when considering the realizability of trace value orderings. We can now formulate the realizability condition itself.

Definition 8 (The Realizability Condition). *We write* $\mathcal{P} : \tau \leq \Gamma[\tau']$ *when* $\mathcal{P}$ *is a cyclic proof containing a node* $\Gamma[\tau'] \vdash \Delta[\tau]$ *satisfying the following: for every positive maximal right-hand trace* $\boldsymbol{\tau}$ *starting at* τ*, there exists a left-hand trace* $\boldsymbol{\tau'}$ *starting with* τ' *and following some prefix of the same path in the proof such that:*

1. $\mathsf{prog}(\boldsymbol{\tau}) \leq \mathsf{prog}(\boldsymbol{\tau'})$ *and*
2. *either a)* $\boldsymbol{\tau}$ *is grounded; or b)* $\boldsymbol{\tau}$ *is partially maximal,* $\mid \boldsymbol{\tau'} \mid = \mid \boldsymbol{\tau} \mid$*, and the final trace values in* $\boldsymbol{\tau}$ *and* $\boldsymbol{\tau'}$ *match.*

Consider the proof $\mathcal{P}_1$ in Fig. 2.

Example 3 ($\mathcal{P}_1 : \mathrm{E}\,x \leq \mathrm{N}\,x$*).* The right-hand trace from $\mathrm{E}\,x$ following the path (1)(2)(4)(5) is positive, maximal and grounded. The left-hand trace (1) follows this path and the sum of both traces is 0. The next longest maximal right-hand trace traverses the cycle once, following the path (1)(3)(9)…(12)(1)(2)(4)(6) along the right-hand side of the (Cut) rule. However, this trace is *negative* and so we need not consider it. The other positive maximal traces are obtained by following the cycle an even number of times before ending at node (5); the progression points occur at $(\mathrm{E}\,\mathrm{R}_2)$ on the odd-numbered traversals and $(\mathrm{O}\,\mathrm{R}_2)$ on the even-numbered ones, which is matched by progressions in the corresponding left-hand trace at the (Case) rule. These traces also suffice to demonstrate that $\mathcal{P}_1 : \mathrm{O}\,x \leq \mathrm{N}\,x$ holds.

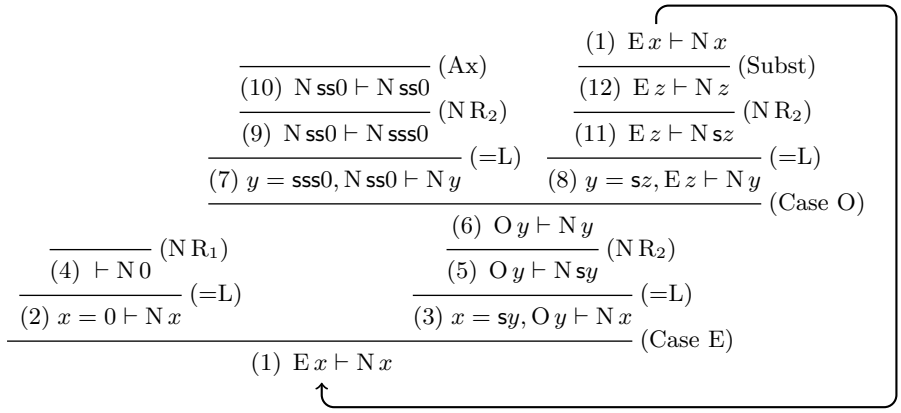


Fig. 3. A cyclic proof of the entailment $\mathrm{E}\,x \vdash \mathrm{N}\,x$, accommodating the extra production $\mathrm{N}\,\mathsf{ss0} \Rightarrow \mathrm{O}\,\mathsf{sss0}$ for O.

Notice that we can obtain a globally sound cyclic proof of $\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x$ without using (Cut), by immediately closing node (4) with $(\mathrm{E\,R_1})$ (See Fig. 7 in Appendix C). In this case the now (partially) maximal right-hand trace from $\mathrm{O}\,x$ in node (1) to $\mathrm{O}\,0$ in node (4) is *positive* and so would have to be considered. Unfortunately this trace is *not* grounded, nor does there exist a matching left-hand trace of equal length ending with $\mathrm{O}\,0$, and so this simpler (and arguably more natural) proof does not satisfy the realizability condition.

It may seem odd that we cannot use the simpler proof to realize the ordering. We must discount the right-hand traces ending with $\mathrm{O}\,0$ since they have no models; yet it is not possible in general to determine syntactically when predicate instances do *not* have models. Our approximation, using negative traces, works at the level of entire sequents and thus the traces ending with $\mathrm{E}\,0$ (which we do consider) must be separated from those ending in $\mathrm{O}\,0$ (which we must not). This highlights the syntactic nature of our results.

Now consider the proof $\mathcal{P}_2$ of $\mathrm{E}\,x \vdash \mathrm{N}\,x$ in Fig. 3, which is a modified version of the proof in Fig. 1 that accommodates an additional production for O.

Example 4 ($\mathcal{P}_2 : \mathrm{N}\,x \leq \mathrm{E}\,x$). The right-hand trace following (1)(2)(4) is maximal, positive and grounded and the left-hand trace $(1, \mathrm{E}\,x)$ follows (a prefix of) the same path; the sum of both of these traces is 0. Similarly, the positive right-hand trace following (1)(3)(5)(6)(7)(9)(10) is not grounded, but is partially maximal and there is a left-hand trace of equal length following this same path with a matching final trace value. The sum of both traces in this case is 2: the right-hand trace progresses once at each instance of the $(\mathrm{N\,R_2})$ rule; the left-hand one at the (Case) rules. Other maximal right-hand traces are obtained by prefixing the cycle (1)...(12) to the two already considered; notice the left-hand trace following the cycle progresses an equal number of times.

Soundness of Realizability. To show that the realizability condition is sufficient to realize trace value orderings, we extend the concept that models realize trace values and use sequences of models to realize *traces*. We say that a sequence of models $\boldsymbol{m}$ realizes a left-hand trace $\boldsymbol{\tau}$ when for every sequent $\Gamma_i[\tau_i] \vdash \Delta_i$ in the corresponding path we have that $m_i \models \Gamma_i$ and $\Theta(\tau_{i+1}, m_{i+1}) + \delta(\tau_i, \tau_{i+1}) \leq \Theta(\tau_i, m_i)$. Dually, $\boldsymbol{m}$ realizes a right-hand trace $\boldsymbol{\tau}$ when $m_i \models \Delta_i$ and $\Theta(\tau_{i+1}, m_{i+1}) + \delta(\tau_i, \tau_{i+1}) \geq \Theta(\tau_i, m_i)$ for every sequent $\Gamma_i \vdash \Delta_i[\tau_i]$ in the path. Trace realizers guarantee the following.

Lemma 1. *If $\boldsymbol{m}$ realizes a trace $\boldsymbol{\tau}$ of length n then $\Theta(\tau_n, m_n) + \mathsf{prog}(\boldsymbol{\tau}) \leq \Theta(\tau_1, m_1)$ holds if $\boldsymbol{\tau}$ is a left-hand trace, and $\Theta(\tau_n, m_n) + \mathsf{prog}(\boldsymbol{\tau}) \geq \Theta(\tau_1, m_1)$ if $\boldsymbol{\tau}$ is a right-hand trace.*

We say a rule instance is *valid* when its conclusion and premises are all valid sequents.[3] We note the following property of the trace realization function.

Property 1 (Descending Model Property). For all valid, non-axiomatic rule instances $r = \langle \Gamma[\tau] \vdash \Delta[\tau'], (S_1, \ldots, S_n) \rangle$ and models $m \models \Gamma$, there exists some $S_j = \Sigma \vdash \Pi$ and a model $m' \models \Sigma$ such that: either τ' is terminal for r, or there exists τ'' with $\delta_{(r,j)}(\tau', \tau'')$ defined; furthermore, for all trace values τ'':

1. if $\delta^{\mathcal{A}}_{(r,j)}(\tau, \tau'') = \alpha$ and $\Theta(\tau, m)\downarrow$, then $\Theta(\tau'', m')\downarrow$ and $\Theta(\tau'', m') + \alpha \leq \Theta(\tau, m)$
2. if $\delta^{\mathcal{C}}_{(r,j)}(\tau, \tau'') = \alpha$ and $\Theta(\tau', m)\downarrow$, then $\Theta(\tau'', m')\downarrow$ and $\Theta(\tau'', m') + \alpha \geq \Theta(\tau', m)$

This property asserts that the trace pair relation soundly *bounds* the difference in how quickly models realize trace pairs. In the case of antecedents this difference is bounded from above, and for consequents from below. The descending model property guarantees every model of a consequent trace value in a globally sound cyclic proof corresponds to a realizer of a positive maximal right-hand trace.

Lemma 2 (Trace Realization). *If $\mathcal{P}$ is a globally sound cyclic proof containing a node $\Gamma[\tau'] \vdash \Delta[\tau]$ and m is a model such that $m \models \Gamma$ and $\Theta(\tau, m)\downarrow$, then there exists a positive, maximal right-hand trace $\boldsymbol{\tau}$ starting from τ and a sequence of models $\boldsymbol{m}$ with $m_1 = m$ that realizes it; moreover, $\boldsymbol{m}$ realizes all left-hand traces following the same path starting from τ'.*

As a result, the realizability condition is sufficient to guarantee trace value orderings (see the technical report for a detailed proof [18, Thm. 22]).

Theorem 2 (Soundness of Realizability). *If $\mathcal{P} : \tau \leq \Gamma[\tau']$ then $\Gamma : \tau \leq \tau'$.*

[3] Note this is a stronger property than local soundness, which only requires the conclusion to be valid whenever all of the premises are.

5 Computing Realizable Orderings using Weighted Automata

In this section, we demonstrate a close connection between cyclic proofs and weighted automata. Under this correspondence, the realizability condition can be seen to be equivalent to an inclusion between particular weighted automata, allowing us to make use of known decision procedures in the world of weighted automata for deciding the realizability condition.

Weighted automata generalise standard finite state automata, assigning to words over alphabet Σ values from a semiring $(V, \oplus, \otimes)$ of weights (see [8]).

Definition 9 (Weighted Automata). *A weighted automaton $\mathscr{A}$ is a tuple (Q, q_I, F, γ) consisting of a set Q of states containing an initial state $q_I \in Q$, a set $F \subseteq Q$ of final states, and a weighting function $\gamma : (Q \times \Sigma \times Q) \to V$.*

A *run* of $\mathscr{A}$ over a (finite) word $\sigma_1 \dots \sigma_n \in \Sigma^*$ is a sequence of states $q_0 \dots q_n$ such that $(q_{j-1}, \sigma_j, q_j) \in \mathsf{dom}(\gamma)$ for each σ_j. We write $\rho : q_0 \xrightarrow{w} q_n$ to denote that ρ is a run over w. The value $\mathsf{V}(\rho)$ of the run is the (left-to-right) semiring product of the weight $\gamma(q_{j-1}, \sigma_j, q_j)$ of each transition. If $q_0 = q_I$ and $q_n \in F$ then ρ is called an *accepting* run. The value of a word is the semiring sum of the values of all the accepting runs of that word, and is undefined if there are no such runs. Sum automata are weighted automata over the max-plus semiring $(\mathbb{N}, \mathsf{max}, +)$, which is also referred to as the arctic semiring.

The (quantitative) language $\mathcal{L}_{\mathscr{A}}$ of an automaton $\mathscr{A}$ is the (partial) function over Σ^* computed by the automaton. The standard notion of inclusion between regular languages extends naturally to quantitative languages:

Definition 10 (Weighted Inclusion). *$\mathcal{L}_1 \leq \mathcal{L}_2$ if and only if for every word w such that $\mathcal{L}_1(w)$ is defined, $\mathcal{L}_2(w)$ is also defined and $\mathcal{L}_1(w) \leq \mathcal{L}_2(w)$.*

The inclusion problem for sum automata is known to be undecidable [13, 2], but has recently been shown to be decidable for *finite-valued* sum automata, for which a finite bound can be given on the number of distinct values for runs over a given word [10].

5.1 Cyclic Proofs as Sum Automata

Given a node $n = \Gamma[\tau] \vdash \Delta[\tau']$ in a cyclic proof $\mathcal{P}$ we construct two sum automata $\mathscr{A}^{\tau}_{\mathcal{P}}$ and $\mathscr{C}^{\tau'}_{\mathcal{P}}$ called left-hand and right-hand *trace automata*, respectively. Each state (n, τ) of a trace automaton corresponds to a particular trace value τ in some node n of $\mathcal{P}$, and the transitions are given by the trace pair relation. That is, there is a transition from (n, τ) to (n', τ') with weight $k \in \{0, 1\}$ precisely when n and n' are the conclusion and j^{th} premise, respectively, of a rule instance r with $\delta_{(r,j)}(\tau, \tau') = k$. The letter accepted on the transition is the node n'. Thus, a run of one of these automata corresponds to a trace in $\mathcal{P}$, and the word accepted by the run is the path followed by the trace.

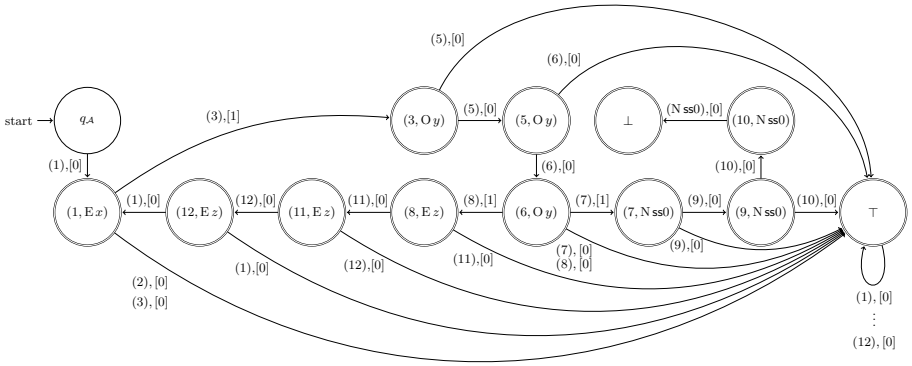


Fig. 4. The left-hand trace automaton $\mathscr{A}_{\mathcal{P}}^{\mathrm{E}\,x}$ for the proof of $\mathrm{E}\,x \vdash \mathrm{N}\,x$ in Fig. 3.

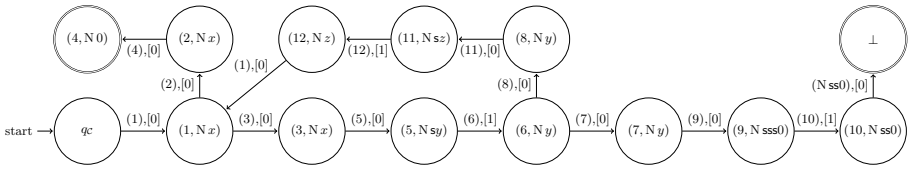


Fig. 5. The right-hand trace automaton $\mathscr{C}_{\mathcal{P}}^{\mathrm{N}\,x}$ for the proof of $\mathrm{E}\,x \vdash \mathrm{N}\,x$ in Fig. 3.

For lack of space, we elide the formal definition of the automata construction (see Appendix D and [18, Def. 23]), but in Figs. 4 and 5 we show the trace automata corresponding to the proof in Fig. 3. Accepting states are indicated by a double circle, and for each transition we show the node accepted in parentheses and the weight of the transition in brackets. We draw attention to the following:

- The left-hand trace automaton also includes (zero-weight) transitions to a state $\top$ with a self-transition accepting any node. Thus, the weight it computes for a path is the maximum value of $\mathsf{prog}(\boldsymbol{\tau})$ over all traces $\boldsymbol{\tau}$ following a *prefix* of that path. In contrast, the right-hand automaton considers only traces following the *full* path.
- Each automaton also includes a state $\bot$, the transitions to which accept a trace value rather than a node. The effect of this is that any word $w = n_1 \dots n_k \tau$ accepted by the right-hand automaton corresponds to a *partially* maximal right-hand trace ending in τ. If the left-hand automaton also accepts w, then we know there is a matching left-hand trace of equal length (cf. Def. 8).
- The accepting states of right-hand trace automata (excluding $\bot$) correspond to *terminal* trace values in non-axiomatic rules instances; when each such trace value is ground, we say the trace automaton is *grounded*.

This construction results in automata polynomial in the size of the proof $\mathcal{P}$, and allows the realizability condition to be encoded by the inclusion of the right-hand trace automaton within the left-hand one.

Theorem 3. *$\mathcal{P} : \tau \leq \Gamma[\tau']$ holds if and only if $\mathscr{C}^{\tau}_{\mathcal{P}} \leq \mathscr{A}^{\tau'}_{\mathcal{P}}$ and $\mathscr{C}^{\tau}_{\mathcal{P}}$ is grounded.*

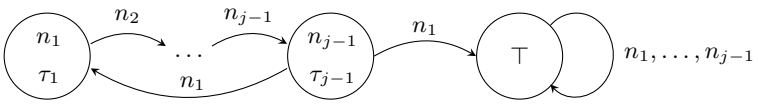
5.2 Decidability of the Realizability Condition

We now demonstrate that under certain conditions our trace automata become *finite-valued*, and so we can decide inclusion between them in polynomial time [10].

Remark 1. The trace pair relation δ satisfies an injectivity property[4]. Namely, if both $\delta_{(r,j)}(\tau', \tau)$ and $\delta_{(r,j)}(\tau'', \tau)$ are defined, then $\tau' = \tau''$. This means that, along any given path, traces may only branch and never converge. Consequently, there is at most one trace along a given path between an initial and final trace value. This immediately gives the following result.

Lemma 3. *Every right-hand trace automaton $\mathcal{C}^{\tau}_{\mathcal{P}}$ is finite-valued.*

Unfortunately, because left-hand trace automata include the state $\top$ and associated transitions, they are not in general finite-valued. When a proof contains a (left-hand) trace cycle (of the form $(n_1, \tau_1) \dots (n_j, \tau_j)$ with nodes $n_1 = n_j$ and trace values $\tau_1 = \tau_j$), the resulting left-hand trace automaton will contain the following configuration of states:



That is, there are runs $(n_{j-1}, \tau_{j-1}) \xrightarrow{w} (n_{j-1}, \tau_{j-1})$, $(n_{j-1}, \tau_{j-1}) \xrightarrow{w} \top$, and $\top \xrightarrow{w} \top$ with $w = n_1 \dots n_{j-1}$. This results in the automaton being infinitely ambiguous [19, §3] and thus when the weight of the cycle is non-zero it is also infinite-valued.

To avoid this we modify our construction to produce a series of *approximate* left-hand trace automata $\mathscr{A}[k]^{\tau}_{\mathcal{P}}$, where $k > 0$ is called the *degree* of approximation. These refine the 'sink' state $\top$ into a finite chain of k sink states for each node (thus, these approximate automata are a factor of k larger than the original automaton). In Appendix D, we show the approximate automaton of degree $k = 2$ corresponding to the proof in Fig. 3. Once a run enters a chain of sink states $\top^{1..k}_{n}$, only a finite number of further occurrences of the node n are accepted. In contrast, the full automaton accepts paths with any number of further occurrences. This construction approximates the original one and results in finite-valued automata.

Lemma 4. *Every approximate left-hand trace automaton $\mathscr{A}[k]^{\tau}_{\mathcal{P}}$ is finite-valued.*

Lemma 5 (Soundness of Approximate Automata). *For each $k > 0$, the inclusion $\mathscr{A}[k]^{\tau}_{\mathcal{P}} \leq \mathscr{A}^{\tau}_{\mathcal{P}}$ holds.*

[4] excepting certain instances of the (=L) rule, e.g. $\mathrm{P}\, x, \mathrm{P}\, x \vdash \Delta \Rightarrow \mathrm{P}\, x, \mathrm{P}\, y, x = y \vdash \Delta$. However, note that one can check whether any given instance of (=L) satisfies the injectivity property, and exclude proofs containing such instances from consideration.

The following further restrictions on proofs allow a relative completeness result. They are expressed in terms of *simple* trace cycles (containing no repeated trace values other than the first and last). A *binary* trace cycle is a pair of trace cycles following the same path.

Definition 11. *Let $S = \Gamma[\tau'] \vdash \Delta[\tau]$ be a node in a cyclic proof $\mathcal{P}$. We say $\mathcal{P}$ is* dynamic *(w.r.t. S) when $\mathsf{prog}(\boldsymbol{\tau}) > 0$ for every simple left- and right-hand trace cycle $\boldsymbol{\tau}$ reachable from τ' and τ, respectively. We say $\mathcal{P}$ is* balanced *(w.r.t. S) when $\mathsf{prog}(\boldsymbol{\tau}_1) = \mathsf{prog}(\boldsymbol{\tau}_2)$ for every simple left-hand binary cycle $(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2)$ reachable from τ'.*

Checking whether a proof is balanced and dynamic requires finding the simple cycles, which can be done in time $\mathsf{O}((N + E)(C + 1))$, where N, E, and C are the number of nodes, edges and basic cycles in the graph, respectively [12]. The number of basic cycles in a *complete* graph is factorial in the number of nodes, thus the worst case complexity is super-exponential. Notwithstanding, cyclic proofs are by nature sparse graphs, so we expect the average runtime complexity to be much lower. All of our example proofs are both balanced and dynamic.

When a balanced, dynamic proof satisfies the realizability condition, its positive fully-maximal right-hand traces are always matched by left-hand traces that can be recognised by an approximate left-hand automaton whose degree of approximation can be bounded by the following two graph-theoretic quantities (which are polynomially bounded in the size of $\mathcal{P}$).

a) The *trace width* $\mathbf{W}(\mathcal{P})$ is the maximum number of trace values occurring in the antecedent or consequent of any node in $\mathcal{P}$. Any trace visiting a given node more than $\mathbf{W}(\mathcal{P})$ times *must* contain a cycle.
b) The binary left-hand *cycle threshold* $\mathbf{C}(\mathcal{P})$ is the number of distinct pairs of left-hand trace values occurring in $\mathcal{P}$. Any pair of left-hand traces following the same path of length greater than $\mathbf{C}(\mathcal{P})$ *must* contain a binary cycle.

Lemma 6 (Relative Completeness). *If $\mathcal{P} : \tau \leq \Gamma[\tau']$ and $\mathcal{P}$ is both dynamic and balanced with respect to $\Gamma[\tau'] \vdash \Delta[\tau]$, then $\mathscr{C}^{\tau}_{\mathcal{P}} \leq \mathscr{A}[N]^{\tau'}_{\mathcal{P}}$ and $\mathscr{C}^{\tau}_{\mathcal{P}}$ is grounded, where $N = 2 + \mathbf{W}(\mathcal{P}) \times (\mathbf{C}(\mathcal{P}) + 1)$.*

From this, a qualified form of decidability follows. Note that when $\mathcal{P}$ is not balanced and dynamic we still have a semi-decision procedure.

Theorem 4. *If $\mathcal{P}$ is dynamic and balanced with respect to $\Gamma[\tau'] \vdash \Delta[\tau]$, then it is decidable whether $\mathcal{P} : \tau \leq \Gamma[\tau']$ holds.*

6 Conclusions and Future Work

In this paper, we have demonstrated that cyclic proofs of entailments involving inductively defined predicates implicitly contain information about the relationship between the semantic approximations of these predicates. This information is useful because indexing ordinals for these approximations can be used, e.g.,

as (components of) ranking functions in a program termination proof. We have shown that this information can be made explicit via a novel trace condition, and furthermore we have proved this condition to be decidable via a construction using weighted automata. Although different in form, we have drawn tacit parallels between our work and the (intuitionistic) concept of realizability because we extract the semantic information directly from the proofs themselves.

Our results also increase the expressive power of the cyclic proof technique. For example, if we can deduce from the proof of $\Gamma, \mathrm{P}\,\boldsymbol{t} \vdash \Sigma, \mathrm{Q}\,\boldsymbol{u}$ that $\mathrm{Q}\,\boldsymbol{u} \leq \mathrm{P}\,\boldsymbol{t}$ then we can safely form a well-founded trace across the active formula in the cut application

$$\frac{\Gamma, \mathrm{P}\,\boldsymbol{t} \vdash \Sigma, \mathrm{Q}\,\boldsymbol{u} \quad \Sigma, \mathrm{Q}\,\boldsymbol{u} \vdash \Delta}{\Gamma, \mathrm{P}\,\boldsymbol{t} \vdash \Delta}$$

from $\mathrm{P}\,\boldsymbol{t}$ in the conclusion to $\mathrm{Q}\,\boldsymbol{u}$ in the right-hand premise, and therefore witness the validity of cyclic pre-proofs that do not satisfy the existing global soundness condition for cyclic proofs.

An obvious direction for future work is to implement our decision procedure and integrate it with existing cyclic proof-based program verifiers, such as [17] which currently relies on explicit ordinal variables to track approximations. A question of practical importance is whether entailment proofs typically encountered in program verification fall under the conditions for decidability of the trace condition. It is interesting to consider whether weaker conditions exist that still guarantee decidability. There are also wider theoretical questions to consider. Our trace condition is sound, but it is also natural to ask for completeness: if $\Gamma : \tau \leq \tau'$ holds does there also exist a proof $\mathcal{P}$ for which $\mathcal{P} : \tau \leq \Gamma[\tau']$ holds?

Acknowledgements. We extend thanks to Radu Grigore, Carsten Fuhs, and the PPLV group at UCL for useful discussions and invaluable comments. We are grateful to Alexandra Silva for suggesting to investigate weighted automata. This work was supported primarily by EPSRC grant EP/K040049/1, and also by EPSRC grant EP/N028759/1.

References

1. Peter Aczel. An Introduction to Inductive Definitions. In Jon Barwise, editor, *Handbook of Mathematical Logic*, pages 739–782. North-Holland, 1977.
2. Shaull Almagor, Udi Boker, and Orna Kupferman. What's Decidable about Weighted Automata? In *ATVA-9*, volume 6996 of *LNCS*, pages 482–491. Springer-Verlag, 2011. doi: 10.1007/978-3-642-24372-1_37.
3. Josh Berdine, Byron Cook, Dino Distefano, and Peter W. O'Hearn. Automatic Termination Proofs for Programs with Shape-shifting Heaps. In *CAV*, volume 4144 of *LNCS*, pages 386–400. Springer, 2006. doi: 10.1007/11817963_35.
4. James Brotherston. Cyclic Proofs for First-Order Logic with Inductive Definitions. In *TABLEAUX-14*, volume 3702 of *LNAI*, pages 78–92. Springer-Verlag, 2005. doi: 10.1007/11554554_8.

5. James Brotherston. Formalised Inductive Reasoning in the Logic of Bunched Implications. In *SAS-14*, volume 4634 of *LNCS*, pages 87–103. Springer-Verlag, 2007. doi: 10.1007/978-3-540-74061-2_6.
6. James Brotherston, Richard Bornat, and Cristiano Calcagno. Cyclic Proofs of Program Termination in Separation Logic. In *POPL-35*, volume 43 of *ACM SIGPLAN Notices*, pages 101–112. ACM, 2008. doi: 10.1145/1328438.1328453.
7. James Brotherston and Alex Simpson. Sequent Calculi for Induction and Infinite Descent. *Journal of Logic and Computation*, 21(6):1177–1216, December 2011. doi: 10.1093/logcom/exq052.
8. Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Monographs in Theoretical Computer Science. Springer-Verlag, 2009. doi: 10.1007/978-3-642-01492-5.
9. Gadi Tellez Espinosa and James Brotherston. Automatically Verifying Temporal Properties of Programs with Cyclic Proof. In *CADE-26*, 2017. To appear.
10. Emmanuel Filiot, Raffaella Gentilini, and Jean-François Raskin. Finite-Valued Weighted Automata. In *FSTTCS-34*, volume 29 of *LIPICS*, pages 133–145, 2014. doi: 10.4230/LIPIcs.FSTTCS.2014.133.
11. Samin Ishtiaq and Peter W. O'Hearn. BI as an assertion language for mutable data structures. In *Proc. POPL-28*, pages 14–26. ACM, 2001. doi: 10.1145/373243.375719.
12. Donald B. Johnson. Finding All the Elementary Circuits of a Directed Graph. *SIAM J. Comput.*, 4(1):77–84, 1975. doi: 10.1137/0204007.
13. Daniel Krob. The Equality Problem for Rational Series with Multiplicities in the Tropical Semiring is Undecidable. *IJAC*, 4(3):405–426, 1994. doi: 10.1142/S0218196794000063.
14. Chin Soon Lee, Neil D. Jones, and Amir M. Ben-Amram. The Size-change Principle for Program Termination. In *POPL-28*, pages 81–92. ACM, 2001. doi: 10.1145/373243.360210.
15. Per Martin-Löf. Hauptsatz for the Intuitionistic Theory of Iterated Inductive Definitions. In *2nd Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of Mathematics*, pages 179–216. North-Holland, 1971.
16. John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proc. LICS-17*, pages 55–74. IEEE, 2002. doi: 10.1109/LICS.2002.1029817.
17. Reuben N. S. Rowe and James Brotherston. Automatic Cyclic Termination Proofs for Recursive Procedures in Separation Logic. In *CPP-6*, pages 53–65. ACM, 2017. doi: 10.1145/3018610.3018623.
18. Reuben N. S. Rowe and James Brotherston. Size Relationships in Abstract Cyclic Entailment Systems. Technical report, 2017. https://arxiv.org/abs/1702.03981.
19. Andreas Weber and Helmut Seidl. On the Degree of Ambiguity of Finite Automata. *Theor. Comput. Sci.*, 88(2):325–349, 1991. doi: 10.1016/0304-3975(91)90381-B.

A Proof Rules

Structural Rules

$$\text{(Axiom)}: \frac{}{F \vdash F} \qquad \text{(Subst)}: \frac{\Gamma \vdash \Delta}{\Gamma\theta \vdash \Delta\theta} \qquad \text{(Cut)}: \frac{\Gamma \vdash F, \Delta \quad \Sigma, F \vdash \Pi}{\Gamma, \Sigma \vdash \Delta, \Pi}$$

$$\text{(WL)}: \frac{\Gamma \vdash \Delta}{\Gamma, F \vdash \Delta} \quad \text{(WR)}: \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, F} \quad \text{(PL)}: \frac{\Gamma, A, B, \Sigma \vdash \Delta}{\Gamma, B, A, \Sigma \vdash \Delta}$$

$$\text{(CL)}: \frac{\Gamma, F, F \vdash \Delta}{\Gamma, F \vdash \Delta} \quad \text{(CR)}: \frac{\Gamma \vdash F, F, \Delta}{\Gamma \vdash F, \Delta} \quad \text{(PR)}: \frac{\Gamma \vdash \Delta, A, B, \Sigma}{\Gamma \vdash \Delta, B, A, \Sigma}$$

Logical Rules

$$\text{(}\vee\text{L)}: \frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta} \qquad \text{(}\vee\text{R)}: \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}$$

$$\text{(}\wedge\text{L)}: \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} \qquad \text{(}\wedge\text{R)}: \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}$$

$$\text{(}\to\text{L)}: \frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \to B \vdash \Delta} \qquad \text{(}\neg\text{L)}: \frac{\Gamma \vdash F, \Delta}{\Gamma, \neg F \vdash \Delta}$$

$$\text{(}\forall\text{L)}: \frac{\Gamma, F[t/x] \vdash \Delta}{\Gamma, \forall x F \vdash \Delta} \qquad \text{(}\forall\text{R)}: \frac{\Gamma \vdash F, \Delta}{\Gamma \vdash \forall x F, \Delta}\ x \notin FV(\Gamma, \Delta)$$

$$\text{(}\exists\text{L)}: \frac{\Gamma, F \vdash \Delta}{\Gamma, \forall x F \vdash \Delta}\ x \notin FV(\Gamma, \Delta) \qquad \text{(}\exists\text{R)}: \frac{\Gamma \vdash F[t/x], \Delta}{\Gamma \vdash \forall x F, \Delta}$$

$$\text{(=L)}: \frac{\Gamma[u/x, t/y] \vdash \Delta[u/x, t/y]}{\Gamma[t/x, u/y], t = u \vdash \Delta[t/x, u/y]} \qquad \text{(=R)}: \frac{}{\Gamma \vdash t = t, \Delta}$$

Excluded Rules

$$\text{(}\neg\text{R)}: \frac{\Gamma, F \vdash \Delta}{\Gamma \vdash \neg F, \Delta} \qquad \text{(}\to\text{R)}: \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \to B, \Delta}$$

Fig. 6. Proof Rules for Gentzen's Sequent Calculus LK with Equality and Substitution.

Proof Rules Induced by Inductive Predicate Definitions

For an inductive definition set Φ, we write Φ_P for the set of productions in Φ for P. Each predicate P defined by Φ induces a single left introduction rule and a number

of right introduction rules for P. For each production $\mathrm{P}_1\,\boldsymbol{t_1}, \ldots, \mathrm{P}_n\,\boldsymbol{t_n} \Rightarrow \mathrm{P}\,\boldsymbol{t_0}$, we include a right introduction rule for P (where θ is a substitution of terms for variables):

$$(\mathrm{P\,R}_i)\colon \frac{\Gamma \vdash \mathrm{P}_1\,\boldsymbol{t_1}\theta, \Delta \quad \ldots \quad \Gamma \vdash \mathrm{P}_n\,\boldsymbol{t_n}\theta, \Delta}{\Gamma \vdash \mathrm{P}\,\boldsymbol{t_0}\theta, \Delta}$$

We also include a single left introduction rule for P, which performs a case split over all the productions in $\Phi_{\mathrm{P}} = \{\Pi_1 \Rightarrow \mathrm{P}\,\boldsymbol{t_1}, \ldots, \Pi_n \Rightarrow \mathrm{P}\,\boldsymbol{t_n}\}$:

$$(\text{Case P})\colon \frac{\Gamma, \boldsymbol{u} = \boldsymbol{t_1}\theta_1, \Pi_1\theta_1 \vdash \Delta \quad \ldots \quad \Gamma, \boldsymbol{u} = \boldsymbol{t_n}\theta_n, \Pi_n\theta_n \vdash \Delta}{\Gamma, \mathrm{P}\,\boldsymbol{u} \vdash \Delta}$$

where, for sequences of terms $\boldsymbol{t}$ and $\boldsymbol{u}$ of equal length m, $\boldsymbol{u} = \boldsymbol{t}\theta$ stands for the sequence of equalities $u_1 = t_1\theta, \ldots, u_m = t_m\theta$, and each θ_i is a substitution that freshens the variables in $\mathsf{FV}(\Pi_i, \mathrm{P}\,\boldsymbol{t_i})$ so that they do not clash with the variables in $\mathsf{FV}(\Gamma, \Delta, \mathrm{P}\,\boldsymbol{u})$.

B The Trace Pair Relation

A trace value τ is an atomic predicate formula (i.e. of the form $\mathrm{P}\,\boldsymbol{t}$). For a sequence of formulas Γ, we write $\tau \in \Gamma$ to indicate that the trace value τ appears in Γ. In the following definition, when we use Γ, Δ, $\mathrm{P}\,\boldsymbol{t_i}$, θ, etc., we mean to refer to these elements as they appear in the appropriate rule schema presented in Appendix A.

Definition 12 (Trace Pair Relation). *Writing $\delta_r(\tau, \tau') = 1$ and $\delta_r(\tau, \tau') = 0$ to indicate progressing and non-progressing trace pairs, respectively, the trace pair relation δ is defined by cases as follows (we elide the subscript indicating the particular premise under consideration, when the rule instance has only a single one):*

$$\delta^{\mathcal{A}}_{(Subst)}(\tau\theta, \tau) = 0, \ \textit{if } \tau \in \Gamma$$
$$\delta^{\mathcal{C}}_{(Subst)}(\tau\theta, \tau) = 0, \ \textit{if } \tau \in \Delta$$

$$\delta^{\mathcal{A}}_{(=L)}(\tau[t/x, u/y], \tau[u/x, t/y]) = 0, \ \textit{if } \tau \in \Gamma$$
$$\delta^{\mathcal{C}}_{(=L)}(\tau[t/x, u/y], \tau[u/x, t/y]) = 0, \ \textit{if } \tau \in \Delta$$

$$\delta^{\mathcal{C}}_{(\mathrm{P}\ R_i,\, j)}(\mathrm{P}\,\boldsymbol{t_0}\theta, \mathrm{P}_j\,\boldsymbol{t_j}\theta) = 1 \qquad (1 \le j \le n)$$
$$\delta^{\mathcal{A}}_{(Case\ \mathrm{P},\, j)}(\mathrm{P}\,\boldsymbol{u}, \tau\theta_j) = 1, \ \textit{if } \tau \in \Pi_j \qquad (1 \le j \le n)$$

$$\delta^{\mathcal{A}}_{(\mathrm{P}\ R_i,\, j)}(\tau, \tau) = 0, \ \textit{if } \tau \in \Gamma \qquad \delta^{\mathcal{C}}_{(\mathrm{P}\ R_i,\, j)}(\tau, \tau) = 0, \ \textit{if } \tau \in \Delta \qquad (1 \le j \le n)$$
$$\delta^{\mathcal{A}}_{(Case\ \mathrm{P},\, j)}(\tau, \tau) = 0, \ \textit{if } \tau \in \Gamma \quad \delta^{\mathcal{C}}_{(Case\ \mathrm{P},\, j)}(\tau, \tau) = 0, \ \textit{if } \tau \in \Delta \qquad (1 \le j \le n)$$

and for all other non-axiomatic rules (r)*, each of the general form:*

$$(r)\colon \frac{\Sigma_1 \vdash \Pi_1 \quad \dots \quad \Sigma_n \vdash \Pi_n \quad (n \leq 2)}{\Gamma \vdash \Delta}$$

we include the following cases $(1 \leq i \leq n)$*:*

$$\delta^{\mathcal{A}}_{(r,i)}(\tau,\tau) = 0, \text{ if } \tau \in \Gamma \text{ and } \tau \in \Sigma_i$$
$$\delta^{\mathcal{C}}_{(r,i)}(\tau,\tau) = 0, \text{ if } \tau \in \Delta \text{ and } \tau \in \Pi_i$$

C Alternative Proof of $\mathbf{N}\,x \vdash \mathbf{E}\,x, \mathbf{O}\,x$

$$\dfrac{\dfrac{}{\vdash \mathrm{E}\,0, \mathrm{O}\,0}\,(\mathrm{E}\,\mathrm{R}_1)}{x = 0 \vdash \mathrm{E}\,x, \mathrm{O}\,x}\,(=\mathrm{L}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x}{\mathrm{N}\,y \vdash \mathrm{E}\,y, \mathrm{O}\,y}\,(\mathrm{Subst})}{\mathrm{N}\,y \vdash \mathrm{O}\,y, \mathrm{E}\,y}\,(\mathrm{PR})}{\mathrm{N}\,y \vdash \mathrm{O}\,y, \mathrm{O}\,\mathsf{s}y}\,(\mathrm{O}\,\mathrm{R}_1)}{\mathrm{N}\,y \vdash \mathrm{E}\,\mathsf{s}y, \mathrm{O}\,\mathsf{s}y}\,(\mathrm{E}\,\mathrm{R}_2)}{x = \mathsf{s}y, \mathrm{N}\,y \vdash \mathrm{E}\,x, \mathrm{O}\,x}\,(=\mathrm{L})$$
$$\dfrac{x = 0 \vdash \mathrm{E}\,x, \mathrm{O}\,x \qquad x = \mathsf{s}y, \mathrm{N}\,y \vdash \mathrm{E}\,x, \mathrm{O}\,x}{\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x}\,(\text{Case N})$$

Fig. 7. An alternative cyclic proof of the entailment $\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x$ which does not satisfy the realizability condition.

D Trace Automata

In the following definitions, we write $(n, n') \in \mathcal{P}$ to mean that there is an edge from node n to n' in $\mathcal{P}$. In an abuse of notation, we also use nodes to refer to the rule instance of which they are a conclusion and write $\delta^{\mathcal{A}}_{(n,n')}$ and $\delta^{\mathcal{C}}_{(n,n')}$ for $\delta^{\mathcal{A}}_{(r,j)}$ and $\delta^{\mathcal{C}}_{(r,j)}$ respectively, when n and n' are, respectively, the conclusion and j^{th} premise of the rule instance r.

Trace automata recognise words over the alphabet $\Sigma_{\mathcal{P}} = \mathbf{N}(\mathcal{P}) \cup \mathbf{T}(\mathcal{P})$, where $\mathbf{N}(\mathcal{P})$ and $\mathbf{T}(\mathcal{P})$ denote the set of nodes and set of trace values (i.e. predicate formulas) occurring in $\mathcal{P}$, respectively.

Definition 13 (Trace Automata). *Let* $n_{init} = \Gamma[\tau] \vdash \Delta[\tau']$ *be a node in a cyclic proof* $\mathcal{P}$*, then the sum automata* $\mathscr{A}^{\tau}_{\mathcal{P}} = (Q_{\mathcal{A}}, q_{\mathcal{A}}, F_{\mathcal{A}}, \gamma_{\mathcal{A}})$ *and* $\mathscr{C}^{\tau'}_{\mathcal{P}} =$

$(Q_{\mathcal{C}}, q_{\mathcal{C}}, F_{\mathcal{C}}, \gamma_{\mathcal{C}})$ *over* $\Sigma_{\mathcal{P}}$ *are called left- and right-hand* trace automata*, respectively, and are defined as follows:*

$$
\begin{aligned}
Q_{\mathcal{A}} &= q_{\mathcal{A}} \uplus (\mathbf{N}(\mathcal{P}) \times \mathbf{T}(\mathcal{P})) \uplus \{\bot\} \uplus \{\top\} \\
Q_{\mathcal{C}} &= q_{\mathcal{C}} \uplus (\mathbf{N}(\mathcal{P}) \times \mathbf{T}(\mathcal{P})) \uplus \{\bot\} \\
F_{\mathcal{A}} &= Q_{\mathcal{A}} \setminus \{q_{\mathcal{A}}\} \\
F_{\mathcal{C}} &= \{\bot\} \cup \{(n,\tau) \mid n \text{ positive and not an axiom, } \tau \text{ terminal for } n\} \\
&\qquad \cup \{(n,\tau) \mid n \text{ positive and an axiom, } \tau \text{ ground}\} \\
\mathsf{dom}(\gamma_{\mathcal{A}}) &= \{(q_{\mathcal{A}}, n_{init}, (n_{init}, \tau))\} \\
&\quad \cup \{((n,\tau_1), n', (n',\tau_2)) \mid (n,n') \in \mathcal{P} \text{ and } (\tau_1,\tau_2) \in \mathsf{dom}(\delta^{\mathcal{A}}_{(n,n')})\} \\
&\quad \cup \{((n,\tau),\tau,\bot) \mid n = \Gamma[\tau] \vdash \Delta \text{ is an axiom}\} \\
&\quad \cup \{((n,\tau), n', \top) \mid (n,n') \in \mathcal{P}\} \cup \{(\top, n, \top) \mid n \in \mathbf{N}(\mathcal{P})\} \\
\gamma_{\mathcal{A}}(q,\sigma,q') &= \begin{cases} k & \text{if } q = (n,\tau_1),\ q' = (n',\tau_2) \text{ and } \delta^{\mathcal{A}}_{(n,n')}(\tau_1,\tau_2) = k \\ 0 & \text{otherwise} \end{cases} \\
\mathsf{dom}(\gamma_{\mathcal{C}}) &= \{(q_{\mathcal{C}}, n_{init}, (n_{init}, \tau'))\} \\
&\quad \cup \{((n,\tau_1), n', (n',\tau_2)) \mid (n,n') \in \mathcal{P} \text{ and } (\tau_1,\tau_2) \in \mathsf{dom}(\delta^{\mathcal{C}}_{(n,n')})\} \\
&\quad \cup \{((n,\tau),\tau,\bot) \mid n = \Gamma \vdash \Delta[\tau] \text{ is positive and an axiom,} \\
&\qquad\qquad \tau \text{ not ground}\} \\
\gamma_{\mathcal{C}}(q,\sigma,q') &= \begin{cases} k & \text{if } q = (n,\tau_1),\ q' = (n',\tau_2) \text{ and } \delta^{\mathcal{C}}_{(n,n')}(\tau_1,\tau_2) = k \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

We say that $\mathscr{C}^{\tau}_{\mathcal{P}}$ *is grounded whenever* τ *is ground for every final state* (n,τ)*. Note, we may assume w.l.o.g. that the automata are* trim*, i.e. every state is reachable from the initial state (accessible) and can reach some final state (co-accessible).*

Definition 14 (Approximate Trace Automata). *Let* $n_{init} = \Gamma[\tau] \vdash \Delta[\tau']$ *be a node in a cyclic proof* $\mathcal{P}$*, then for each* $k > 0$*, we construct the* approximate *left-hand trace automaton* $\mathscr{A}[k]^{\tau}_{\mathcal{P}} = (Q_{\mathcal{A}}, q_{\mathcal{A}}, F_{\mathcal{A}}, \gamma_{\mathcal{A}})$ *using the following:*

$$
\begin{aligned}
Q_{\mathcal{A}} &= q_{\mathcal{A}} \uplus (\mathbf{N}(\mathcal{P}) \times \mathbf{T}(\mathcal{P})) \uplus \{\bot\} \uplus \{\top^i_n \mid 0 < i \leq k, n \in \mathbf{N}(\mathcal{P})\} \\
F_{\mathcal{A}} &= Q_{\mathcal{A}} \setminus \{q_{\mathcal{A}}\} \\
\mathsf{dom}(\gamma_{\mathcal{A}}) &= \{(q_{\mathcal{A}}, n_{init}, (n_{init}, \tau))\} \\
&\quad \cup \{((n,\tau_1), n', (n',\tau_2)) \mid (n,n') \in \mathcal{P} \text{ and } (\tau_1,\tau_2) \in \mathsf{dom}(\delta^{\mathcal{A}}_{(n,n')})\} \\
&\quad \cup \{((n,\tau),\tau,\bot) \mid n = \Gamma[\tau] \vdash \Delta \text{ is an axiom}\} \\
&\quad \cup \{((n,\tau), n', \top^1_{n'}) \mid (n,n') \in \mathcal{P}\} \cup \{(\top^i_n, n, \top^{i+1}_n) \mid 0 < i < k\} \\
&\quad \cup \{(\top^i_n, n', \top^i_n) \mid n' \in \mathbf{N}(\mathcal{P}), n \neq n', 0 < i \leq k\} \\
\gamma_{\mathcal{A}}(q,\sigma,q') &= \begin{cases} k & \text{if } q = (n,\tau_1),\ q' = (n',\tau_2) \text{ and } \delta^{\mathcal{A}}_{(n,n')}(\tau_1,\tau_2) = k \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

Fig. 8 shows the (trimmed) approximate automaton for the proof in Figure 3 with $k = 2$. Accepting states are indicated by a double circle, and the transitions between states show (in parentheses) the nodes or trace values accepted by the transition and (in brackets) their associated weights. We write, e.g., $\neg(3), [0]$ on a transition to indicate that the transition accepts with weight 0 all nodes *except* node (3).

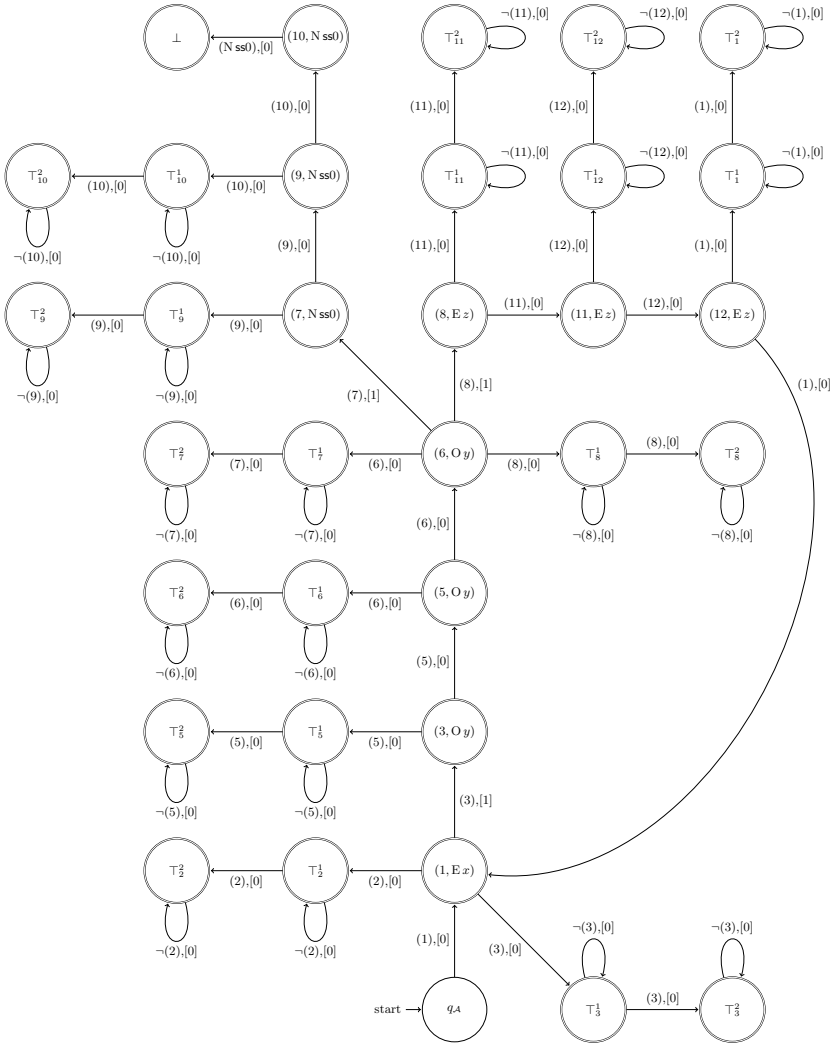


Fig. 8. The approximate trace automaton $\mathscr{A}[2]^{\mathrm{E}\,x}_{\mathcal{P}}$ for the proof of $\mathrm{E}\,x \vdash \mathrm{N}\,x$ in Figure 3