


PAPER

Abstract cyclic proofs

Bahareh Afshari^{1,2} and Dominik Wehr² 

¹Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands

²Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg, Gothenburg, Sweden

Corresponding author: Dominik Wehr; Email: dominik.wehr@gu.se

(Received 20 March 2023; revised 12 February 2024; accepted 26 February 2024; first published online 19 April 2024)

Abstract

Cyclic proof systems permit derivations that are finite graphs in contrast to conventional derivation trees. The soundness of such proofs is ensured by imposing a soundness condition on derivations. The most common such condition is the *global trace condition* (GTC), a condition on the infinite paths through the derivation graph. To give a uniform treatment of such cyclic proof systems, Brotherston proposed an abstract notion of trace. We extend Brotherston's approach into a category theoretical rendition of cyclic derivations, advancing the framework in two ways: first, we introduce *activation algebras* which allow for a more natural formalisation of trace conditions in extant cyclic proof systems. Second, accounting for the composition of trace information allows us to derive novel results about cyclic proofs, such as introducing a Ramsey-style trace condition. Furthermore, we connect our notion of trace to automata theory and prove that verifying the GTC for abstract cyclic proofs with certain trace conditions is PSPACE-complete.

Keywords: Cyclic proof theory; category theory; fixed-point logic; Mu-calculi

1. Introduction

In a cyclic proof system, proofs are finite graphs which represent the ill-founded derivations obtained by unravelling them. Broadly speaking, the logics benefiting from cyclic proofs often feature notions of (co-)induction or fixed points (e.g., Brotherston 2006; Das 2021; Fortier and Santocane 2013; Niwiński and Walukiewicz 1996; Simpson 2017; Sprenger and Dam 2003). Differing from conventional proof systems, cyclic systems typically eschew explicit induction axioms which can instead be simulated by cyclic derivations. Cut-free cyclic proof systems lend themselves well to proof theoretic investigations (e.g., Afshari and Leigh 2017; Afshari et al. 2021; Marti and Venema 2021) and proof search procedures (e.g., Brotherston et al. 2011, 2012; Tellez and Brotherston 2017).

Since cyclic proofs contain infinite paths, their soundness cannot be reduced to the local soundness of their derivation rules. Instead, one usually needs to impose further soundness conditions on these paths. The most common such condition is known as the *global trace condition* (GTC). While the concrete formulation of the trace condition differs between logics, it usually adheres to a certain form: a path satisfies the trace condition if it has a suffix along which some parameter (such as a term or fixed-point quantifier) can be traced and which *progresses* (e.g., decreases or is unfolded) infinitely often.

Motivated by this structural similarity, Brotherston (2006) developed an abstract framework to uniformly represent and reason about cyclic proof systems. The formalism has been used to give

a general proof of the decidability of proof checking for cyclic proofs and to study various transformations of cyclic derivations which maintain the GTC. While Brotherston's notion of trace condition encompasses most cyclic proof systems in the literature, it does not readily capture the common trace condition of μ -calculi, which hold a prominent position in cyclic proof theory.

This article introduces an abstract representation of cyclic proof systems which extends Brotherston's approach. It is formed by an abstract, category theoretical rendition of cyclic derivations and their trace conditions. By replacing Brotherston's notion of single progress points with a more nuanced, algebraic notion of progress, we are able to succinctly express most trace conditions, including those of μ -calculi. Our categorical formalism, in addition, allows for the composition of trace information, yielding two novel results: derivation compression and an alternative soundness condition. Derivation compression allows a cyclic derivation with n simple cycles to be represented as a graph of size $O(n)$, which we believe should yield performance-gains in implementations of, for example, cyclic proof checking algorithms. Second, we give a soundness condition on derivations that induces a proof checking algorithm reliant on Ramsey's theorem instead of automata-theoretic machinery. While a similar condition has been known in the field of program termination (Lee et al. 2001), this is, to the best of our knowledge, the first time such a condition has been considered for cyclic proofs. Furthermore, we reprove some known results to demonstrate applicability of our representation in cyclic proof theory: we show decidability of proof checking and regularisability of ill-founded proofs in finite proof systems via automata theory. Lastly, we show that the proof checking problem for our abstract notion of proof is PSPACE-complete.

Overview. In Section 2, we introduce cyclic proof systems, the modal μ -calculus serving as a concrete example which is frequently revisited throughout the rest of the article. Section 3 presents our abstract notion of trace condition which is then used in Section 4 to introduce an abstract notion of cyclic derivation. In Section 5, we give a soundness condition, equivalent to the GTC, inspired by a result from program termination. Section 6 relates our abstract notion of trace to prevalent uses of automata theory in cyclic proof theory. In Section 7, we give a proof of the PSPACE-completeness of checking the GTC of abstract cyclic proofs with certain kind of trace conditions. We close with a discussion of related and future work in Section 8.

This article is an extended version of Afshari and Wehr (2022) with the following notable additions.

- (1) The notion of trace interpretations is introduced in Section 3 to formally express the connection of concrete and abstract cyclic proofs.
- (2) The connection between our notion of trace categories and the most common uses of automata theory in cyclic proof theory is included in Section 6.
- (3) Section 7 is extended to incorporate a complete proof of PSPACE-hardness. By applying the automata-theoretic results of Section 6, it is shown that proof checking for certain trace categories is in PSPACE.

2. Cyclic Proof Systems

We begin by giving a general account of cyclic proof systems and their associated notion of cyclic proof. We illustrate the definitions by presenting a cyclic proof system for the modal μ -calculus from the literature.

A *tree* is a finite, non-empty set of sequences $T \subseteq \omega^*$ which is prefix closed, that is, if $tn \in T$ for $t \in \omega^*$ and $n \in \omega$ then $t \in T$ as well. We call $t \in T$ a *node* of T and any $tn \in T$ a *child* of t . A node $t \in T$ is a *leaf* of T if it has no children. The *root* of any tree T thus is the empty sequence $\varepsilon \in \omega^*$. A *cyclic tree* is a pair $C = (T, \beta)$ consisting of a tree T and a partial function $\beta: \text{Leaf}(T) \rightarrow T$ mapping (some) leaves of T to nodes of T with $\beta(s) \notin \text{dom}(\beta)$ for all $s \in \text{dom}(\beta)$. Each $s \in \text{dom}(\beta)$ is called a *bud* and $\beta(s)$ its *companion*.

$$\begin{array}{c}
 \text{Ax} \frac{}{p, \neg p} \qquad \text{Wk} \frac{\Gamma}{\Gamma, \varphi} \qquad \vee \frac{\Gamma, \varphi, \psi}{\Gamma, \varphi \vee \psi} \qquad \wedge \frac{\Gamma, \varphi \quad \Gamma, \psi}{\Gamma, \varphi \wedge \psi} \qquad \text{MOD} \frac{\Gamma, \varphi}{\diamond \Gamma, \square \varphi} \\
 \\
 \mu \frac{\Gamma, \varphi[\mu x. \varphi/x]}{\Gamma, \mu x. \varphi} \qquad \nu \frac{\Gamma, \varphi[\nu x. \varphi/x]}{\Gamma, \nu x. \varphi}
 \end{array}$$

Figure 1. Derivation rules of the modal μ -calculus. Γ ranges over finite sets of formulas; $\varphi[\psi/x]$ denotes the standard substitution of ψ for x in φ .

A *derivation system* is a triple $(\text{SEQ}, \mathcal{R}, \rho)$ consisting of a set of *sequents* SEQ , a set of *derivation rules* and a *rule interpretation* $\rho : \mathcal{R} \rightarrow \text{SEQ}^+$ such that for each $r \in \mathcal{R}$, $\rho(r) = (\Gamma, \Delta_1, \dots, \Delta_n) \in \text{SEQ}^{n+1}$ for some $n \in \omega$. In this case, we call Γ the *conclusion* of r and the Δ_i its *premises*. Henceforth, we refer to a derivation system $(\text{SEQ}, \mathcal{R}, \rho)$ simply by \mathcal{R} . An \mathcal{R} -*pre-proof* is a triple $\Pi = (C, \lambda, \delta)$ consisting of a cyclic tree $C = (T, \beta)$ together with a labelling $\lambda : T \rightarrow \text{SEQ}$ such that $\lambda(t) = \lambda(\beta(t))$ for every $t \in \text{dom}(\beta)$. $\delta : T \setminus \text{dom}(\beta) \rightarrow \mathcal{R}$ is a function with $\rho(\delta(t)) = (\Gamma, \Delta_1, \dots, \Delta_n)$, $\lambda(t) = \Gamma$ and $\lambda(ti) = \Delta_i$ where $\text{Chld}(t) = \{t_1, \dots, t_n\}$ for each $t \in \text{dom}(\delta)$. The sequent $\lambda(\varepsilon)$ is called the *endsequent* of Π .

We denote by $\text{PP}(\mathcal{R})$ the set of \mathcal{R} -pre-proofs. A *cyclic proof system* is a tuple $(\text{SEQ}, \mathcal{R}, \rho, \text{PFS})$ consisting of a derivation system $(\text{SEQ}, \mathcal{R}, \rho)$ and a set $\text{PFS} \subseteq \text{PP}(\mathcal{R})$ called \mathcal{R} -*proofs*. A pre-proof is said to satisfy the *soundness condition* of \mathcal{R} if it is an \mathcal{R} -proof. An \mathcal{R} -proof with endsequent Γ is called a *proof of* Γ . We extend the naming convention for derivation systems to cyclic derivation systems, referring to $(\text{SEQ}, \mathcal{R}, \rho, \text{PFS})$ by \mathcal{R} .

To illustrate these notions, we present a cyclic proof system for the modal μ -calculus. It will also serve as an example motivating the abstract definitions of Sections 3 and 4. This presentation of the system is taken from Afshari and Leigh (2017) and is an adaptation of the tableaux of Nikiński and Walukiewicz (1996). The choice of logic for this example is secondary, the main focus being the cyclic aspects of the proof system.

For a set PROP of propositional letters and a countable set VAR of variables, the μ -formulas are given by the following grammar:

$$\varphi \in \text{FORM} ::= p \mid \neg p \mid x \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \square \varphi \mid \diamond \varphi \mid \mu x. \varphi \mid \nu x. \varphi \quad p \in \text{PROP}, x \in \text{VAR}$$

If $x, y \in \text{VAR}$ occur in φ , we say x *subsumes* y , writing $x <_{\varphi} y$, if $\sigma y. \psi$ occurs as a subformula of φ for some $\sigma \in \{\mu, \nu\}$ and ψ , and furthermore x is free in $\sigma y. \psi$. If the relation $<_{\varphi}$ is a strict preorder, we call φ *well-named*. In the remainder of this article, we assume all μ -formulas are well-named. This is a reasonable restriction as any μ -formula is α -equivalent to a well-named one. Any μ -formula is positive in all variables. The fixed-point formulas $\mu x. \varphi$ and $\nu x. \varphi$ denote, respectively, the least and greatest fixed point of the semantic counterpart to the function $x \mapsto \varphi(x)$. These are well defined by the observation on positivity and the Knaster–Tarski theorem. The semantics of the modalities and connectives are as in the modal logic K.

The derivation rules of the cyclic μ -calculus are given in Fig. 1. The sequents of this calculus are finite sets of μ -formulas. Not all pre-proofs derive valid endsequents. For example, $\mu x. \square x$ is invalid but is concluded by the μ -pre-proof given in Fig. 2. It is thus necessary to impose an additional soundness condition which delineates μ -proofs from μ -pre-proofs.

A *branch* through a pre-proof $((T, \beta), \lambda, \delta)$ is an infinite sequence $t \in T^{\omega}$ such that $t_0 = \varepsilon$ and for any t_i , either (a) $t_{i+1} \in \text{Chld}(t_i)$ or (b) $t_i \in \text{dom}(\beta)$ and $t_{i+1} = \beta(t_i)$. This induces the sequence $(\Gamma_i)_{i \in \omega} \in \text{SEQ}^{\omega}$ with $\Gamma_i := \lambda(t_i)$ and the partially defined $r : \omega \rightarrow \mathcal{R}$ with $r_i := \delta(t_i)$, both of which we use interchangeably with $t \in T^{\omega}$ to denote a branch.

Given a branch $(\Gamma_i)_{i \in \omega}$ through a μ -pre-proof, a formula $\varphi' \in \Gamma_{i+1}$ is called a *precursor* of $\varphi \in \Gamma_i$, written $\varphi' \leftarrow_i \varphi$, if $t_{i+1} \in \text{Chld}(t_i)$ and either φ is principal in r_i , that is, φ is ‘altered by r_i ’, and φ' is one of the residual formulas, or φ is not principal in r_i and $\varphi = \varphi'$. If $t_i \in \text{dom}(\beta)$ and $\varphi = \varphi'$ then $\varphi' \leftarrow_i \varphi$ as well. A sequence of formulas $(\varphi_i)_{i \in \omega}$ is called a *trace* along $(\Gamma_i)_{i \in \omega}$ if

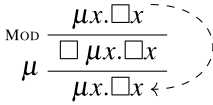


Figure 2. A μ -pre-proof of an invalid μ -formula. The dashed arrow represents the bud-companion relation β .

$\varphi_{i+1} \leftarrow_i \varphi_i$ for all $i \in \omega$. It is easily observed that the subsumption order is preserved along traces, that is, $\leftarrow_{\varphi_{i+1}} \subseteq \leftarrow_{\varphi_i}$ whenever $\varphi_{i+1} \leftarrow_i \varphi_i$, which is why we henceforth associate with each trace $(\varphi_i)_{i \in \omega}$ a *global subsumption order* $\leftarrow_{\varphi} := \leftarrow_{\varphi_0} = \bigcup_{i \in \omega} \leftarrow_{\varphi_i}$.

Let $(\Gamma_i)_{i \in \omega}$ be a branch through a μ -pre-proof and $(\varphi_i)_{i \in \omega}$ a trace along it. The trace is called a ν -trace if there exists an $x \in \text{VAR}(\varphi_0)$ such that $\varphi_i = \nu x. \psi$ and $\varphi_{i+1} = \psi[\varphi_i/x]$ for infinitely many $i \in \omega$, and furthermore for any $y \in \text{VAR}(\varphi_0)$ if there are infinitely many $\varphi_i = \mu y. \theta$ then $x \leftarrow_{\varphi} y$. In other words, there is a greatest fixed-point variable x that occurs infinitely often and subsumes all infinitely occurring μ -variables. A μ -proof is a μ -pre-proof that satisfies the *global trace condition*, that is, every infinite branch has a ν -trace. As the unique branch through the pre-proof in the example of Fig. 2 does not have a ν -trace, it fails to be a proof.

At this point, we want to clearly distinguish between a ‘trace condition’ and a ‘global trace condition’. A trace condition is a specification of which traces along infinite branches of a pre-proof are considered progressing. A global trace condition is a certain type of condition on cyclic proofs, usually formulated via a trace condition, used to ensure soundness of proofs. Alternative soundness conditions have been considered, such as the reset condition (Jungteerapanich 2009), induction orders (Sprenger and Dam 2003) and trace manifolds (Brotherston 2006). These soundness conditions are often still defined in reference to a trace condition, or at least its implicit notion of progress. It is, therefore, possible for two differently formulated soundness conditions for a certain derivation system to share the same underlying trace condition. For example, this is the case for the GTC of the μ -proofs specified above and the reset proof system for the modal μ -calculus given by Stirling (2013).

3. Abstracting the Trace Condition

In this section, we demonstrate our formalism for capturing the trace conditions of cyclic proof systems. It encompasses two levels of abstraction: The notion of a *trace category* which captures what it means to be a trace condition, and a family of concrete trace categories, generated by the notion of an *activation algebra*.

An abstract notion of trace condition requires an abstract notion of branches through a proof. Working in a category theoretical framework, we observe that branches are infinite sequences of rule applications and identify them with infinite sequences of morphisms called *paths*. A trace condition is then a condition on such paths which is invariant under certain path transformations.

A *semi-category* is a category which may not have (all) identity morphisms. That is, a semi-category \mathcal{C} consists of a collection of objects $\text{Ob}(\mathcal{C})$ and collections of morphisms $\text{HOM}_{\mathcal{C}}(X, Y)$ between each pair of objects $X, Y \in \text{Ob}(\mathcal{C})$. There is a composition $\circ : \text{PX}, Y, Z \in \text{Ob}(\mathcal{C}). \text{HOM}_{\mathcal{C}}(Y, Z) \times \text{HOM}_{\mathcal{C}}(X, Y) \rightarrow \text{HOM}_{\mathcal{C}}(X, Z)$ which is associative. A *semi-functor* is a semi-category homomorphism. That is, for semi-categories \mathcal{C}, \mathcal{D} , a semi-functor $F : \mathcal{C} \rightarrow \mathcal{D}$ consists of a map on objects $F_0 : \text{Ob}(\mathcal{C}) \rightarrow \text{Ob}(\mathcal{D})$ and a further map F_2 on morphisms of \mathcal{C} such that for $f : X \rightarrow Y$ we have $F_1(f) : F_0(X) \rightarrow F_0(Y)$. F_1 distributes over the composition operation, that is, $F_1(g \circ f) = F_1(g) \circ F_1(f)$. As is also common for standard functors, we denote both F_0 and F_1 by F .

The standard $<$ -ordering of the natural numbers ω induces a semi-category whose objects are the natural numbers and in which there is a (unique) morphism between n and m , denoted ‘ $n < m : n \rightarrow m$ ’, if $n < m$. The \leq -ordering induces an analogous proper category. In this article, we denote both of these categories by ω ; which one is meant will be clear from the context.

A path through a category \mathcal{T} is a functor $P: \omega \rightarrow \mathcal{T}$. Given $P, P': \omega \rightarrow \mathcal{T}$, we call P a subpath of P' , written $P \subseteq P'$, if there is a semi-functor $S: \omega \rightarrow \omega$ between the ω -semi-categories such that $P = P' \circ S$. The transitive, symmetric closure of \subseteq is denoted \sim . This means $P \subseteq P'$ holds if P' can be transformed into the P by applying a combination of two path transformations: (1) discarding a finite prefix of path P' , for example, by taking $S(m) := m + n$. (2) Composing morphisms along P' . For example, if P' is of the form

$$X_0 \xrightarrow{R_0^0} X_0^1 \xrightarrow{R_0^1} \dots X_0^{n_0} \xrightarrow{R_0^{n_0}} X_1^0 \xrightarrow{R_1^0} \dots X_1^{n_1} \xrightarrow{R_1^{n_1}} X_2^0 \xrightarrow{R_2^0} \dots X_2^{n_2} \xrightarrow{R_2^{n_2}} \dots$$

then $P \subseteq P'$ for the path P below

$$X_0^0 \xrightarrow{R_0^{n_0} \circ \dots \circ R_0^0} X_1^0 \xrightarrow{R_1^{n_1} \circ \dots \circ R_1^0} X_2^0 \xrightarrow{R_2^{n_2} \circ \dots \circ R_2^0} \dots$$

witnessed by $S(i) := \sum_{j < i} (n_j + 1)$.

Definition 3.1. Given a category \mathcal{T} , a trace condition is a predicate on paths invariant under \sim . That is, for any two paths $P \sim P'$ the trace condition holds for P if and only if it holds for P' .

A trace category is a category equipped with a trace condition.

Note that the notion of trace category in the above definition is unrelated to the ‘traced monoidal categories’ of Joyal et al. (1996).

Remark 3.1. Any trace condition is invariant under taking suffixes and composition. All concrete trace conditions in the literature are closed under taking suffixes, making closure under suffixes a natural criterion for identifying trace conditions.

Composition of rules along branches is not part of cyclic proof systems and hence there is a priori no precedent for it. The cumulative nature of trace conditions, however, suggests that composition of traces should not invalidate them. Indeed, all trace categories which we use to model concrete trace conditions from the literature have a trace condition closed under composition. Furthermore, it is precisely this closure condition that has proven instrumental in deriving new results in our framework.

The remainder of this section is concerned with defining a family of concrete trace categories which can model the trace conditions of many cyclic proof systems we know of.

Definition 3.2. An activation algebra is a tuple $\mathcal{A} = (A, \leq, \vee, 0, \alpha)$ consisting of a finite semilattice $(A, \leq, \vee, 0)$ and an activation element $\alpha \in A$ where $0 \neq \alpha$. We often write \mathcal{A} to refer to the carrier set A .

Definition 3.3. Let \mathcal{A} be an activation algebra. The \mathcal{A} -activated trace category $\mathcal{T}_{\mathcal{A}}$ has as its objects the finite sets. The morphisms between sets X, Y are relations $R \subseteq X \times \mathcal{A} \times Y$. The identities are $1_X := \{(x, 0, x) \mid x \in X\}$. We often write $xR^a y$ to mean $(x, a, y) \in R$. Given morphisms $R: X \rightarrow Y, R': Y \rightarrow Z$, their composition is specified by $(x, c, z) \in R' \circ R$ iff

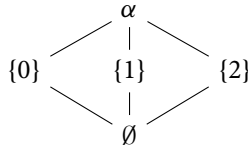
$$\exists y \in Y \exists a, b \in \mathcal{A}. (x, a, y) \in R \text{ and } (y, b, z) \in R' \text{ and } a \vee b = c.$$

A path $P: \omega \rightarrow \mathcal{T}_{\mathcal{A}}$ satisfies the trace condition if there exists a subpath $P' \subseteq P$ and a sequence $\sigma: \Pi i \in \omega. P'(i)$ along it such that $\sigma_i P'(i < i + 1)^\alpha \sigma_{i+1}$ for all $i \in \omega$.

Example 3.1. The simplest example of an activation algebra is given by the simplest semilattice of at least two elements: the binary Boolean algebra $\mathbb{B} = \{\top, \perp\}$. The choice $\alpha := \top$ is forced as necessarily $\perp = 0 \neq \alpha$. This activation algebra is implicit in Brotherston’s (2006) abstract notion of trace and suffices to model most trace conditions in the literature.

Example 3.2. The activation algebra used to formalise the trace condition of the modal μ -calculus is the three-value failure algebra $\mathbb{F} := (\{0, 1, 2\}, \leq, \vee, 0, 1)$. In this algebra, the value 2 can be used to represent a ‘failure’ state (see the proof of Proposition 3.1).

Example 3.3. More complex examples are the ‘ k out of n ’ algebras for $0 < k \leq n$ given by $\binom{n}{k} := (A, \leq, \vee, \emptyset, \alpha)$ where $A := \{X \subseteq n \mid |X| < k\} \cup \{\alpha\}$, the order \leq is such that $X \leq Y$ iff $X \subseteq Y$ for $X, Y \subseteq n$, and $a \leq \alpha$ for all $a \in A$. More concretely, observe the Hasse diagram of $\binom{3}{2}$.



The idea behind $\binom{n}{k}$ is to view the singleton sets $\{i\}$ as ‘events’ which can occur along a trace. To achieve activation, k distinct ‘events’ need to take place along a segment. As opposed to \mathbb{B} and \mathbb{F} , we are not yet aware of a cyclic proof system whose trace condition is best expressed in terms of some $\binom{n}{k}$ (except of course $\binom{1}{1}$ which the same as \mathbb{B}). We conjecture that cyclic proof systems whose trace condition is naturally modelled in some non-trivial $\binom{n}{k}$ would require a kind of fairness condition of their progressing traces.

Lemma 3.1. The trace condition in Definition 3.3 is well defined, that is, it fulfils the invariance condition of Definition 3.1.

Proof. It suffices to prove invariance under \subseteq . Let $P \subseteq Q$ with $P = Q \circ S$. Suppose P satisfies the trace condition meaning there exists $P' \subseteq P$ with $P' = P \circ S'$ and a validating sequence σ . Then, $P' \subseteq Q$ via $P' = Q \circ S \circ S'$, meaning σ is a validating sequence through a subpath of Q as well.

For the converse direction, let $P = Q \circ S$ and suppose Q satisfies the trace condition as witnessed by $Q' = Q \circ S'$ and a sequence $\sigma : \Pi i \in \omega. Q'(i)$. It remains to show that there is $P' \subseteq P$ and a suitable sequence $\sigma'' : \Pi i \in \omega. P'(i)$ along it. By fixing $b := S'(0)$ and analysing Definition 3.3, one can conclude there are two sequences $\sigma' : \Pi i \in \omega. Q(b + i)$ and $a : \omega \rightarrow \mathcal{A}$ such that $\sigma'_i Q(b + i < b + i + 1)^{a_i} \sigma'_{i+1}$ with $a_i \leq \alpha$. For $i_j := S'(j) - b$, we then have $\sigma'_{i_j} = \sigma_j$ and $\bigvee_{i=i_j}^{i < i_{j+1}} a_i = \alpha$. Now construct the following sequence:

$$k_0 := \text{least } i \in \omega \text{ such that } S(i) \geq b$$

$$k_{n+1} := \text{least } i > k_n \text{ s.t. } S(k_n) \leq S'(j) < S'(j + 1) \leq S(i) \text{ for some } j \in \omega$$

We claim setting $S''(i) := S(k_i)$ induces the desired subpath $P' := Q \circ S'' \circ P$ with $P'(i) = P(k_i)$ as witnessed by $\sigma'' : \Pi i \in \omega. P'(i)$ given by $\sigma''_i := \sigma'_{S(k_i)}$. For this, we need to check that $\sigma''_{i+1} P'(i < i + 1)^\alpha \sigma''_{i+1}$. Let $j \in \omega$ be such that $S''(i) \leq S'(j) < S'(j + 1) \leq S''(i + 1)$. Then, $P'(i < i + 1) = Q(b + i_{j+1} \leq S(k_{i+1})) \circ Q(b + i_j < b + i_{j+1}) \circ Q(S(k_i) \leq b + i_j)$, meaning

$$\left(\begin{array}{c} \sigma''_i, \underbrace{\bigvee_{l=b+i_{j+1}}^{l < S(k_{i+1})} a_{l-b}}_{\leq \alpha} \vee \underbrace{\bigvee_{l=i_j}^{l < i_{j+1}} a_l}_{=\alpha} \vee \underbrace{\bigvee_{l=S(k_i)}^{l < b+i_j} a_{l-b}}_{\leq \alpha}, \underbrace{\sigma'_{k_{i+1}-b}}_{=\sigma''_{i+1}} \end{array} \right) \in P'(i < i + 1)$$

and thus $(\sigma''_i, \alpha, \sigma''_{i+1}) \in P'(i < i + 1)$ as desired. □

We now proceed to demonstrate how trace categories can be used to specify the trace conditions and, thereby, the GTCs of cyclic proof systems. Fix a derivation system $\mathcal{R} = (\text{SEQ}, \mathcal{R}, \rho)$ and a trace category \mathcal{T} . A trace interpretation $\iota : \mathcal{R} \rightarrow \mathcal{T}$ consists of a function $\iota : \text{SEQ} \rightarrow \text{Ob}(\mathcal{T})$ mapping sequents to their trace sets and for each rule $r \in \mathcal{R}$ with $\rho(r) = (\Gamma, \Delta_1, \dots, \Delta_n)$ a morphism

$r_i : \iota(\Gamma) \rightarrow \iota(\Delta_i)$ for each $1 \leq i \leq n$ called a *trace map*. Let (C, λ, δ) be a pre-proof and $t \in T^\omega$ be a branch through C . Its corresponding path $P : \omega \rightarrow \mathcal{T}$ is defined as follows:

$$P(i) := \iota(\lambda(\pi_i)) \quad P(i < i + 1) := \begin{cases} r_j : \iota(\lambda(\pi_i)) \rightarrow \iota(\lambda(\pi_{i+1})) & \pi_i \notin \text{Leaf}(T) \text{ and } \pi_{i+1} = \pi_j \\ 1_{P(i)} & \pi_i \in \text{dom}(\beta) \end{cases}$$

This induces a cyclic proof system $\iota(\mathcal{R}) := (\text{SEQ}, \mathcal{R}, \rho, \text{PFS})$ in which PFS contains those \mathcal{R} -pre-proofs for which every induced path $P : \omega \rightarrow \mathcal{T}$ through them satisfies the trace condition of \mathcal{T} .

In the following, we demonstrate how to specify the trace condition for the modal μ -calculus by a trace interpretation $\iota : \mu \rightarrow \mathbb{F}$.

Definition 3.4. *The trace interpretation $\iota : \mu \rightarrow \mathcal{T}_{\mathbb{F}}$ is given by $\iota(\Gamma) := \{(\varphi, x) \mid \varphi \in \Gamma \text{ and } x \in \text{VAR}_v(\varphi)\}$ in which $\text{VAR}_v(\varphi) := \{x \mid x \text{ is bound by } v \text{ in } \varphi\}$. For each $r \in \mu$ with $\rho(r) = (\Gamma, \Delta_1, \dots, \Delta_n)$, the trace maps $r_i : \iota(\Gamma) \rightarrow \iota(\Delta_i)$ are defined by $r_i := \{((\varphi, x), a^*, (\varphi', x)) \mid \varphi' \leftarrow_r^i \varphi\}$ where a^* is defined by:*

$$a^* := \begin{cases} 2, & \text{if } r \text{ instance of } \mu, \varphi = \mu y.\theta, \varphi' = \theta[\mu y.\theta/y] \text{ and } y <_\varphi x, \\ 1, & \text{if } r \text{ instance of } \nu, \varphi = \nu x.\theta, \varphi' = \theta[\nu x.\theta/x], \\ 0, & \text{otherwise.} \end{cases}$$

Proposition 3.1. *The notion of μ -proofs and that induced by $\iota : \mu \rightarrow \mathbb{F}$ coincide.*

Proof. It suffices to prove that a branch $(\Gamma_i)_{i \in \omega}$ through a μ -pre-proof has a ν -trace if and only if its induced path $P : \omega \rightarrow \mathcal{T}_{\mathbb{F}}$ satisfies the trace condition of $\mathcal{T}_{\mathbb{F}}$.

Suppose $(\Gamma_i)_{i \in \omega}$ has a ν -trace $(\varphi_i)_{i \in \omega}$. Then there exists $x \in \bigcap_{i \in \omega} \text{VAR}(\varphi_i)$ bounded by a ν -quantifier and an increasing sequence $(j_i)_{i \in \omega}$ such that

- (i) $\varphi_{j_i} = \nu x.\psi$ and $\varphi_{j_{i+1}} = \psi[\varphi_{j_i}/x]$, and
- (ii) no formula $\mu y.\theta$ with $y <_\varphi x$ is unfolded along $(\varphi_i)_{i > j_0}$.

The subpath $P \circ S \subseteq P$ induced by $S(i) := j_i$ and the sequence $\sigma_i := (\varphi_{j_i}, x)$ witness that P satisfies the trace condition: clearly always $(\sigma_i, a, \sigma_{i+1}) \in P(j_i, j_{i+1})$ for some $a \in \mathbb{F}$. We know $1 \leq a$ since between Γ_{j_i} and $\Gamma_{j_{i+1}}$, $\nu x.\psi$ is unfolded, and further $a < 2$, as no $\mu y.\theta$ with $y <_\varphi x$ is unfolded after j_0 , yielding $a = 1$ as desired.

Conversely, suppose P satisfied the trace condition. Then there exist $S : \omega \rightarrow \omega$ and $\sigma : \Pi i \in \omega. P(S(i))$ such that $(\sigma_i, 1, \sigma_{i+1}) \in P(S(i) < S(i+1))$ for every $i \in \omega$. Necessarily, $\sigma_i = (\varphi_i, x)$ for some fixed ν -variable x and $\varphi_i \in \Gamma_{S(i)}$. Furthermore, because the activation algebra element along that trace is precisely 1, we can conclude that between $\Gamma_{S(i)}$ and $\Gamma_{S(i+1)}$:

- (i) the formula $\nu x.\psi$ corresponding to x in φ_i is unfolded (as $1 \leq \alpha$),
- (ii) no $\mu y.\theta$ with $y <_\varphi x$ is unfolded (as $\alpha < 2$).

By scrutinising the derivation rules, it can be deduced that $\varphi_{S(0)}$ can be ‘traced back’ to some $\varphi \in \Gamma_0$ and subsequently completed into a trace $(\varphi'_i)_{i \in \omega}$ along $(\Gamma_i)_{i \in \omega}$ with $\varphi'_{S(i)} = \varphi_i$. By the observations above, this must be a ν -trace. □

Remark 3.2. *We have claimed that the trace condition given for the modal μ -calculus in Section 2 cannot be represented naturally in terms of the activation algebra \mathbb{B} . From the description in Section 2, it is clear that any natural representation of it as $\iota : \mu \rightarrow \mathcal{T}_{\mathbb{B}}$ must coincide with the trace interpretation given in Definition 3.4 on trace objects and relations. That is, $\iota(\Gamma)$ must be the set $\{(\varphi, x) \mid \varphi \in \Gamma \text{ and } x \in \text{VAR}_v(\varphi)\}$ (or equivalent) and for a derivation rule with $\rho(r) = (\Gamma, \Delta_1, \dots, \Delta_n)$ any $((\varphi, x), a, (\varphi', y)) \in r_i : \iota(\Gamma) \rightarrow \iota(\Delta_i)$ should be such that $\varphi' \leftarrow_r^i \varphi$ and $x = y$. It thus remains to describe how to assign the values a in the triples in r_i . Clearly, non-unfolding*

rules should assign $a = 0$ and v -unfoldings $a = 1$. For μ -unfoldings, if such an unfolding occurs infinitely often, the trace should be ‘spoiled’. Neither an assignment of $a = 0$ nor $a = 1$ can model this behaviour. The assignment of $a = f$ in Definition 3.4, on the other hand, succinctly takes care of this case.

For further examples of modelling of the trace conditions of cyclic proof systems in trace categories, in particular, those of cyclic arithmetic (Simpson 2017), $\text{HFL}_{\mathbb{N}}$ (Kori et al. 2021) and Grzegorzczuk modal logic (Savateev and Shamkanov 2021), we refer the reader to Wehr (2021).

We close this section by stressing that the purpose of activation algebras is to specify trace conditions of cyclic proof systems in a *natural* manner. Indeed, if naturality is of no concern, the following result shows that the trace category $\mathcal{T}_{\mathbb{B}}$, or equivalently the formalism of Brotherston (2006), is sufficient to model the majority of trace conditions from the literature, including that of the modal μ -calculus.

Concretely, the notion of *naturality* we allude to here is embodied in the fact that the information of traces may be separated into two parts: the elements of the trace sets signify *which objects* is being tracked, while the elements of the activation algebra describe *how progress* of these trace objects is detected. Indeed, Theorem 3.1 is achieved by collapsing this separation. An example of the importance of maintaining this distinction is given by (Leigh and Wehr 2023): the article describes how to generate so-called reset proof systems for cyclic proof systems whose trace condition is specified in terms of a trace interpretation into some \mathcal{A} -activated category. One of the examples considered there is the modal μ -calculus. The trace interpretation in terms of \mathbb{F} (as in Definition 3.4) leads to a very natural reset system, whereas the reset system induced by the trace interpretation into \mathbb{B} given by Theorem 3.1 would be highly artificial.

Theorem 3.1. *For any activation algebra \mathcal{A} , there exists a function I mapping objects of $\mathcal{T}_{\mathcal{A}}$ to objects of $\mathcal{T}_{\mathbb{B}}$ and maps $R : X \rightarrow Y$ in $\mathcal{T}_{\mathcal{A}}$ to maps $I(R) : I(X) \rightarrow I(Y)$ in $\mathcal{T}_{\mathbb{B}}$. Furthermore, associate to each path $P : \omega \rightarrow \mathcal{T}_{\mathcal{A}}$ a path $\hat{P} : \omega \rightarrow \mathcal{T}_{\mathbb{B}}$ given by $\hat{P}(i) := I(P(i))$ and $\hat{P}(i < i + 1) := I(P(i < i + 1))$. Then P satisfies the trace condition iff \hat{P} does.*

Proof. Writing $\mathcal{A} = (A, \leq, \vee, 0, \alpha)$, take $I(X) := X \times A$ and, for $R : X \rightarrow Y$,

$$I(R) := \{((x, a), \perp, (y, a \vee b)) \mid a \in A, xR^b y\} \cup \{((x, a), \top, (y, 0)) \mid a \in A, xR^b y, a \vee b = \alpha\}$$

Suppose that P satisfied the trace condition. That means there is a subpath $P \circ S$ and a sequence $\sigma : \Pi i \in \omega. P(S(i))$ such that $\sigma_i P(S(i))^\alpha \sigma_{i+1}$. We claim that $\hat{P} \circ S$ is the witnessing subpath of P with the sequence $\hat{\sigma} : \Pi i \in \omega. \hat{P}(S(i))$ given by $\hat{\sigma}_i := (\sigma_i, 0)$. We prove that $\hat{\sigma}_i \hat{P}(S(i < i + 1))^\top \hat{\sigma}_{i+1}$: There must be $\sigma_i = x_0, \dots, x_n = \sigma_{i+1}$ and a_0, \dots, a_{n-1} for $n := S(i + 1) - S(i)$ and $b := S(i)$ such that $x_j P(b + j)^{a_j} x_{j+1}$ and $\alpha = \bigvee_{j < n} a_j$. Then define $\hat{x}_j := (x_j, \bigvee_{k < j-1} a_k)$ and observe that $\hat{\sigma}_i = \hat{x}_0 \hat{P}(b)^\perp \hat{x}_1 \hat{P}(b + 1)^\perp \dots \hat{x}_{n-1} \hat{P}(b + n - 1)^\top (x_n, 0) = \hat{\sigma}_{i+1}$ because $\hat{x}_{n-1} = (x_{n-1}, \bigvee_{k < n-2} a_k)$ and $(\bigvee_{k < n-2} a_k) \vee a_{n-1} = \alpha$.

Conversely, suppose that \hat{P} satisfied the trace condition. Then there is a subpath $\hat{P} \circ S$ and a sequence $\hat{\sigma} : \Pi i \in \omega. \hat{P}(S(i))$ such that $\hat{\sigma}_i \hat{P}(S(i < i + 1))^\top \hat{\sigma}_{i+1}$. There must be $\sigma_i = (x_0, a_0), \dots, (x_n, a_n) = \sigma_{i+1}$ and $b_0, \dots, b_{n-1} \in \mathbb{B}$ for $n := S(i + 1) - S(i)$ and $k := S(i)$ such that $(x_j, a_j) \hat{P}(k + j)^{b_j} (x_{j+1}, a_{j+1})$ and $\top = \bigvee_{j < n} b_j$. Per construction, the latter means there is some $J < n$ such that $(x_j, a_j) \hat{P}(k + J)^\top (x_{j+1}, 0)$. Consider analogous $\sigma_{i+1} = (x'_0, a'_0), \dots, (x'_m, a'_m) = \sigma_{i+2}$ and $b'_0, \dots, b'_{m-1} \in \mathbb{B}$ such that $(x'_j, a'_j) \hat{P}(k' + j)^{b'_j} (x'_{j+1}, a'_{j+1})$ with $k' := S(i + 1)$, yielding an analogous J' such that $(x'_{j'}, a'_{j'}) \hat{P}(k + J')^\top (x_{j'+1}, 0)$. Intuitively, this means that the a_j and a'_j between $(x_{j+1}, 0)$ and $(x'_{j'+1}, 0)$ ‘accumulate’ activation algebra elements up to α . That is, there must be $c_0, \dots, c_{l-1} \in A$ with $l := n - (J + 1) + (J' + 1)$ such that $\alpha = \bigvee_{j < l} c_j$ and $x_{j+1} P(k + J + 1 < k + J + 2)^{c_0} \dots P(k + n - 1 < k + n)^{c_{n-J-1}} x_n = x'_0 P(k' < k' + 1)^{c_{n-J}} \dots P(k' + J' < k' + J' + 1)^{c_{l-1}} x'_{j'+1}$.

In other words, $x_{j+1}P(k + J + 1 < k' + J' + 1)^\alpha x'_{j'+1}$, which can be extended to $x_0P(k < k' + m)^\alpha x'_m$ by observing that necessarily $a_j < \alpha$ and $a'_j < \alpha$ for all $j < n$ and $j < m$, respectively, for the original trace along \hat{P} to be successful. Based on this observation, we may conclude that the sub-path $P \circ S'$ of P with $S'(i) = S(2i)$ satisfies the trace condition for the sequence $\sigma' : \Pi i \in \omega.P(S'(i))$ with $\sigma'_i := \pi_1(\sigma_{2i})$, that is, $\pi_1(\sigma_{2i})P(S(2i < 2i + 2))^\alpha \pi_1(\sigma_{2i+2})$ as demonstrated above. \square

Remark 3.3. *The statement of Theorem 3.1 is given in terms of functions, rather than functors. This is so because the resulting functions fail to be functors on both accounts: preserving identities and distributing over composition. The failure of identity preservation is easily observed. Notice that any $I(1_X)$ will contain triples of the form $((x, \alpha), \top, (x, 0))$ which the identities of $\mathcal{T}_{\mathbb{B}}$ do not contain. This issue could be alleviated by taking the definition to be*

$$I(R) := \{((x, a), \perp, (y, a \vee b)) \mid a \in A, xR^b y\} \\ \cup \{((x, a), \top, (y, 0)) \mid a \in A \setminus \{\alpha\}, xR^b y, a \vee b = \alpha\}$$

instead, that is, explicitly excluding that kind of transition. However, as this still does not resolve the distributivity over composition, we chose to forgo this in favour of a simpler definition and proof.

The failure of distributivity over composition is a bit more subtle. For this, suppose there were $a < b < c < \alpha \in A$ such that $a \vee b = \alpha$ and consider $R := \{(\star, b, \star)\}$ and $R' := \{(\star, c, \star)\}$. Then clearly, $(\star, a)I(R)^\top (\star, 0)I(R')^\perp (\star, c)$. However, as $R' \circ R = \{(\star, b \vee c, \star)\}$, the only two transitions possible via $I(R' \circ R)$ are $(\star, a)I(R' \circ R)^\perp (\star, \alpha)$ and $(\star, a)I(R' \circ R)^\top (\star, 0)$. Thus, $I(R') \circ I(R) \neq I(R' \circ R)$.

4. Abstract Cyclic Derivations

This section combines the abstract notions of branches, traces and the trace condition into an abstract presentation of cyclic derivations. Abstract cyclic derivations (ACDs) are pairs (C, Tr) of cyclic trees C and maps Tr which ‘decorate’ the edges of C with maps of a trace category. Such an ACD is considered to be a proof if all paths which can be generated by traversing C satisfy the trace condition imposed by the trace category. To express these ideas in a category theoretical manner, we begin by giving a categorical representation of cyclic trees, by defining the (semi-)category induced by the finite paths through these trees.

Fix a cyclic tree $C = (T, \beta)$. The finite paths from s to t , denoted by $\text{Path}_C(s, t) \subseteq T^+$, are defined as the smallest sets satisfying the following three conditions:

- (1) For any $s \in T$, we have $s \in \text{Path}_C(s, s)$,
- (2) For any $t, u \in T$ with u child of t , if $p \in \text{Path}_C(s, t)$ then $pu \in \text{Path}_C(s, u)$,
- (3) For any $t \in \text{dom}(\beta)$, if $p \in \text{Path}_C(s, t)$ then $p\beta(t) \in \text{Path}_C(s, \beta(t))$.

The category \mathcal{P}_C of paths through $C = (T, \beta)$ has the nodes of T as its objects and fixes $\text{Hom}_{\mathcal{P}_C}(s, t) = \text{Path}_C(s, t)$. The identities are $1_s = s : s \rightarrow s$ and, given morphisms $p : s \rightarrow t$ and $q : t \rightarrow u$, we define $q \circ p = pq'$ where $q = tq'$ for $q' \in T^*$. The semi-category \mathcal{P}_C^S of progressing paths is the same as \mathcal{P}_C except that $\text{Hom}_{\mathcal{P}_C^S}(s, t) := \{p \in \text{Path}(s, t) \mid |p| > 1\}$.

The informal notion of ‘decorating a cyclic tree with trace information’ can thus be expressed as a functor.

Definition 4.1. *An abstract cyclic derivation over a trace category \mathcal{T} is a pair (C, Tr) consisting of a cyclic tree $C = (T, \beta)$ and a functor $\text{Tr} : \mathcal{P}_C \rightarrow \mathcal{T}$ such that for any $s \in \text{dom}(\beta)$, $\text{Tr}(s) = \text{Tr}(\beta(s))$ and $\text{Tr}(s\beta(s)) = 1_{\text{Tr}(s)}$.*

We delineate the abstract cyclic proofs from mere ACDs via a GTC.

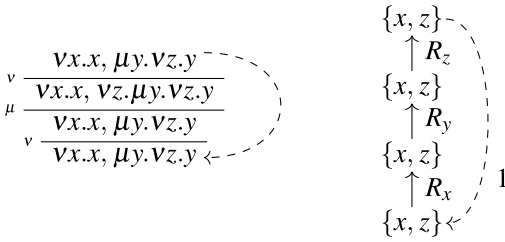


Figure 3. A μ -proof and its corresponding ACD, discussed in Example 4.1.

Definition 4.2. Let D be an ACD given by $(C, Tr: \mathcal{P}_C \rightarrow \mathcal{T})$. A path through D is a path $P: \omega \rightarrow \mathcal{T}$ such that there exists a semi-functor $P': \omega \rightarrow \mathcal{P}_C^S$ with $P = Tr \circ P'$. D satisfies the GTC if every path through D satisfies the trace condition of \mathcal{T} . We call an ACD satisfying the GTC an abstract cyclic proof.

Requiring P' to be a semi-functor in the definition above rules out constant ‘paths’ obtained via $P(i) := Tr(s), P(i < i + 1) := 1_{Tr(s)}$ which are not generated by traversing the ACD along any branch of the cyclic tree.

As ACDs serve as an abstract representation of pre-proofs, each pre-proof naturally induces an ACD.

Definition 4.3. Let $(SEQ, \mathcal{R}, \rho, PFS)$ be a cyclic proof system whose soundness condition is induced by a trace interpretation $\iota: \mathcal{R} \rightarrow \mathcal{T}$. Any pre-proof (C, λ, δ) induces an abstract cyclic derivation (C, Tr) with

$$Tr(s) := \iota(\lambda(s)) \quad Tr(s < si) := r_i : \iota(\lambda(s)) \rightarrow \iota(\lambda(si)) \text{ where } r = \delta(s)$$

Example 4.1. Consider the μ -pre-proof and its corresponding ACD depicted in Fig. 3. For the sake of readability, we have opted to write the carrier sets of the ACD as simple sets of variables x, y , etc., instead of sets of pairs $(\varphi, x), (\psi, y)$, etc., since each variable occurs in only one of the formulas in the sequent. Fully specified, the first set should be $\{(vx.x, x), (\mu y.vz.y, z)\}$. The annotated morphisms are $1 := \{(x, 0, x), (z, 0, z)\}$, $R_x := \{(x, 1, x), (z, 0, z)\}$, $R_y := \{(x, 0, x), (z, 2, z)\}$ and $R_z := \{(x, 0, x), (z, 1, z)\}$.

Note that $(z, 2, z) \in R_y$ as $y <_\varphi z$ with $\varphi := \mu y.vz.y$. The μ -pre-proof above has only one branch Γ which has one v -trace (that on x) and is thus a proof. Similarly, its corresponding path $\hat{\Gamma}: \omega \rightarrow \mathcal{T}_{\mathbb{F}}$ is a path through the ACD and satisfies the trace condition with $S(i) := 5i$ and $\sigma := i \mapsto x$. In principle, verifying that the ACD is a proof would require checking infinitely many other paths P . This is because paths through ACDs need not start at their root and a step $P(i < i + 1)$ may correspond to a path segment $p \in Path_C(s, t)$ with $|p| > 2$. However, as all such paths are subpaths of $\hat{\Gamma}$, they satisfy the trace condition by subpath invariance. Indeed, this observation extends to arbitrary ACDs obtained by transforming concrete cyclic derivations: any path through an ACD is always a subpath of $\hat{\Gamma}$ for some branch Γ through the corresponding concrete cyclic derivation. This means that the GTC in Definition 4.2, although it may have seemed stricter than needed, corresponds precisely to that of concrete cyclic proof systems.

We close this section by showing that, via composition of trace maps, ACDs can be transformed into equivalent ACDs whose number of nodes $|T|$ is linear in their number of cyclic edges $|\beta|$. For ACDs generated from concrete cyclic derivations, as defined in Definition 4.3, this will usually result in a drastic reduction of $|T|$. The procedure can be viewed as a sort of ‘compression algorithm’ which could prove useful for implementing programs, such as proof checking (cf. Theorem 5.3), that have $|T|$ as one of their complexity parameters. Indeed, it seems the automated theorem prover CYCLIST already relies on a similar optimisation (see Brotherston et al.

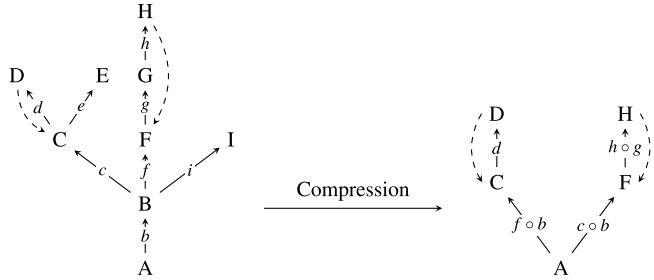


Figure 4. Compressing an ACD via the procedure outlined in the proof of Theorem 4.1.

2012, Section 4.2) though we are not aware of a formal characterisation of this optimisation in the literature. An example illustrating the procedure is given in Figure 4.

Theorem 4.1. Any $D = ((T, \beta), Tr)$ can be transformed into $D' = ((T', \beta'), Tr')$ such that D' is a proof if and only if D is a proof and $|T'| \leq 2|\beta| + 1$.

Proof. Fix $B := \text{im}(\beta) \cup \text{dom}(\beta) \cup \{\varepsilon \mid \nexists s \in \text{im}(\beta), \forall t \in \text{im}(\beta) \cup \text{dom}(\beta), s < t \text{ if } t \text{ is a proper prefix of } s\}$. We now construct a sequence of partial maps $(f_i : B \rightarrow \omega^*)_i$. Fix $f_0(s) := \varepsilon$ where s is the $<$ -least element of B . To construct f_{n+1} consider each $s \in \text{dom}(f_n)$, that is, every $s \in B$ which was added by the previous construction step. For each such s , let $\{t_0, \dots, t_n\} \subseteq B$ be all $t \in B$ such that s can reach t without crossing any other node from B (formally, if there is $spt \in \text{Path}_C(s, t)$ with $p \in (T \setminus B)^*$) and set $f_{n+1}(t_i) := f_n(s)i$. Fixing $f := \bigcup_{n \leq |B|} f_n$ and $T' := \text{im}(f)$, observe that $f : B \rightarrow T'$ is a $<$ -isomorphism between B and T' . Thus $C' := (T', \beta' := f \circ \beta \circ f^{-1})$ is a cyclic tree with $\beta'(x) = y$ iff $\beta(f^{-1}(x)) = f^{-1}(y)$. Also, clearly $|T'| = |B| \leq 2|\beta| + 1$.

To extend C' to an ACD D' , take $Tr'(s) := Tr(f^{-1}(s))$ for $s \in T'$. For any $s < t \in T'$, observe that per construction of T' , there exists a unique path $f^{-1}(s)p_{st}f^{-1}(t) \in \text{Path}_C(f^{-1}(s), f^{-1}(t))$ with the property that $p_{st} \in (T \setminus B)^*$. Pick $Tr'(st) := Tr(f^{-1}(s)p_{st}f^{-1}(t))$ and note that this is both well defined and fully specifies $Tr' : \mathcal{P}_{C'} \rightarrow \mathcal{T}$. The claim now follows from the following fact, which is easily verified: for any path P through D , there exists a path $P' \sim P$ through D' and vice versa. \square

5. A Ramsey-based Soundness Condition

This section presents a soundness condition on pre-proofs which is equivalent to the common global trace condition. It is similar to the condition put forward by Lee et al. (2001) for the purpose of program termination based on the size-change principle, a condition analogous to the trace conditions of cyclic proof systems such those put forward in Sprenger and Dam (2003) and Simpson (2017). The correctness proof of the soundness condition relies on Ramsey’s (1930) theorem, rather than the automata-theoretic methods prevalent in cyclic proof theory. We employ the notation $[A]^n := \{X \subseteq A \mid |X| = n\}$.

Theorem 5.1. Ramsey’s theorem. Let A be a countable set, C finite and $n \in \omega$. For any colouring $f : [A]^n \rightarrow C$, there exists a colour $c \in C$ and a countable $B \subseteq A$ such that $f(X) = c$ for any $X \in [B]^n$.

The central insight motivating the soundness condition is that the Ramsey theorem guarantees the existence of certain well-behaved subpaths of paths through HOM-finite categories. For every $R : X \rightarrow X$ in a category \mathcal{T} , the periodic R path is $R^\omega : \omega \rightarrow \mathcal{T}$ with by $R^\omega(i) := X$ and $R^\omega(i < i + 1) := R$. A morphism $R : X \rightarrow X$ is idempotent if $R = R \circ R$.

Lemma 5.1. Let $P : \omega \rightarrow \mathcal{T}$ be a path and $X \in \text{Ob}(\mathcal{T})$ such that $P(i) = X$ infinitely often. If $\text{HOM}_{\mathcal{T}}(X, X)$ is finite, then $R^\omega \subseteq P$ for some idempotent $R : X \rightarrow X$.

Proof. $P(i) = X$ holding infinitely often means there exists $Q \subseteq P$ such that $Q(i) = X$ for all $i \in \omega$. Then $Q(\{i, j\}) := Q(i < j)$ induces a colouring $Q : [\omega]^2 \rightarrow \text{HOM}_{\mathcal{T}}(X, X)$ on ω . By the Ramsey

theorem, there exists $R \in \text{HOM}_{\mathcal{T}}(X, X)$ and an infinite $M \subseteq \omega$ such that $Q(i < j) = R$ for $i < j \in M$. Then, $R^\omega = Q \circ S$ where $S(i) :=$ the i th least $m \in M$. The idempotence of R follows as $R = Q(S(0 < 2)) = Q(S(1 < 2)) \circ Q(S(0 < 1)) = R \circ R$. \square

Similarly to Definition 4.2, we limit our attention to the image of the trace functor $\widehat{\text{Tr}}: \mathcal{P}_C^S \rightarrow \mathcal{T}$ restricted to the category of progressing paths. Fixing some ACD (C, Tr) , we thus write $\overline{\text{HOM}}(s, t)$ for $\text{HOM}_{\text{im}(\widehat{\text{Tr}})}(\widehat{\text{Tr}}(s), \widehat{\text{Tr}}(t))$, that is, the set of all trace maps that can be generated by walking along some path $p \in \text{Path}_C(s, t)$ with $|p| > 1$.

Definition 5.1. *Let (C, Tr) be an ACD such that for every $u \in \text{dom}(\beta)$, $\overline{\text{HOM}}(u, u)$ is finite. Then it satisfies the Ramsey trace condition if for every $u \in \text{dom}(\beta)$ and every idempotent $R \in \overline{\text{HOM}}(u, u)$, the path $R^\omega: \omega \rightarrow \mathcal{T}$ satisfies the trace condition.*

Theorem 5.2. *Let (C, Tr) be an ACD such that for every $u \in \text{dom}(\beta)$, $\overline{\text{HOM}}(u, u)$ is finite. Then it satisfies the GTC if and only if it satisfies the Ramsey trace condition.*

Proof. First, suppose D satisfies the GTC. Then pick $u \in \text{dom}(\beta)$ and $R \in \overline{\text{HOM}}(u, u)$. As R is in the image of $\widehat{\text{Tr}}$, we know that there exists some $p \in \text{Path}_C(u, u)$ with $|p| > 1$ and $\text{Tr}(p) = R$. Then $R^\omega = \text{Tr} \circ p^\omega$, meaning R^ω is a path through D and thus satisfies the trace condition.

Conversely, if D satisfies the Ramsey trace condition and pick some path $\text{Tr} \circ P$ for $P: \omega \rightarrow \mathcal{P}_C^S$. Clearly, there exists $P \subseteq P'$ with $|P'(i < i + 1)| = 2$ for all $i \in \omega$, that is, a representation of P which does not ‘skip’ any nodes. As P' describes an infinite path through C , there needs to be some $u \in \text{dom}(\beta)$ with $P'(i) = u$ for infinitely many $i \in \omega$ and thus by Lemma 5.1 an idempotent $R: \text{Tr}(u) \rightarrow \text{Tr}(u)$ such that $R^\omega \subseteq \text{Tr} \circ P' \supseteq \text{Tr} \circ P$. By the Ramsey trace condition, R^ω satisfies the trace condition, meaning $\text{Tr} \circ P$ does so as well. \square

Remark 5.1. *By restricting our attention to $\text{HOM}_{\text{Tr}^s}(u, u)$, we guarantee that $1_{\text{Tr}(u)}$ can only occur in $\text{HOM}_{\text{Tr}^s}(u, u)$ if there is some $p \in \text{Path}_C(u, u)$ such that $\text{Tr}(p) = 1_{\text{Tr}(u)}$. This is important as in most sensible trace categories – including all trace categories defined in this article – the path $1_{\text{Tr}(u)}^\omega$ does not satisfy the trace condition. Naïvely including $1_{\text{Tr}(u)}$ in the collection of idempotent morphisms to consider when checking for GTC satisfaction would thus invalidate the condition given above.*

Note that this soundness condition can only be stated in a setting like ours, in which the composition of trace maps is considered. This is because the closed collection of compositions, in this case the HOM -set, gives rise to the finite colouring required to apply the Ramsey theorem. The Ramsey trace condition induces a novel algorithm for checking whether an ACD is a proof. This algorithm is analogous to that given for program termination by Lee et al. (2001). A more careful analysis of algorithms of this type in the realm of cyclic proofs is undertaken by Cohen et al. (2024).

Theorem 5.3. *Let \mathcal{T} be such that*

- (1) *from $R: X \rightarrow Y$ and $R': Y \rightarrow Z$ one can compute $R' \circ R: X \rightarrow Z$,*
- (2) *for $R, R': X \rightarrow Y$ one can decide whether $R = R'$, and*
- (3) *for idempotent $R: X \rightarrow X$ one can decide whether $R^\omega: \omega \rightarrow \mathcal{T}$ satisfies the trace condition.*

Further let $D = (C, \text{Tr}: \mathcal{P}_C \rightarrow \mathcal{T})$ be an ACD with Tr computable and such that for every $u, v \in T$ the set $\overline{\text{HOM}}(u, v)$ is finite. Then it is decidable whether D is a proof.

Proof. By Theorem 5.2, we know that it suffices to check that for each $u \in \text{dom}(\beta)$ and for all idempotent $R \in \overline{\text{HOM}}(u, u)$, R^ω satisfies the trace condition. Thus, simply compute all of the $\overline{\text{HOM}}$ -sets and then check each idempotent endomorphism via procedure (3). The $\overline{\text{HOM}}$ -sets are computed by an iterative procedure with base cases:

$$H^0(u, v) := \{\text{Tr}(uv) \mid v \text{ child of } u\} \cup \{1_{\text{Tr}(u)} \mid u \in \text{dom}(\beta) \text{ and } \beta(u) = v\}$$

and the iterative steps, using the procedure from assumption (1),

$$H^{i+1}(u, v) := H^i(u, v) \cup \{R \circ R' \mid w \in T, R' \in H^i(u, w), R \in H^i(w, v)\}.$$

We carry out the procedure until $H^i(u, v) = H^{i+1}(u, v)$ for all $u, v \in T$, that is, until a fixed point is reached, which can be detected using procedure (2). That such a fixed point will be reached is guaranteed by the finiteness of the $\overline{\text{HOM}}(u, v)$. It is easily observed that $H(u, v) = \overline{\text{HOM}}(u, v)$. Now check whether D satisfies the Ramsey trace condition by using the procedure (3) on all $R \in H(u, u)$ for $u \in \text{dom}(\beta)$ that satisfy $R = R \circ R$, which can be detected by using the procedures (1) and (2). □

Remark 5.2. *As the assumptions of Theorem 5.3 are phrased in terms of computability, nothing more can be said about the complexity of the procedure in general. In Section 7, we prove the GTC, and equivalently the RTC, to be PSPACE-complete for the category \mathcal{T}_A of any activation algebra A . The witnessing decision procedure in PSPACE (see Lemma 7.4) is based on infinite word automata. Lee et al. (2001) demonstrate how an approach similar to Theorem 5.3 can be carried out in PSPACE by constructing the HOM-sets ‘not all at once’.*

While the GTC and RTC are equivalent, it may seem that a procedure designed for verification of the RTC could be more efficient than one implemented in terms of the GTC, based on the fact that in the RTC case a property is only checked per simple cycle (rather than every possible path). However, every morphism of a simple cycle’s $\overline{\text{HOM}}$ -set must be checked, which can lead to an exponential blowup in cases such as $\mathcal{T}_{\mathbb{B}}$. As pointed out above, procedures in PSPACE ‘designed for’ both are known.

Corollary 5.1. *Let A be a activation algebra. It is decidable whether an ACD $(C, \text{Tr}: \mathcal{P}_C \rightarrow \mathcal{T}_A)$ is a proof.*

Proof. Simply observe that \mathcal{T}_A satisfies the criteria above. Notably, for an idempotent $R: X \rightarrow X$ in \mathcal{T}_A , one can decide whether R^ω satisfies the trace condition by checking if there exists an $x \in X$ such that $(x, \alpha, x) \in R$. □

Corollary 5.2. *It is decidable whether a μ -pre-proof constitutes a μ -proof.*

Proof. For a μ -pre-proof Π , compute its induced ACD $\hat{\Pi}$ over $\mathcal{T}_{\mathbb{F}}$ as in Definition 4.3 and then decide whether $\hat{\Pi}$ satisfies the GTC via Corollary 5.1. The decision extends to Π via Proposition 3.1. □

6. Relating Trace Categories and Automata Theory

This section connects our abstract framework for cyclic derivations to automata theory, a field instrumental to cyclic proof theory. The main theorems of this section are abstractions of properties common to many cyclic proof systems. They serve to illustrate that ACDs allow reasoning uniformly about a large class of cyclic proof systems by abstracting away the logic-specific details.

The main point of interaction between cyclic proofs and automata theory is based on the observation that the trace condition of cyclic proof systems tends to be ω -regular. That is, the branches of cyclic proofs which satisfy the trace condition can be recognised by infinite word automata. We thus begin by recalling Büchi-automata, one of the classes of automata characterising ω -regularity. A Büchi-automaton is a tuple $\mathfrak{B} = (\Sigma, Q, \Delta, s, F)$ consisting of a finite alphabet Σ , a finite set of states Q , a starting state $s \in Q$, a transition relation $\Delta \subseteq Q \times \Sigma \times Q$ and a set $F \subseteq Q$ called the acceptance condition.

Given a Büchi-automaton \mathfrak{B} and a word $\sigma \in \Sigma^\omega$, a sequence $\rho \in Q^\omega$ is called a run of \mathfrak{B} on σ if $\rho_0 = q_0$ and for each $i \in \omega$ we have $(\rho_i, \sigma_i, \rho_{i+1}) \in \Delta$. A run ρ is accepting if there is some $q \in F$

such that $\rho_i = q$ for infinitely many $i \in \omega$. A word σ is accepted by \mathfrak{B} if there exists an accepting run of \mathfrak{B} on σ . The set $L(\mathfrak{B}) := \{\sigma \in \Sigma^\omega \mid \sigma \text{ is accepted by } \mathfrak{B}\}$ is the language of \mathfrak{B} .

For most cyclic proof systems, given a cyclic pre-proof, there exists a infinite word automaton which recognises precisely the branches through that pre-proof that satisfy the trace condition (see Niwiński and Walukiewicz 1996 and Sprenger and Dam 2003 for examples of such constructions). This principle is extended to the setting of trace categories below.

Definition 6.1. Fix a trace category \mathcal{T} . Its trace condition is ω -regular if, for any finite set M of morphisms of \mathcal{T} and starting object $S \in \text{Ob}(\mathcal{T})$, there exists a Büchi-automaton \mathfrak{B} such that

$$L(\mathfrak{B}) = \{\pi \in M^\omega \mid P(i < i + 1) := \pi_i \text{ is a valid path with } P(0) = S, \text{ satisfying the trace condition}\}$$

This notion of ω -recognisability captures most uses of automata theory in the cyclic proof theory literature. For example, it allows us to carry out the most common proof of the decidability of the GTC in our abstract setting.

Theorem 6.1. Let the trace condition of \mathcal{T} be ω -regular in a computable manner, that is, the recognising automaton \mathfrak{B} for every set M of morphisms and $S \in \text{Ob}(\mathcal{T})$ can be computed. It is decidable whether an ACD $D = (C, \text{Tr} : \mathcal{P}_C \rightarrow \mathcal{T})$ satisfies the GTC.

Proof. For $C = (T, \beta)$, consider the set of trace maps along the edges of D :

$$M := \{\text{Tr}(uv) \mid u \in T, v \in \text{Chld}(u)\} \cup \{1_{\text{Tr}(u)} \mid u \in \text{dom}(\beta)\}$$

which is finite as T is. Compute the recognising automaton \mathfrak{B} for $S := \text{Tr}(\varepsilon)$ and construct a second Büchi-automaton $\mathfrak{A} = (T, M, \varepsilon, \Delta, T)$ with

$$\Delta := \{(u, \text{Tr}(uv), v) \mid u \in T, v \in \text{Chld}(u)\} \cup \{(u, 1_{\text{Tr}(u)}, \beta(u)) \mid u \in \text{dom}(\beta)\}$$

\mathfrak{A} accepts precisely the sequences of morphisms along the infinite branches of D . Thus, deciding whether D satisfies the GTC reduces to deciding $L(\mathfrak{B}) \subseteq L(\mathfrak{A})$. Such inclusions between Büchi-automata are decidable (McNaughton 1966). □

As a consequence of the Ramsey trace condition Theorem 5.2, every trace category with finite HOM-sets has an ω -regular trace condition. Note that under computability conditions analogous to those of Theorem 5.3, the ω -recognisability proven below is ‘computable’ as required for Theorem 6.1. Notably, the result below applies (in a computable manner) to every \mathcal{A} -activated trace category.

Theorem 6.2. Let \mathcal{T} be a trace category with finite $\text{HOM}(X, Y)$ for every $X, Y \in \text{Ob}(\mathcal{T})$. Then its trace condition is ω -regular.

Proof. As a corollary to Lemma 5.1, a path $P : \omega \rightarrow \mathcal{T}$ satisfies the trace condition iff there exists an idempotent $R : X \rightarrow X$ in \mathcal{T} such that $R^\omega \subseteq P$ and R^ω satisfies the trace condition. Fix some finite M and $S \in \text{Ob}(\mathcal{T})$. Define $O := \{\text{dom}(R) \mid R \in M\} \cup \{\text{cod}(R) \mid R \in M\}$. Define the set of good, idempotent morphisms on $X \in O$ as:

$$\text{Gi}(X) := \{R : X \rightarrow X \mid R \text{ idempotent and } R^\omega \text{ satisfies the trace condition}\}$$

and construct a Büchi-automaton $\mathfrak{B} = (M, Q, \Delta, F, S)$ with

$$Q := O \cup \Sigma X, Y \in O. \text{Gi}(X) \times \text{HOM}(X, Y) \tag{a}$$

$$\Delta := \{(X, R, Y) \mid X, Y \in O, R : X \rightarrow Y \in M\} \tag{a}$$

$$\cup \{(X, R', (X, Y, R, R')) \mid X, Y \in O, R \in \text{Gi}(X), R' : X \rightarrow Y\} \tag{b}$$

$$\cup \{((X, Y, R, R'), R'', (X, Z, R, R'' \circ R')) \mid X, Y, Z \in O, R'' : Y \rightarrow Z \in M, R \neq R'\} \tag{c}$$

$$\cup \{((X, X, R, R), R', (X, Y, R, R')) \mid X \in O, R \in \text{Gi}(X), R : X \rightarrow Y \in M\} \tag{c}$$

$$F := \{(X, X, R, R) \mid X, Y \in \text{Ob}(\mathcal{T}), R \in \text{Gi}(X), R' : X \rightarrow Y \in M\}$$

Q is finite as O and each of the $\text{HOM}(X, Y)$ are. First, note that \mathfrak{B} clearly rejects every sequence $\pi \in M^\omega$ which does not represent a well-formed path starting at S . It thus remains to show \mathfrak{B} accepts a path $P: \omega \rightarrow \mathcal{T}$ iff it is such that $R^\omega \subseteq P$ for some $R \in \text{GI}(X)$ and $X \in O$. First, suppose $R^\omega = P \circ S$ as desired. Then an accepting run on the sequence $\pi_i := P(i < i + 1)$ is obtained by:

- Taking (a)-transitions, reading $\pi[0, S(1) - 1]$, then taking a (b)-transition on $P(S(1) - 1, S(1))$, the ‘first part of’ $P(S(1) < S(2))$: $(X, Y, R, P(S(1) < S(1) + 1))$ where $R: X \rightarrow X$. For this, note that $S(1) - 1 \geq 0$ because $S(0) < S(1)$. Continue reading $\pi[S(1), S(2) - 1]$, reaching (X, X, R, R) as $P(S(1) < S(2)) = R$.
- From then on, taking (c)-transitions, reading $\pi[S(i), S(i + 1) - 1]$, always arriving at (X, X, R, R) as $P(S(i) < i + 1) = R$.

This run is accepting as $(X, X, R, R) \in F$ is passed infinitely often. For the converse direction, simply observe that every accepting run on \mathfrak{B} needs to be structured as above, that is, eventually ‘picking’ an $X \in O$ and $R \in \text{GI}(X)$ via a (b)-transition and then ‘assembling’ R along π infinitely often via (c)-transitions, thereby demonstrating $R^\omega \subseteq P$. □

Remark 6.1. *The converse of Theorem 6.2 need not hold. Consider a trace category \mathcal{T} with ω -regular trace condition. Then $\mathcal{T} \times \text{SET}$, where SET is the usual category of sets, can be equipped with an ω -regular trace condition, namely that of its first component. However, the HOM -sets of $\mathcal{T} \times \text{SET}$ are not all finite.*

Remark 6.2. *It is also possible to construct recognising Büchi-automata for \mathcal{T}_A more directly in terms of A . Roughly, the states of such automata are triples $\Sigma X \in O. X \times A$ which take transitions $((X, x, a), R: X \rightarrow Y, (Y, y, a \vee b))$ for $(x, b, y) \in R$. Whenever the third component reaches α , it is reset to 0, crossing a state in the acceptance condition F . The resulting automata resemble more closely the automata constructions usually found in the cyclic proof theory literature. The full details of this automata construction can be found in Wehr (2021, Proposition 5.11).*

The second result connecting the theories of cyclic proofs and automata we cover in this section relates cyclic proofs and ∞ -proofs. ∞ -proofs allow ill-founded, finitely branching derivation trees and thus require a soundness condition, similar to cyclic proofs. From this point of view, cyclic proofs are simply regular ∞ -proofs, that is, those which are representable as finite graphs. The result states that on finite derivation systems, the cyclic and ∞ -proof systems induced by a trace interpretation prove the same sequents. This property does generally not hold for infinite derivation systems: for example, in cyclic arithmetic (Simpson 2017), the ∞ -proofs prove all true sentences of first-order arithmetic, whereas the cyclic proofs only prove the same sentences as Peano arithmetic. We begin by formally defining ∞ -proof systems and the infinite tree Büchi-automata, which play a key role in the result’s proof.

Given a derivation system $(\text{SEQ}, \mathcal{R}, \rho)$, an ∞ -derivation is a triple (T, λ, δ) consisting of a (possibly infinite) tree T and functions $\lambda: T \rightarrow \text{SEQ}$ and $\delta: T \rightarrow \mathcal{R}$ such that for every $t \in T$ with $\text{Chld}(t) = \{t1, \dots, tn\}$ the functions λ and δ agree: $\rho(\delta(t)) = (\lambda(t), \lambda(t1), \dots, \lambda(tn))$. In other words, an ∞ -derivation is a derivation which might have infinite branches. Denote the set of ∞ -proofs in \mathcal{R} by $\text{ID}(\mathcal{R})$. An ∞ -proof system is a tuple $(\text{SEQ}, \mathcal{R}, \rho, \text{PFS})$ consisting of a derivation system $(\text{SEQ}, \mathcal{R}, \rho)$ and a set of ∞ -derivations $\text{PFS} \subseteq \text{ID}(\mathcal{R})$ called ∞ -proofs. An ∞ -proof $\Pi = (T, \lambda, \delta)$ with $\lambda(\varepsilon) = \Gamma$ is called a proof of Γ . Analogously to the case for cyclic proof systems, a derivation system $(\text{SEQ}, \mathcal{R}, \rho)$ and a trace interpretation $\iota: \mathcal{R} \rightarrow \mathcal{T}$ induce an ∞ -proof system in which $\Pi \in \text{PFS}$ iff every path along its branches satisfies the trace condition of \mathcal{T} .

For a finite alphabet Σ , a Σ -labelled tree is a pair $(T, \lambda: T \rightarrow \Sigma)$ for a tree T . A Σ -labelled tree (T, λ) is a subtree of Σ -labelled (T', λ') if it is a ‘suffix’ of T' , that is, there exists some $t \in T'$ such that $T = \{ts \in T' \mid s \in T'\}$ and $\lambda(s) = \lambda'(ts)$. A Büchi tree automaton is a tuple $\mathfrak{A} = (\Sigma, Q, \Delta, s, F)$ consisting of a finite alphabet Σ , a set of states Q , a set of transitions $\Delta \subseteq Q \times \Sigma \times Q^*$, a starting

state $s \in Q$ and an acceptance condition $F \subseteq Q$. Let (T, λ) be a Σ -labelled tree. A run of \mathfrak{A} on (T, λ) is a Q -labelling $\rho: T \rightarrow Q$ of T such that $\rho(\varepsilon) = s$ and for each $t \in T$ with $\text{Chld}(t) = \{t_1, \dots, t_n\}$ the transition $(\rho(t), \lambda(t), \rho(t_1), \dots, \rho(t_n)) \in \Delta$. A run is *accepting* if for every infinite branch $b \in T^\omega$ of T , there exists a $q \in F$ such that $\rho(b_i) = q$ infinitely often. A Σ -labelled tree (T, λ) is *accepted* by \mathfrak{A} if there is an accepting run of \mathfrak{A} on it. The set $L(\mathfrak{A}) := \{(T, \lambda) \mid (T, \lambda) \text{ is accepted by } \mathfrak{A}\}$ is the language of \mathfrak{A} .

Theorem 6.3. *Let $(\text{SEQ}, \mathcal{R}, \rho)$ be a derivation system with \mathcal{R} finite. Fix a trace interpretation $\iota: \mathcal{R} \rightarrow \mathcal{T}$ such that the trace condition of \mathcal{T} is ω -regular. Then any sequent $\Gamma \in \text{SEQ}$ is proven by a cyclic proof iff it is proven by an ∞ -proof.*

Proof. For the left-to-right direction, simply observe that unfolding the cyclic proof yields an ∞ -derivation satisfying the trace condition induced by ι .

Conversely, note that any ∞ -derivation (T, λ, δ) in \mathcal{R} thus constitutes a \mathcal{R} -labelled tree (T, δ) . We begin by constructing a Büchi tree automaton which accepts precisely the ∞ -proofs of Γ , represented as \mathcal{R} -labelled trees. By ω -regularity of \mathcal{T} , there exists a Büchi-automaton $\mathfrak{A} = (Q, M, \Delta, s, F)$ for $M := \{r_i : \iota(\Gamma) \rightarrow \iota(\Delta_i) \mid r \in \mathcal{R}, \rho(r) = (\Gamma, \Delta_1, \dots, \Delta_n), i \leq n\}$ and $S := \iota(\Gamma)$. From it, the desired Büchi tree automaton $\mathfrak{B} := (\text{SEQ} \times Q, \mathcal{R}, \Delta_{\mathfrak{B}}, (\Gamma, s), \text{SEQ} \times F)$ is constructed, taking

$$\Delta_{\mathfrak{B}} := \bigcup_{r \in \mathcal{R}} \left\{ \left((\Gamma, q), r, \left((\Sigma_1, q_1), \dots, (\Sigma_n, q_n) \right) \right) \mid \begin{array}{l} q \in Q \text{ and } \rho(r) = (\Gamma, \Sigma_1, \dots, \Sigma_n) \text{ and} \\ (q, r_i : \iota(\Gamma) \rightarrow \iota(\Sigma_i), q_i) \in \Delta \text{ for each } i \end{array} \right\}.$$

That is, the automaton takes transition steps corresponding to the derivation rule $r \in \mathcal{R}$ labelling the tree, ‘walking the state from Q along according to \mathfrak{A} for the trace maps r_i chosen by the trace interpretation ι .

For the correctness of the automaton, observe that by the choice of $\Delta_{\mathfrak{B}}$, \mathfrak{B} has a run on a \mathcal{R} -labelled tree if it constitutes an ∞ -derivation with endsequent Γ , that is, its rules were applied with matching premises and conclusions. Furthermore, such a run is accepting if and only if the run of the \mathcal{T} -path induced by each infinite branch of the ∞ -derivation is accepted by \mathfrak{A} . Thus, \mathfrak{B} accepts precisely the ∞ -proofs of Γ .

Because there is a ∞ -proof of Γ , the language $L(\mathfrak{B})$ of \mathfrak{B} is not empty. It is a classic result of infinite tree automata theory (see e.g., Corollary 8.20 in Nieer 2002) that in such a case, $L(\mathfrak{B})$ contains a regular tree Π . Such regular trees can be represented as finite graphs, allowing the proof Π to be represented as a cyclic proof Π' . As any element of $L(\mathfrak{B})$ is a proof of Γ , so is Π' . \square

An application of the result above is establishing the equivalence between cyclic proof systems with and without a Cut-rule. Many Cut-elimination procedures for cyclic proof systems in the literature are corecursive algorithms which lazily transform cyclic proofs with Cut-applications into ∞ -proofs without Cuts (e.g., Baelde et al. 2016; Fortier and Santocanale 2013; Savateev and Shamkanov 2021). If the Cut-free fragment of the derivation system is finite, the result above can then be applied to conclude that there must also exist a Cut-free cyclic proof. An example of the result being applied in this way is given by Savateev and Shamkanov (2021). A disadvantage of this method of Cut-elimination is that the Cut-free cyclic proof need not be related to the original Cut-free ∞ -proof. This means this Cut-elimination result does not preserve computational content.

The automata construction employed in the proof of Theorem 6.3 also yields a decision procedure for provability in the cyclic and ∞ -proof systems. The result again requires some light computability assumptions.

Corollary 6.1. *Let $(\text{SEQ}, \mathcal{R}, \rho)$ be a derivation system with \mathcal{R} finite. Fix a trace interpretation $\iota: \mathcal{R} \rightarrow \mathcal{T}$ such that the trace condition of \mathcal{T} is ω -regular in a computable manner and for any $r \in \mathcal{R}$ with $\rho(r) = (\Gamma, \Delta_1, \dots, \Delta_n)$ the $r_i : \iota(\Gamma) \rightarrow \iota(\Delta_i)$ can be computed. Then it is decidable whether $\Gamma \in \text{SEQ}$ is provable in the induced cyclic and ∞ -proof systems.*

Proof. The proof of Theorem 6.3 constructs a Büchi tree automaton \mathfrak{B} such that $L(\mathfrak{B})$ is precisely the ∞ -proofs of Γ . By the computability assumptions, the automaton \mathfrak{B} can be computed. As the emptiness problem for parity tree automata is decidable (Rabin 1969), one can thus decide if there exists an ∞ -proof of Γ by deciding whether $L(\mathfrak{B})$ is empty. \square

Corollary 6.2. *The μ -sequents provable by the cyclic proof system and ∞ -proof system induced by the trace interpretation in Definition 3.4 coincide. Furthermore, the provability of a μ -sequent is decidable.*

Proof. As noted above, $\mathcal{T}_{\mathbb{F}}$ is ω -regular in a computable manner. It is also clear that the trace interpretation $\iota : \mu \rightarrow \mathcal{T}_{\mathbb{F}}$ is computable as required by Corollary 6.1. The derivation system given for the modal μ -calculus in Fig. 1 is not finite. However, because it is Cut-free, one can restrict the sequents occurring in a ∞ -derivation of a μ -sequent Γ to the so-called *Fischer-Ladner closure* of Γ . This closure is well known to be finite, as for example argued by Kozen (1983). Thus, it suffices to consider a finite fragment of the derivation system given in Fig. 1 for each sequent Γ , meaning Theorem 6.3 and Corollary 6.1 apply. \square

7. PSPACE-Completeness of Cyclic Proof Checking

In addition to putting forward the size-change criterion for program termination, Lee et al. (2001) also prove that the associated decision is PSPACE-complete. This result has been transferred to the setting of cyclic proofs by Nollet et al. (2019) who prove that checking the GTC of linear logic with least and greatest fixed points (μ MALL) is PSPACE-complete. We extend this result to ACDs over $\mathcal{T}_{\mathcal{A}}$. The proof of PSPACE-hardness proceeds analogously to Nollet et al. (2019) by a reduction to BOOLE program termination. Indeed, the ACDs given in Definition 7.2 are obtained by representing the cyclic derivations given by Nollet et al. as ACDs, applying the compression procedure (Theorem 4.1) and removing some spurious elements from their trace sets.

Definition 7.1. *A BOOLE program is a numbered sequence of instructions $1 : I_1; 2 : I_2; \dots; m : I_m$ composed according to the following grammar*

$$I ::= X := \neg X \mid \text{IF } X \text{ THEN } \ell \text{ ELSE } \ell'$$

where the labels $\ell, \ell' \in \{0, \dots, m\}$ and the variables X stem from some stock of variable letters.

Fix a BOOLE program p of length m making use of the variables $\Xi := \{X_1, \dots, X_n\}$. Given assignments $\sigma, \sigma' : \Xi \rightarrow \mathbb{B}$ and labels $\ell, \ell' \in \{0, \dots, m\}$, we write $(\ell, \sigma) \rightsquigarrow (\ell', \sigma')$ if $\ell \neq 0$ and either:

- (1) the instruction labelled by ℓ in p is $X := \neg X$ for some $X \in \Xi$, $\ell' \equiv \ell + 1 \pmod{m + 1}$ and $\sigma' = \sigma[X \mapsto \neg\sigma(X)]$, or
- (2) the instruction labelled by ℓ in p is $\text{IF } X \text{ THEN } \ell_1 \text{ ELSE } \ell_0$ for some $X \in \Xi$, $\ell' = \ell_{\sigma(X)}$ and $\sigma' = \sigma$.

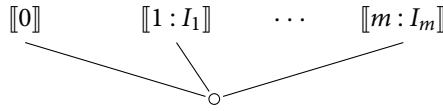
We write $(\ell, \sigma) \rightsquigarrow^* (\ell', \sigma')$ if there are ℓ_1, \dots, ℓ_k and $\sigma_1, \dots, \sigma_k$ such that $(\ell, \sigma) \rightsquigarrow (\ell_1, \sigma_1)$ and $(\ell_i, \sigma_i) \rightsquigarrow (\ell_{i+1}, \sigma_{i+1})$ and $(\ell_k, \sigma_k) \rightsquigarrow (\ell', \sigma')$. To make the labels explicit, we sometimes write $(\ell, \sigma) \rightsquigarrow_{\ell_1 \dots \ell_k}^* (\ell', \sigma')$.

Writing ϕ for the constant assignment $x \mapsto 0$, define

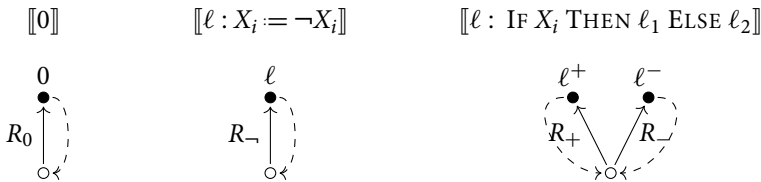
$$\text{BOOLE}_{\text{false}} := \{p \text{ a BOOLE program} \mid (1, \phi) \rightsquigarrow^* (0, \phi) \text{ under } p\}.$$

For the remainder of this section, we fix some program $p = 1 : I_1, \dots, m : I_m$ making use of variables X_1, \dots, X_n .

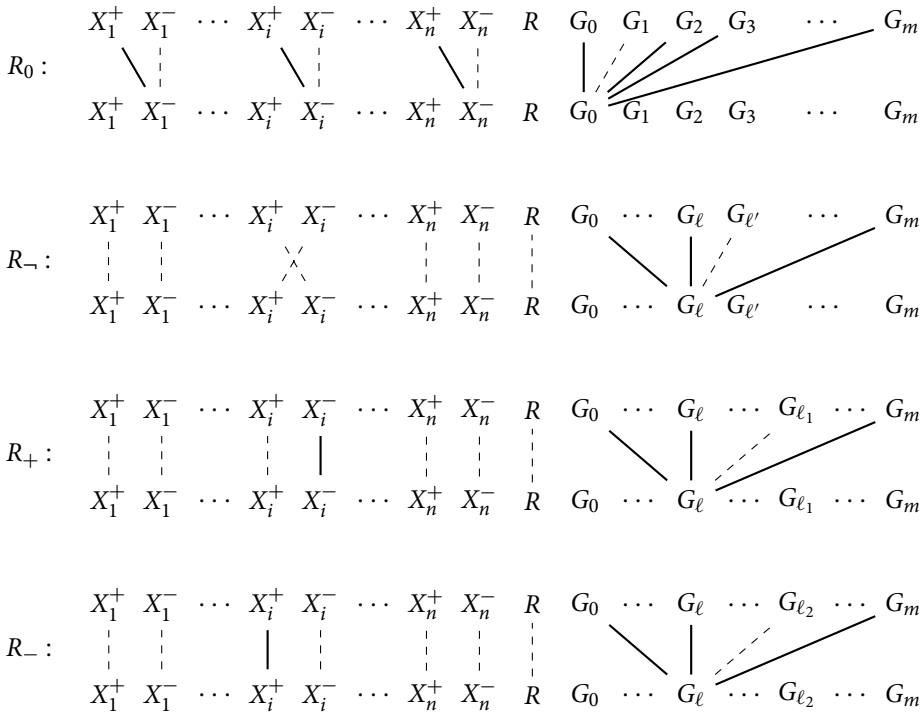
Definition 7.2. The ACD associated with p is $\llbracket p \rrbracket := (C, Tr : \mathcal{P}_C \rightarrow \mathcal{T}_A)$ for an arbitrary activation algebra A can be sketched as follows.



Every node s of C shares the same trace values $Tr(s) = \{X_1^+, X_1^-, \dots, X_n^+, X_n^-, R, G_0, \dots, G_m\}$. The ‘subtrees’ $\llbracket 0 \rrbracket$ and $\llbracket \ell : I_\ell \rrbracket$ each comprise of one or two buds whose companion is the root of $\llbracket p \rrbracket$ (drawn in white below). We refer to these buds by the names written above them. The \mathcal{T}_A -morphisms ‘decorating’ the subtrees’ edges are written next to them. For notational convenience, we often treat ℓ^+ and ℓ^- as the label ℓ .



The morphisms are specified below. A dashed line between A and B indicates the pair $(A, 0, B)$, a bold line the pair (A, α, B) and the absence of a connecting line between A and B the absence of all pairs (A, c, B) . In R_{-} , $\ell' = \ell + 1 \pmod{m + 1}$.



Write B for the set of names for buds of $\llbracket p \rrbracket$, that is,

$$B := \{0\} \cup \{\ell \mid \ell : X := \neg X \text{ in } p\} \cup \bigcup \{ \{\ell^+, \ell^-\} \mid \ell : \text{IF } X \text{ THEN GOTO } \ell_1 \text{ ELSE GOTO } \ell_0 \text{ in } p \}.$$

Any path through $\llbracket p \rrbracket$ corresponds to an infinite sequence $\ell \in B^\omega$ describing the sequence in which the buds of $\llbracket p \rrbracket$ are passed to generate said path. For this reason, we treat such sequences and paths $P : \mathcal{P}_C \rightarrow \mathcal{T}_A$ interchangeably. The property central to the reduction is that if $p \in \text{BOOLE}_{\text{false}}$, then $\llbracket p \rrbracket$ does not satisfy the trace condition. Thus, sequences $\ell \in B^\omega$ representing *unsuccessful runs* will satisfy the trace condition. The various aspects of the trace maps given in Definition 7.2 can thus be reframed as ensuring certain conditions on such $\ell \in V^\omega$.

We begin by showing that all sequences $\ell \in B^\omega$ which eventually diverge or do not even adhere to the control structure of p have progressing traces through R or the G_i s, respectively. For this, we define the *control-graph* of p as $G = (B, E)$ with

$$E := \{(0, 1)\} \cup \{(\ell, \ell + 1 \pmod{m+1}) \mid \ell : X := \neg X \text{ in } p\} \cup \bigcup \{ \{(\ell^+, \ell_1), (\ell^-, \ell_0)\} \mid \ell : \text{IF } X \text{ THEN GOTO } \ell_1 \text{ ELSE GOTO } \ell_0 \text{ in } p \}$$

where, if $\ell_2 : \text{IF } X \text{ THEN GOTO } \ell \text{ ELSE GOTO } \ell'$ in p , then pairs (ℓ_1, ℓ_2) are included as (ℓ_1, ℓ_2^+) and (ℓ_1, ℓ_2^-) . For a sequence ℓ of labels from B (finite or infinite), ℓ is a path through G , writing $\ell \in G$, if for each $i < |\ell|$, $(\ell_i, \ell_{i+1}) \in E$.

Lemma 7.1. Let $\ell \in B^\omega$ be a path through $\llbracket p \rrbracket$. Then,

- (1) ℓ has a progressing trace along R iff eventually 0 never occurs along ℓ .
- (2) ℓ has a progressing trace along the G_0, \dots, G_m iff no suffix of ℓ is a path through G .

Proof.

- (1) Observe that R -traces are interrupted by R_0 whenever a path passes through $0 \in B$ and are activated when passing through any other bud.
- (2) An activation on a G -trace takes place whenever it ‘jumps incorrectly’. That is, if $(\ell_i, \ell_{i+1}) \notin E$, then there is an activating trace from G_{ℓ_i} to $G_{\ell_{i+1}}$ in the trace maps of the subtree $\llbracket \ell_i \rrbracket$. Thus, a progressing trace through the G s indicates that such violations take place infinitely often, meaning no suffix of ℓ can be a path through G . Conversely, no suffix of ℓ being a path through G indicates infinitely many violations taking place. □

The only paths through $\llbracket p \rrbracket$ which remain unclassified are of the shape $u0u_10u_20u_30\dots$ where each u_i0 describes a *potential* run of p which at least is a path through the control-graph of p . We continue by analysing the traces on the X_i^\bullet along such potential runs. Consider a sequence $u = \ell_0 \dots \ell_n \in B^{n+1}$ and denote by R^i the morphism in $\llbracket p \rrbracket$ from the root to the bud ℓ_i . In the following, we denote the *trace map of the sequence* u by $R_u := R^{n-1} \circ \dots \circ R^0 \circ R_0$.

Lemma 7.2. Let $u \in (B \setminus \{0\})^+$ with $|u| = n + 1$ be such that $0u$ is a path through G . Then

- (1) There is no $X_j^\bullet \in \text{Tr}(\varepsilon)$ and no activation algebra element c such that $(X_i^+, c, X_j^\bullet) \in R_u$.
- (2) There are no $X_i^\bullet, X_j^\bullet \in \text{Tr}(\varepsilon)$ with $i \neq j$ and $(X_i^\bullet, c, X_j^\bullet) \in R_u$ for any activation algebra element c .
- (3) If $(\ell_0, \phi) \rightsquigarrow_u^* (\ell_n, \sigma)$ for some σ then for each X_i , there is exactly one a_i^+ and exactly one a_i^- such that $(X_i^-, a_i^+, X_i^+) \in R_u$ and $(X_i^-, a_i^-, X_i^-) \in R_u$. Furthermore,

- if $\sigma(X_i) = 0$ then $a_i^+ = \alpha$ and $a_i^- = 0$
 - if $\sigma(X_i) = 1$ then $a_i^- = \alpha$ and $a_i^+ = 0$
- (4) If there is no σ such that $(\ell_0, \phi) \rightsquigarrow_u^* (\ell_n, \sigma)$, then there exists some X_i such that $(X_i^-, 1, X_i^+), (X_i^-, 1, X_i^-) \in R_u$

Proof.

- (1) Observe that R_0 , the map from the root to 0, interrupts all traces starting at the X_i^+ .
- (2) Observe that all trace maps in $\llbracket p \rrbracket$ only ever connect trace values corresponding to the same variable X_i .
- (3) At the beginning of a BOOLE-run, all variables are 0. This is mirrored by R_0 in the sense of (2), as it connects each X_i^- to the X_i^\bullet according to the first clause. We continue by reasoning inductively along the run $(\ell_0, \phi) \rightsquigarrow_u^* (\ell_n, \sigma)$, performing a case distinction on the final transition $(\ell_{n-1}, \sigma') \rightsquigarrow (\ell_n, \sigma)$. The negation step inverts the value $\sigma'(X_i)$ of some variable X_i . This is mirrored in R_- by ‘swapping’ the traces on X_i^+ and X_i^- which preserves the property. For the IF -instruction, observe that $(\ell_{n-1}, \sigma') \rightsquigarrow (\ell_n, \sigma)$ means branch dictated by $\sigma(X_i)$ is taken (i.e., ℓ_{n-1} is some ℓ^+ and $\sigma(X_i) = 1$). Then the trace of the X_i^\bullet dual to $\sigma(X_i)$ is activated (i.e., X_i^- if $\sigma(X_i) = 1$), preserving the property.
- (4) We again reason inductively on u . If $|u| = 1$, then $(\ell_0, \phi) \rightsquigarrow_u^* (\ell_0, \phi)$, meaning the claim does not apply. Thus, $u = u'\ell_n$ for some $u' \in (B \setminus \{0\})^+$ and we perform a case distinction on whether there exists a σ such that $(\ell_0, \phi) \rightsquigarrow_{u'}^* (\ell_{n-1}, \sigma)$. If there exists no such σ , this property holds for $R_{u'}$ per inductive hypothesis and it is easily observed that any possible choice of R^{n-1} preserves the property for $R_{u'\ell_n}$. If $(\ell_0, \phi) \rightsquigarrow_{u'}^* (\ell_{n-1}, \sigma)$ then, as $(\ell_{n-1}, \ell_n) \in E$, this must mean that $\ell_{n-1} : \text{IF } X_i \text{ THEN } \ell \text{ ELSE } \ell'$ and ℓ_{n-1} is the ‘incorrect bud’ (i.e., ℓ_{n-1} is some ℓ^+ and $\sigma(X_i) = 0$). In such cases, the X_i^\bullet corresponding to the value of $\sigma(X_i)$ is activated by R^{n-1} (i.e., X_i^- if $\sigma(X_i) = 0$). Combining this with what is known about $R_{u'}$ by property (3), this means that $(X_i^-, \alpha, X_i^-), (X_i^-, \alpha, X_i^+) \in R_u$. □

With this classification of the potential runs, we can connect the trace condition on branches not covered by Lemma 7.1 to the runs of the program p .

Lemma 7.3. Let $\ell \in B^\omega$ be a path through $\llbracket p \rrbracket$ which does not have a progressing trace along R or the G_0, \dots, G_m . Then, $\ell = u0u_10u_20u_30 \dots$ with each $u_i \in (B \setminus \{0\})^+$. Furthermore, ℓ does not satisfy the trace condition iff from some $N \in \omega$ onwards, $(1, \phi) \rightsquigarrow_{u_i}^* (0, \phi)$ for all $i > N$.

Proof. That $\ell = u0u_10u_20 \dots$ follows directly from Lemma 7.1.

First suppose that $(1, \phi) \rightsquigarrow_{u_i}^* (0, \phi)$ for $i > N$. As BOOLE-programs are deterministic, this means $u_i = u$ for all $i > N$ and some fixed u . From Lemma 7.2 (3), it follows that for every X_j we have $(X_j^-, \alpha, X_j^+), (X_j^-, 0, X_j^-) \in R_u$. By Lemma 7.2 (1), the trace on X_j^+ is interrupted in R_0 . Thus, there can be no progressing trace along any X_j^+, X_j^- along the path $0u0u \dots$. Then, because of the previous results, there is no progressing trace along any of the elements of $\text{Tr}(\varepsilon)$ meaning ℓ does not satisfy the trace condition.

Conversely, suppose that infinitely many of the u_i did not satisfy $(1, \phi) \rightsquigarrow_{u_i}^* (0, \phi)$. There are two possibilities for such u_i s: either there is some $\sigma \neq \phi$ with $(1, \phi) \rightsquigarrow_{u_i}^* (0, \sigma)$ or not. In the former case, there thus must be some X_j with $\sigma(X_j) = 1$, meaning $(X_j^-, 1, X_j^-) \in R_{u_i}$. In the latter case, there also must be some X_j with $(X_j^-, 1, X_j^-)$. As the traces on X_j^- are never interrupted, if infinitely many such ‘deviant’ runs exist along ℓ , some X_j^- must be activated infinitely often, as there are only finitely many variables. Then, ℓ satisfies the trace condition. □

Corollary 7.1. For any given BOOLE program p , $p \in \text{BOOLE}_{\text{false}}$ iff $\llbracket p \rrbracket$ does not satisfy the trace condition of \mathcal{T}_A .

Theorem 7.1. Verifying the trace condition on ACDs in \mathcal{T}_A for any A is PSPACE-hard.

Proof. A BOOLE-program can be transformed into the ACD given in Definition 7.2 in LOGSPACE. By Corollary 7.1, this constitutes a LOGSPACE-reduction of a PSPACE-hard problem ($\text{BOOLE}_{\text{false}}$) to cyclic proof checking in \mathcal{T}_A for any A . □

Remark 7.1. The PSPACE-hardness result in Theorem 7.1 is only given in terms of a concrete family of trace categories, the activation algebra induced \mathcal{T}_A . It remains an open question if a more general condition on trace categories can be found which would yield PSPACE-hardness.

HOM-set finiteness of a trace category \mathcal{T} does not entail PSPACE-hardness. For instance, consider a HOM-set finite trace category \mathcal{T} in which every path satisfies the trace condition and which thus makes checking it trivially $O(1)$. This example also illustrates that ω -regularity of the trace condition of \mathcal{T} does not entail PSPACE-hardness.

Remark 7.2. Theorem 7.1 does not directly imply that the proof checking of any concrete cyclic proof system is PSPACE-hard, even if its trace condition can be expressed in terms of A . Instead, Theorem 7.1 only proves that there exist ‘suitably small’ ACDs over A whose GTC satisfaction codes the termination of BOOLE-programs. To extend this result to a concrete cyclic proof system, one must prove that for any BOOLE-program p , there exists a ‘suitably small’ cyclic proof whose (compressed) ACD is $\llbracket p \rrbracket$. Usually, this can be accomplished by finding formulas which can produce various traces. For example, for the variables, the following ‘trace gadgets’ are required:



To conclude PSPACE-completeness, it remains to show that proof checking for \mathcal{T}_A -ACDs is in PSPACE.

Lemma 7.4. For a given ACD $D = (C, \text{Tr} : \mathcal{P}_C \rightarrow \mathcal{T}_A)$, the verification whether it constitutes an abstract cyclic proof can be carried out in PSPACE.

Proof. Following the proof of Theorem 6.1, one can verify whether D is a proof by deciding whether the language of a Büchi-automaton \mathfrak{A} recognising all paths through D is included in the language of a Büchi-automaton \mathfrak{B} recognising all paths consisting of morphisms from Tr which satisfy the trace condition of \mathcal{T}_A . The language inclusion problem for regular languages is known to be in PSPACE (Kupferman and Vardi 1996). Thus, it remains to find automata \mathfrak{A} and \mathfrak{B} adhering to the aforementioned specifications whose sizes are polynomial in the size of D . The automaton \mathfrak{A} given in Theorem 6.1 is already suitable for this. On the other hand, the automaton \mathfrak{B} constructed according to Theorem 6.2 is too large, as its set of states contains the sets $\text{HOM}(X, X)$ for each set X of trace values in the image of Tr , whose size is exponential in the size of X . However, the \mathcal{T}_A -specific construction for the \mathfrak{B} given in Wehr (2021, Proposition 5.11) is of size polynomial to the size of D . Thus, there are suitable automata such that deciding $L(\mathfrak{A}) \subseteq L(\mathfrak{B})$ decides whether D is a proof, meaning this can be decided in PSPACE. □

Corollary 7.2. The problem of verifying whether an ACD over \mathcal{T}_A is a proof is PSPACE-complete.

The fact that proof checking for \mathcal{T}_A -ACDs is in PSPACE extends to a concrete cyclic proof system \mathcal{R} much more readily than PSPACE-hardness. The only restriction is that the set of trace

values of each \mathcal{R} -sequent is of polynomial size and that the trace data of an \mathcal{R} -pre-proof can be computed in PSPACE. All cyclic proof systems we know of satisfy these properties.

Lemma 7.5. Let $(\text{SEQ}, \mathcal{R}, \rho, \text{PFS})$ be a cyclic proof system whose trace condition is given by a trace interpretation $\iota : \mathcal{R} \rightarrow \mathcal{T}_{\mathcal{A}}$. Suppose that ι is such that for sequents $\Gamma \in \text{SEQ}$ the set $\iota(\Gamma)$ of trace values in Γ is of size polynomial to the size of Γ and can be computed from Γ in PSPACE. Suppose further that for each rule $r \in \mathcal{R}$ with $\rho(r) = (\Gamma, \Delta_1, \dots, \Delta_n)$, the maps $r_i : \iota(\Gamma) \rightarrow \iota(\Delta_i)$ can be computed in PSPACE. Then deciding whether an \mathcal{R} -pre-proof is a proof is in PSPACE.

Proof. Given an \mathcal{R} -pre-proof $\Pi = (C, \lambda, \delta)$, simply compute its induced ACD $D = (C, \text{Tr} : \mathcal{P}_C \rightarrow \mathcal{T}_{\mathcal{A}})$ as described in Definition 4.3. By the assumptions about ι , D can be computed in PSPACE. Thus, if D is of size polynomial to the size of Π , checking whether Π is an \mathcal{R} -proof can be carried out in PSPACE as described in Lemma 7.4. As C is simply ‘copied’, it suffices to show that Tr is of polynomial size. Per assumption, each $\text{Tr}(s)$ is $\iota(\lambda(s))$ and thus of polynomial size of the sequent $\lambda(s)$ which is accounted for in the size of Π . Similarly, $\text{Tr}(s_i) \subseteq \text{Tr}(s) \times \mathcal{A} \times \text{Tr}(s_i)$ for each $s \in C$ and $s_i \in \text{Chld}(s)$ and is thus of polynomial size to $\max\{|\lambda(s)|, |\lambda(s_i)|\}$. As this data suffices to code Tr , D overall is of polynomial size to Π as desired. \square

Corollary 7.3. Verifying whether a μ -pre-proof is a proof is in PSPACE.

8. Conclusion

The derivations of cyclic proof systems, which are finite, directed graphs, rather than finite trees, may not be sound, that is, they might conclude invalid sequents. Thus, cyclic proof systems distinguish between *pre-proofs* (well-formed derivations) and *proofs* (sound, well-formed derivations). The most common method is to impose a *global trace condition*: for a pre-proof to be a proof, all of its infinite branches must satisfy a so-called *trace condition*. We capture this trace condition in categorical terms: infinite branches are represented by *paths*, functors $P : \omega \rightarrow \mathcal{T}$ from the pre-order category on ω to a *trace category* \mathcal{T} . A trace category \mathcal{T} is a category equipped with a *trace condition*, a predicate on paths through \mathcal{T} which is closed under taking suffixes and internal composition. We further define a family of trace categories $\mathcal{T}_{\mathcal{A}}$ whose morphisms are relations between finite sets annotated with an *activation algebra* \mathcal{A} : a finite \vee -semilattice with a distinguished *activation element* α . A path through $\mathcal{T}_{\mathcal{A}}$ satisfies the trace condition if it carries an infinite sequence of connected elements which attains the activation value α infinitely often. Almost all notions of trace from the literature can be represented naturally in terms of some $\mathcal{T}_{\mathcal{A}}$. Notably, the usual trace condition for the cyclic systems of μ -calculus is given in terms of the *three-value failure algebra* \mathbb{F} .

Using this abstracted notion of trace, a pre-proof with underlying graph C may be represented as an annotation of C with maps of a trace category \mathcal{T} . In categorical terms, the latter is a functor $\text{Tr} : \mathcal{P}_C \rightarrow \mathcal{T}$ from the category of finite paths through C into \mathcal{T} . When representing a pre-proof in this manner, all details unrelated to the trace condition are abstracted away, leading us to call such functors *abstract cyclic derivation*. The requirement of trace conditions being closed under internal composition allows us to prove the following two novel results in cyclic proof theory.

- *Compression*: ACDs may be ‘compressed’ to trace-condition-equivalent ACDs over graphs whose size is linear in the number of simple cycles in C (Theorem 4.1).
- *Ramsey-style soundness*: An ACD $\text{Tr} : \mathcal{P}_C \rightarrow \mathcal{T}$ is a proof (i.e., every path through it satisfies the trace condition) if and only if for every idempotent endomorphism $R : X \rightarrow X$ in the image of Tr , the periodic path $R^\omega : \omega \rightarrow \mathcal{T}$ (which simply repeats R) satisfies the trace condition (Theorem 5.2).

We also prove some well-known results of cyclic proof theory in terms of our abstract notion of trace and derivation in order to demonstrate the adequacy of our notions for tackling questions

of cyclic proof theory and connect them to the wider field. In Section 6, we consider trace categories with ω -regular trace conditions, connecting ACDs with ω -automata theory. As a result of the Ramsey-style soundness condition, every trace category with finite HOM-sets, including all \mathcal{T}_A , has an ω -regular trace condition. We prove the following results using automata theory.

- *Decidability*: If the trace condition of \mathcal{T} is ω -regular in a ‘computable’ manner, the GTC of ACDs on \mathcal{T} is decidable (Theorem 6.1).
- *Regularisation*: If a proof system is finite (i.e., the set of sequents occurring in a proof is finite), then every sequent provable via a ill-founded proof is provable via a cyclic proof (Theorem 6.3).

In Section 7, we generalise a result about the cyclic proof system for multiplicative-additive linear logic with fixed points (μ MALL) by Nolle et al. (2019) to obtain.

- *PSPACE-completeness*: The problem of deciding whether an ACD over \mathcal{T}_A satisfies the GTC is PSPACE-complete (Theorem 7.1 & Lemma 7.4).

The above result is proven in such a way that the decision problem being in PSPACE readily transfers to concrete cyclic proof systems with a \mathcal{T}_A -trace condition. Transferring PSPACE-hardness, however, requires certain assumptions on the concrete cyclic proof system.

8.1 Related work

Definitions and results in this article originate from Wehr’s Masters thesis (Wehr 2021), although most have undergone significant changes. The thesis contains results that are not presented in this article, including a categorical treatment of a further three cyclic proof systems: cyclic arithmetic (Simpson 2017), $\text{HFL}_{\mathbb{N}}$ (Kori et al. 2021) and Grzegorzczuk modal logic (Savateev and Shamkanov 2021).

To the best of our knowledge, Brotherston (2006) is the only previous work on abstracting cyclic proofs. The present article can be viewed as an extension of Brotherston’s in two ways: first, his abstract notion of derivation is closer to the common definition of pre-proofs, being presented in terms of abstract sequents and derivation rules. Crucially, this bars him from considering the composition of trace information, which enabled us to derive the ACD compression result (Theorem 4.1) and the alternative soundness condition (Theorem 5.2). Second, Brotherston only considers trace conditions expressed in terms of the Booleans \mathbb{B} which, as we have remarked, is insufficient to directly express the trace condition of the modal μ -calculus, a problem our activation algebras alleviate.

Lee et al. (2001) propose a termination criterion for first-order programs called the *size-change principle*. This criterion can be considered an instance of the $\mathcal{T}_{\mathbb{B}}$ trace condition. The authors give a decision procedure for the size-change principle which is a variant of the decision procedure for the Ramsey trace condition we give in Theorem 5.3, specialised to $\mathcal{T}_{\mathbb{B}}$. They also prove PSPACE-completeness of their criterion by reducing it to BOOLE termination. This argument has been adapted by Nolle et al. (2019) to the cyclic proof system for multiplicative, additive linear logic with fixed points (μ MALL) which we in turn adapted to ACDs in Section 7.

The Ramsey trace condition (Definition 5.1) is not the first application of Ramsey’s theorem in the field of cyclic proof theory. Notably, existing proofs of the equivalence of (extensions of) cyclic arithmetic and (extensions of) Peano arithmetic (Berardi and Tatsuta 2017; Das 2020; Simpson 2017) rely on (arithmetised) variants of Ramsey’s theorem for concluding provability in Peano arithmetic from cyclic provability. In their proofs, it is applied to ‘internalise’ the soundness justification given by the GTC into non-cyclic PA proofs.

Cohen et al. (2024) use a variant of Brotherston’s framework, that is, $\mathcal{T}_{\mathbb{B}}$ in our setting, to carry out a parameterised analysis of the worst-case complexity of various methods for checking soundness of cyclic proofs. The two parameters they considered, presented in terms of an ACD $(C, \text{Tr} : \mathcal{P}_C \rightarrow \mathcal{T}_{\mathbb{B}})$ with $C = (T, \beta)$, are *vertex count* $n = |T|$ and *vertex width* $w = \max_{t \in T} |\text{Tr}(t)|$. They find that many checking procedures, such as those based on automata akin to those sketched in Remark 6.2 and the Ramsey decision procedure of Section 5, are polynomial in vertex count n and exponential in vertex width w . They also propose an optimisation to the Ramsey decision procedure, called Order-reduced Transitive Looping procedure, which reduces the exponential degree in the vertex width w . We conjecture that all of their results can be extended to arbitrary $\mathcal{T}_{\mathcal{A}}$.

In some sense, every cyclic proof system in the literature might be considered related work, as one can ask whether its trace condition can be modelled by our formalism. For most cyclic proof system we are aware of, including all those referred to in this article, this seems to be the case with three exceptions: The first is the *bouncing-thread trace condition* of Baelde et al. (2022) which considers traces that do not reside along the branches of a pre-proof. The second is the *limit condition*, presented by Hazard and Kuperberg (2022) in their system for transfinite word languages, which has traces running through multiple separate pre-proofs. The third are the *higher-dimensional trace conditions* of the hypersequent calculi of Das and Girlando (2022) and Afshari et al. (2023).

Since the publication of the conference version of this article, further work using the proposed framework of \mathcal{A} -activated categories has been carried out by Leigh and Wehr (2023). The article covers reset proof systems, cyclic proof systems with a soundness that can be verified in polynomial time and which have proven fruitful for proof-theoretic investigation (see, e.g., Afshari and Leigh 2017; Afshari et al. 2021; Marti and Venema 2021). In the article, it is shown that for each cyclic proof system whose GTC is specified in terms of some $\mathcal{T}_{\mathcal{A}}$, there exists an associated reset proof system which proves the same theorems. To obtain the reset proof system, the original proof system is equipped with an annotation mechanism which is derived from the trace sets and trace maps of the trace interpretation. In this application, the ‘naturalness’ of a trace interpretation culminates in a natural reset proof system.

Acknowledgements. We thank the anonymous referees for their insightful comments and suggestions. This work was supported by the Knut and Alice Wallenberg Foundation [2020.0199, 2015.0179] and the Swedish Research Council [2017-05111, 2016-03502].

References

- Afshari, B. and Leigh, G. E. (2017). Cut-free completeness for modal Mu-calculus. In: *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 1–12.
- Afshari, B., Leigh, G. E. and Menéndez Turata, G. (2021). Uniform interpolation from cyclic proofs: The case of modal Mu-calculus. In: *Automated Reasoning with Analytic Tableaux and Related Methods*, vol. 12842, Cham, Springer International Publishing, 335–353.
- Afshari, B., Leigh, G. E. and Turata, G. M. (2023). A cyclic proof system for full computation tree logic. In: Klin, B. and Pimentel, E. (eds.) *31st EACSL Annual Conference on Computer Science Logic, CSL 2023, February 13–16, 2023, Warsaw, Poland*, LIPIcs, vol. 252, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 5:1–5:19.
- Afshari, B. and Wehr, D. (2022). Abstract cyclic proofs. In: Ciabattoni, A., Pimentel, E. and de Queiroz, R. J. G. B. (eds.) *Logic, Language, Information, and Computation*, Lecture Notes in Computer Science, Springer International Publishing, 309–325.
- Baelde, D., Doumane, A., Kuperberg, D. and Saurin, A. (2022). Bouncing threads for circular and non-wellfounded proofs: Towards compositionality with circular proofs. In: Baier, C. and Fisman, D. (eds.) *LICS’22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2–5, 2022*, ACM, 63:1–63:13.
- Baelde, D., Doumane, A. and Saurin, A. (2016). Infinitary proof theory: The multiplicative additive case. In: Talbot, J.-M. and Regnier, L. (eds.) *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, Leibniz International Proceedings in Informatics (LIPIcs), vol. 62, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 42:1–42:17. ISSN: 1868-8969.
- Berardi, S. and Tatsuta, M. (2017). Equivalence of inductive definitions and cyclic proofs under arithmetic. In: *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 1–12.
- Brotherston, J. (2006). *Sequent Calculus Proof Systems for Inductive Definitions*. Phd thesis, University of Edinburgh.

- Brotherston, J., Distefano, D. and Petersen, R. L. (2011). Automated cyclic entailment proofs in separation logic. In: Björner, N. and Sofronie-Stokkermans, V. (eds.) *Automated Deduction – CADE-23*, Lecture Notes in Computer Science, Berlin, Heidelberg, Springer, 131–146.
- Brotherston, J., Gorogiannis, N. and Petersen, R. L. (2012). A generic cyclic theorem prover. In: Jhala, R. and Igarashi, A. (eds.) *Programming Languages and Systems*, Lecture Notes in Computer Science, Berlin, Heidelberg, Springer, 350–367.
- Cohen, L., Jabarin, A., Popescu, A. and Rowe, R. N. S. (2024). The complex(ity) landscape of checking infinite descent. *Proceedings of the ACM on Programming Languages* **8**. (conditionally accepted).
- Das, A. (2020). On the logical complexity of cyclic arithmetic. *Logical Methods in Computer Science* **16** (1) 1:1–1:39.
- Das, A. (2021). On the logical strength of confluence and normalisation for cyclic proofs. In: Kobayashi, N. (ed.) *6th International Conference on Formal Structures for Computation and Deduction (FSCD 2021)*, Leibniz International Proceedings in Informatics (LIPIcs), vol. 195, Dagstuhl, Germany, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 29:1–29:23.
- Das, A. and Giraldo, M. (2022). Cyclic proofs, hypersequents, and transitive closure logic. In: Blanchette, J., Kovács, L. and Pattinson, D. (eds.) *Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8–10, 2022, Proceedings*, Lecture Notes in Computer Science, vol. 13385, Springer, 509–528.
- Fortier, J. and Santocanale, L. (2013). Cuts for circular proofs: Semantics and cut-elimination. In: Rocca, S. R. D. (ed.) *Computer Science Logic 2013 (CSL 2013)*, Leibniz International Proceedings in Informatics (LIPIcs), vol. 23, Dagstuhl, Germany, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 248–262.
- Hazard, E. and Kuperberg, D. (2022). Cyclic proofs for transfinite expressions. In: Manea, F. and Simpson, A. (eds.) *30th EACSL Annual Conference on Computer Science Logic (CSL 2022)*, Leibniz International Proceedings in Informatics (LIPIcs), vol. 216, Schloss Dagstuhl Leibniz-Zentrum für Informatik, 23:1–23:18. ISSN: 1868-8969.
- Joyal, A., Street, R. and Verity, D. (1996). Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society* **119** (3) 447–468.
- Jungteerapanich, N. (2009). A tableau system for the modal μ -calculus. In: Giese, M. and Waaler, A. (eds.) *Automated Reasoning with Analytic Tableaux and Related Methods*, Lecture Notes in Computer Science, Berlin, Heidelberg, Springer, 220–234.
- Kori, M., Tsukada, T. and Kobayashi, N. (2021). A cyclic proof system for HFLN. In: Baier, C. and Goubault-Larrecq, J. (eds.) *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, Leibniz International Proceedings in Informatics (LIPIcs), vol. 183, Dagstuhl, Germany, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 29:1–29:22.
- Kozen, D. (1983). Results on the propositional μ -calculus. *Theoretical Computer Science* **27** (3) 333–354. Number: 3 Publisher: Elsevier.
- Kupferman, O. and Vardi, M. Y. (1996). Verification of fair transition systems. In: Alur, R. and Henzinger, T. A. (eds.) *Computer Aided Verification*, Lecture Notes in Computer Science, Springer, 372–382.
- Lee, C. S., Jones, N. D. and Ben-Amram, A. M. (2001). The size-change principle for program termination. In *Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL'01, New York, NY, USA, Association for Computing Machinery, 81–92.
- Leigh, G. E. and Wehr, D. (2023). From GTC to Reset: Generating Reset Proof Systems from Cyclic Proof Systems. (arXiv:2301.07544).
- Marti, J. and Venema, Y. (2021). A focus system for the alternation-free μ -calculus. In: Das, A. and Negri, S. (eds.) *Automated Reasoning with Analytic Tableaux and Related Methods - 30th International Conference, TABLEUX 2021, Birmingham, UK, September 6–9, 2021, Proceedings*, Lecture Notes in Computer Science, vol. 12842, Springer, 371–388.
- McNaughton, R. (1966). Testing and generating infinite sequences by a finite automaton. *Information and Control* **9** (5) 521–530.
- Nier, F. (2002). Nondeterministic tree automata. In: Grädel, E., Thomas, W. and Wilke, T. (eds.) *Automata, Logics, and Infinite Games: A Guide to Current Research*, Lecture Notes in Computer Science, Springer-Verlag.
- Niwiński, D. and Walukiewicz, I. (1996). Games for the μ -calculus. *Theoretical Computer Science* **163** (1–2) 99–116.
- Nollet, R., Saurin, A. and Tasson, C. (2019). PSPACE-completeness of a thread criterion for circular proofs in linear logic with least and greatest fixed points. In: Cerrito, S. and Popescu, A. (eds.) *Automated Reasoning with Analytic Tableaux and Related Methods*, Lecture Notes in Computer Science, Cham, Springer International Publishing, 317–334.
- Rabin, M. O. (1969). Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society* **141** 1–35.
- Ramsey, F. P. (1930). On a problem of formal logic. *Proceedings of the London Mathematical Society* **s2-30** (1) 264–286.
- Savateev, Y. and Shamkanov, D. (2021). Non-well-founded proofs for the Grzegorzczuk modal logic. *The Review of Symbolic Logic* **14** (1) 22–50.
- Simpson, A. (2017). Cyclic arithmetic is equivalent to Peano arithmetic. In: Esparza, J. and Murawski, A. S. (eds.) *Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science, Berlin, Heidelberg, Springer, 283–300.
- Sprenger, C. and Dam, M. (2003). On global induction mechanisms in a μ -calculus with explicit approximations. *RAIRO - Theoretical Informatics and Applications* **37** (4) 365–391.
- Stirling, C. (2013). A proof system with names for modal μ -calculus. *Electronic Proceedings in Theoretical Computer Science* **129** 18–29.

- Tellez, G. and Brotherston, J. (2017). Automatically verifying temporal properties of pointer programs with cyclic proof. In: de Moura, L. (ed.) *Automated Deduction – CADE 26*, Lecture Notes in Computer Science, Cham, Springer International Publishing, 491–508.
- Wehr, D. (2021). *An Abstract Framework for the Analysis of Cyclic Derivations*. MSc thesis, University of Amsterdam.