Tree Automata as Algebras: Minimisation and Determinisation

Gerco van Heerdt
University College London, United Kingdom
gerco.heerdt@ucl.ac.uk

Tobias Kappé
University College London, United Kingdom
tkappe@cs.ucl.ac.uk

Jurriaan Rot
University College London, United Kingdom and Radboud University, The Netherlands
jrot@cs.ru.nl

Matteo Sammartino
University College London, United Kingdom
m.sammartino@ucl.ac.uk

Alexandra Silva
University College London, United Kingdom
alexandra.silva@ucl.ac.uk

Abstract

We study a categorical generalisation of tree automata, as Σ-algebras for a fixed endofunctor Σ endowed with initial and final states. Under mild assumptions about the base category, we present a general minimisation algorithm for these automata. We then build upon and extend an existing generalisation of the Nerode equivalence to a categorical setting and relate it to the existence of minimal automata. Finally, we show that generalised types of side-effects, such as non-determinism, can be captured by this categorical framework, leading to a general determinisation procedure.

2012 ACM Subject Classification Theory of Computation → Formal languages and automata theory

Keywords and phrases tree automata, algebras, minimisation, determinisation, Nerode equivalence

Digital Object Identifier 10.4230/LIPIcs...

Funding This work was partially supported by the ERC Starting Grant ProFoundNet (grant code 679127), a Leverhulme Prize (PLP–2016–129) and a Marie Curie Fellowship (grant code 795119).

1 Introduction

Automata have been extensively studied using category theory, both from an algebraic and a coalgebraic perspective [26, 6, 43, 40]. Categorical insights have enabled the development of generic algorithms for minimisation [4], determinisation [45], and equivalence checking [15].

A fruitful line of work has focussed on characterising the semantics of different types of automata as final coalgebras. The final coalgebra contains unique representatives of behaviour, and the existence of a minimal automaton can be formalised by a suitable factorisation of the map from a given automaton into the final coalgebra. Algorithms to compute the minimal automaton can be devised based on the final sequence, which yields procedures resembling classical partition refinement [32, 18]. Unfortunately, bottom-up tree automata do not fit the abstract framework of final coalgebras.[1] This impeded the application of abstract

[1] The language semantics of top-down tree automata represented as coalgebras is given in [30], based on a transformation to bottom-up tree automata. In this paper, we focus on bottom-up automata only.

Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

algorithms for minimisation, determinisation, and equivalence. We embrace the categorical *algebraic* view on automata due to Arbib and Manes [8] to study bottom-up tree automata (Section 3). This algebraic approach is also treated in detail by Adámek and Trnková [6], who, among other results, give conditions under which minimal realisations exist (see also [3]) and constructions to determinise partial and non-deterministic bottom-up tree automata. However, generic *algorithms* for minimisation, and a more abstract and uniform picture of determinisation, have not been studied in this context.

The contributions of this paper are three-fold. First, we explore the notion of *cobase* to devise an iterative construction for minimising tree automata, at the abstract level of algebras, resembling partition refinement (Section 4). The notion of cobase is dual to that of base [11], which plays a key role in reachability of coalgebras [46, 10] and therefore in minimisation of automata. Second, we study a different characterisation of minimality via the Nerode equivalence, again based on work of Arbib and Manes [8], and provide a generalisation using monads that allows to treat automata with equations (Section 5). Third, we extend bottom-up tree automata to algebras in the Kleisli category of a monad, which enables us to study tree automata enriched with side-effects and derive an associated determinisation procedure (Section 6). We demonstrate the generality of our approach by applying it both to classical examples—deterministic, non-deterministic, and multiplicity/weighted tree automata—and to a novel kind of tree automata, namely *nominal* tree automata.

2 Preliminaries

We assume basic knowledge of category theory. Throughout this paper, we fix a category C.

Monads A *monad* on C is a triple (T, η, μ) consisting of an endofunctor T on C and two natural transformations: a *unit* $\eta\colon \mathsf{Id} \Rightarrow T$ and a *multiplication* $\mu\colon T^2 \Rightarrow T$, which satisfy the compatibility laws $\mu \circ \eta_T = \mathsf{id}_T = \mu \circ T\eta$ and $\mu \circ \mu_T = \mu \circ T\mu$.

▶ **Example 2.1.** The triple $(\mathcal{P}_\mathsf{f}, \{-\}, \bigcup)$ is a monad on Set, where $\mathcal{P}_\mathsf{f}$ is the finite powerset functor , $\{-\}$ is the singleton operation, and $\bigcup$ is union of sets. Another example is the *multiplicity monad* $(\mathcal{M}_\mathbb{F}, e, m)$ for a field $\mathbb{F}$, where $\mathcal{M}_\mathbb{F}$ is the functor sending a set X to $\mathcal{M}_\mathbb{F}X = \{\varphi\colon X \to \mathbb{F} \mid \varphi \text{ has finite support}\}$. An element φ can be seen as a formal finite sum $\sum_i s_i x_i$, where each x_i has multiplicity s_i. The unit e sends x to $1x$ and the multiplication is $m_X(\sum_i s_i\varphi_i)(x) = \sum_i s_i \cdot \varphi_i(x)$, where $\cdot$ is the field multiplication.

Algebras and Varietors We fix a functor $\Sigma : \mathsf{C} \to \mathsf{C}$ and write $\mathsf{Alg}(\Sigma)$ for the category of Σ-algebras. Throughout this paper, we assume that Σ is a *varietor* [6], i.e., that the forgetful functor $U\colon \mathsf{Alg}(\Sigma) \to \mathsf{C}$ admits a left adjoint $F\colon \mathsf{C} \to \mathsf{Alg}(\Sigma)$. The varietor Σ induces a monad $(\Sigma^\diamond, \eta, \mu)$ on C, where $\Sigma^\diamond = UF$. Given an object X of C, we refer to $FX = (\Sigma^\diamond X, \alpha_X)$ as the *free* Σ*-algebra* over X. The free Σ-algebra satisfies the following: for every Σ-algebra Q and every morphism $x\colon X \to UQ$ of C, there is a unique Σ-algebra morphism $x^\sharp\colon (\Sigma^\diamond X, \alpha_X) \to Q$ with $U(x^\sharp) \circ \eta_X = x$.

▶ **Example 2.2.** A functor is *finitary* if it preserves filtered colimits. Any finitary Set functor is a varietor: free algebras over a set X can be obtained as a colimit of a transfinite sequence [2, 28]. A *polynomial functor* on Set is a functor inductively defined by $P := \mathsf{id} \mid A \mid P_1 \times P_2 \mid \coprod_{i\in I} P_i$ where A is any constant functor. Polynomial functors are finitary and therefore varietors.

3 Tree automata, categorically

In this section we start our categorical investigation of (bottom-up) tree automata. We first discuss a general notion of automaton over an endofunctor Σ due to Arbib and Manes [8] and then discuss how this notion can be instantiated to obtain various kinds of automata.

▶ **Definition 3.1** (Σ-tree automaton). *A Σ-tree automaton over objects I and O in C is a tuple (Q, δ, i, o) such that (Q, δ) is a Σ-algebra and $i\colon I \to Q$ and $o\colon Q \to O$ are morphisms of C. The objects I and O are referred to as the* input object *and* output object *respectively. A* homomorphism *from an automaton (Q, δ, i, o) to an automaton (Q', δ', i', o') is a Σ-algebra homomorphism $h\colon Q \to Q'$ (i.e., $\delta' \circ \Sigma h = h \circ \delta$) such that $h \circ i = i'$ and $o' \circ h = o$.*

Throughout this paper, we fix input and output objects I and O respectively. If Σ is clear from the context we sometimes refer to a Σ-tree automaton simply as an automaton.

A *language* (over Σ) is a morphism $L\colon \Sigma^\diamond I \to O$. In the context of an automaton $\mathcal{A} = (Q, \delta, i, o)$, we can think of $U(i^\sharp)\colon \Sigma^\diamond I \to Q$, induced by the free Σ-algebra FI on I, as the *reachability map*, telling us which state is reached by parsing an element of the free algebra over I. We will write $\mathsf{rch}_\mathcal{A}$ (or rch if $\mathcal{A}$ is obvious) for $U(i^\sharp)$. The *language of* $\mathcal{A}$ is the morphism $\mathcal{L}(\mathcal{A})\colon \Sigma^\diamond I \to O$ in C given by $\mathcal{L}(\mathcal{A}) = o \circ \mathsf{rch}_\mathcal{A}$.

▶ **Example 3.2** (Deterministic bottom-up tree automata). Let us see how Σ-tree automata can capture deterministic bottom-up tree automata. We first recall some basic concepts.

A *ranked alphabet* is a finite set of symbols Γ, where each $\gamma \in \Gamma$ is equipped with an *arity* $\mathsf{ar}(\gamma) \in \mathbb{N}$. A *frontier alphabet* is a finite set of symbols I. The set of Γ-*trees* over I, denoted $\mathcal{T}_\Gamma(I)$, is the smallest set such that $I \subseteq \mathcal{T}_\Gamma(I)$, and for all $\gamma \in \Gamma$ we have that $t_1, \dots, t_{\mathsf{ar}(\gamma)} \in \mathcal{T}_\Gamma(I)$ implies $(\gamma, t_1, \dots, t_{\mathsf{ar}(\gamma)}) \in \mathcal{T}_\Gamma(I)$. In other words, $\mathcal{T}_\Gamma(I)$ consists of finite trees with leaves labelled by symbols from I and internal nodes labelled by symbols from Γ; the number of children of each internal node matches the arity of its label.

A ranked alphabet Γ gives rise to a polynomial *signature endofunctor* $\Sigma\colon \mathsf{Set} \to \mathsf{Set}$ given by $\Sigma X = \coprod_{\gamma\in\Gamma} X^{\mathsf{ar}(\gamma)}$. A *deterministic bottom-up tree automaton* is a Σ-tree automaton $\mathcal{A} = (Q, \delta, i, o)$ where Q is finite, Σ is a signature endofunctor, and $O = 2$. Here Q is the set of *states*, $i\colon I \to Q$ is the *initial assignment*, $o\colon Q \to 2$ is the characteristic function of *final* states, and for each $\gamma \in \Gamma$ we have a transition function $\delta_\gamma = \delta \circ \kappa_\gamma\colon Q^{\mathsf{ar}(\gamma)} \to Q$. The *language* $\mathcal{L}(\mathcal{A})$ is the set of all Γ-trees t such that $(o \circ \hat{\delta})(t) = 1$, where $\hat{\delta}\colon \mathcal{T}_\Gamma(I) \to Q$ extends δ to trees by structural recursion:

$$\hat{\delta}(\ell) = i(\ell) \quad (\ell \in I) \qquad\qquad \hat{\delta}(\gamma, t_1, \dots, t_k) = \delta_\gamma(\hat{\delta}(t_1), \dots, \hat{\delta}(t_k))$$

In other words, $\mathcal{L}(\mathcal{A})$ contains the trees that evaluate to a final state. The map $\hat{\delta}$ above is the transpose $i^\sharp$ in the relevant adjunction between Set and $\mathsf{Alg}(\Sigma)$, where the left adjoint sends a set I to the Σ-algebra with carrier $\mathcal{T}_\Gamma(I)$ and the obvious structure map.

3.1 Nominal tree automata

To show the versatility of our definition, we instantiate it in the category Nom of nominal sets and equivariant functions. This results in a notion of nominal tree automaton—along the lines of nominal automata theory [13]—which, as we shall see below, provides a useful model for languages of trees with variables and variable binding. We first recall some basic notations of nominal set theory [41]. Let $\mathbb{A}$ be a countable set of *atoms*, and let $\mathsf{Sym}(\mathbb{A})$ be the associated symmetry group, consisting of all permutations on $\mathbb{A}$. A *nominal set* is a pair $(X, \cdot)$ of a set X and a function $\cdot\colon \mathsf{Sym}(\mathbb{A}) \times X \to X$ forming a left action of $\mathsf{Sym}(\mathbb{A})$ on X.

Each $x \in X$ is required to have *finite support*, i.e., there must exist a finite $A \subseteq \mathbb{A}$ such that for all $\pi \in \mathsf{Sym}(\mathbb{A})$, if π is equal to id_A when restricted to A, then $\pi \cdot x = x$. The minimal such A is denoted $\mathsf{supp}(x)$, and can be understood as the set of "free" names of x. Given $x \in X$, its *orbit* is the set $\{\pi \cdot x \mid \pi \in \mathsf{Sym}(\mathbb{A})\}$. We say that a nominal set X is *orbit-finite* whenever it has finitely many orbits. An *equivariant function* $f\colon (X, \cdot) \to (Y, \cdot)$ is a function $X \to Y$ that respects permutations, i.e., $f(\pi \cdot x) = \pi \cdot f(x)$.

Polynomial functors in Nom support additional operations [20], such as the *name abstraction* functor $[\mathbb{A}]\colon \mathsf{Nom} \to \mathsf{Nom}$, which "binds" a name in the support. For instance, if $x \in X$, then $\langle a \rangle x \in [\mathbb{A}]X$, with $\mathsf{supp}(\langle a \rangle x) = \mathsf{supp}(x) \setminus \{a\}$. The element $\langle a \rangle x$ should be thought of as an equivalence class *up to* α*-conversion* w.r.t. the binder $\langle a \rangle$. We can then define tree automata for parsing trees with binders. Consider for instance $\Sigma_\lambda\colon \mathsf{Nom} \to \mathsf{Nom}$ given by

$$\Sigma_\lambda X = \underbrace{X \times X}_{\mathsf{appl}} + \underbrace{[\mathbb{A}]X}_{\mathsf{lambda}}$$

describing the syntax of the λ-calculus [21]. This functor is finitary [20], which implies the existence of free algebras. Fixing $I = \mathbb{A}$, the carrier of the free Σ_λ-algebra over I consists of parse trees for λ-terms (up to α-conversion) with variables in $\mathbb{A}$. We can then define automata parsing these trees as $\mathcal{A} = (Q, \delta, i, o)$, where

- Q is a nominal set;
- δ consists of two equivariant functions $\delta_{\mathsf{appl}}\colon Q \times Q \to Q$ and $\delta_{\mathsf{lambda}}\colon [\mathbb{A}]Q \to Q$;
- $i\colon \mathbb{A} \to Q$ is an equivariant function, selecting states for parsing variables;
- $o\colon Q \to 2$ is an equivariant characteristic function of final states, which implies that if a state is final, so are all the states in its orbit.

The reachability function is the equivariant function given by

$$\mathsf{rch}_{\mathcal{A}}(t) = \begin{cases} i(t) & \text{if } t \in \mathbb{A} \\ \delta_{\mathsf{appl}}(\mathsf{rch}_{\mathcal{A}}(t_1), \mathsf{rch}_{\mathcal{A}}(t_2)) & \text{if } t = (t_1, t_2) \\ \delta_{\mathsf{lambda}}(\langle a \rangle \mathsf{rch}_{\mathcal{A}}(t')) & \text{if } t = \langle a \rangle t'. \end{cases}$$

The most interesting case is the last one: in order to parse the α-equivalence class $\langle a \rangle t'$, we first parse any tree t' such that $\langle a \rangle t'$ is in the class, and then we take the resulting state up to α-conversion w.r.t. $\langle a \rangle$. Note that $\mathcal{L}(\mathcal{A})$ is equivariant, i.e., invariant under permutations of atoms. Thus $\mathcal{A}$ recognises λ-trees up to bijective renamings of variables.

4 Minimisation

In this section we define a construction that allows to minimise a given tree automaton. We start with a few basic preliminary notions related to quotients and factorisation systems.

Factorisations An $(\mathcal{E}, \mathcal{M})$*-factorisation system* on C consists of classes of morphisms $\mathcal{E}$ and $\mathcal{M}$, closed under composition with isos, such that for every morphism f in C there exist $e \in \mathcal{E}$ and $m \in \mathcal{M}$ with $f = m \circ e$, and we have a unique diagonal fill-in property.

We list a few properties of factorisation systems. First, both $\mathcal{E}$ and $\mathcal{M}$ are closed under composition. Furthermore, if $g \circ f \in \mathcal{E}$ and $f \in \mathcal{E}$, then $g \in \mathcal{E}$. Lastly, if $\mathcal{E}$ consists of epimorphisms, then it is closed under cointersections, i.e., wide pushouts of epimorphisms [5]. A functor Σ is said to *preserve* $\mathcal{E}$*-cointersections* if it preserves wide pushouts of epimorphisms in $\mathcal{E}$. In that case, for an epimorphism e, if $e \in \mathcal{E}$ then Σe is again an epimorphism.

Quotients Define by $\leq$ the order on morphisms with common domain given by $f \leq g$ iff $\exists h. g = h \circ f$. This induces an equivalence relation on such morphisms. A *quotient* of an object X is an epimorphism $q\colon X \twoheadrightarrow X'$ identified up to the equivalence, i.e., an equivalence class. We denote by $\mathsf{Quot}(X)$ the class of all quotients of X. The underlying category C is said to be *cowellpowered* if $\mathsf{Quot}(X)$ is a set for every X. In that case, if C is cocomplete, $\mathsf{Quot}(X)$ forms a complete lattice, with the order given by $\leq$, and the least upper bound (join) given by cointersection. We denote by $\mathsf{Quot}_{\mathcal{E}}(X)$ the set of quotients of X that are in $\mathcal{E}$. (This is well-defined because $\mathcal{E}$ is closed under isomorphisms.)

▶ **Assumption 4.1.** Throughout this section, C is cocomplete and cowellpowered. Moreover, we fix an $(\mathcal{E}, \mathcal{M})$-factorisation system in C, where $\mathcal{E}$ contains epimorphisms only.

▶ Remark 4.2. The category Set is cocomplete and cowellpowered, and so is Nom introduced in Section 3.1. In general, the existence of an (epi, strong mono)-factorisation system already follows from C being cocomplete and cowellpowered [16]. Allowing a more general choice of factorisation system will be useful in Section 5, where we work with a different $\mathcal{E}$.

Let (Q, δ) be a Σ-algebra. A *quotient algebra* is a Σ-algebra (Q', δ') together with a quotient $q\colon Q \twoheadrightarrow Q'$ in $\mathcal{E}$ that is an algebra homomorphism. Given a Σ-tree automaton (Q, δ, i, o), a *quotient automaton* is a Σ-tree automaton (Q', δ', i', o') together with a quotient $q\colon Q \twoheadrightarrow Q'$ in $\mathcal{E}$ that is a homomorphism of automata.

▶ **Definition 4.3** (Minimisation)**.** *The* minimisation *of a Σ-tree automaton (Q, δ, i, o) is a quotient automaton $(Q_m, \delta_m, i_m, o_m)$, $q\colon Q \twoheadrightarrow Q_m$, such that for any quotient automaton (Q', δ', i', o'), $q'\colon Q \twoheadrightarrow Q'$ of (Q, δ, i, o) there exists a (necessarily unique) automaton homomorphism $h\colon Q' \twoheadrightarrow Q_m$ such that $h \circ q' = q$.*

Minimisation is called *minimal reduction* in [6]. Note that the morphism h in the definition of minimisation is in $\mathcal{E}$, since q' and q are. In the sequel, we sometimes refer to a quotient $q\colon Q \twoheadrightarrow Q_m$ as the minimisation if there exist δ_m, i_m, o_m turning $(Q_m, \delta_m, i_m, o_m), q$ into the minimisation of (Q, δ, i, o).

▶ **Definition 4.4.** *A Σ-tree automaton $\mathcal{A}$ is said to be* reachable *if the associated reachability map* rch *is in $\mathcal{E}$. It is* minimal *if it is reachable and for every reachable Σ-tree automaton $\mathcal{A}'$ s.t. $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$ there exists a (necessarily unique) homomorphism from $\mathcal{A}'$ to $\mathcal{A}$.*

The above definition of minimality relies on reachability; a more orthogonal (but equivalent) definition of minimality is explored in Section 4.2.

We conclude with a few observations on the connection between minimisation and minimality, treated in detail in [6]. We say Σ *admits minimisation* of reachable automata if every reachable automaton over Σ has a minimisation.

▶ **Lemma 4.5.** *An automaton $\mathcal{A}$ is minimal iff it is the minimisation of $(\Sigma^\diamond I, \alpha_I, \eta_I, \mathcal{L}(\mathcal{A}))$.*

▶ **Lemma 4.6.** *The functor Σ admits minimisation of reachable automata if and only if a minimal automaton exists for every language over Σ. In that case, if $\mathcal{A}$ is reachable, then the minimisation of $\mathcal{A}$ is minimal.*

Proof. For the equivalence, the implication left to right follows from Lemma 4.5. For the converse, one readily shows that the minimisation of an automaton $\mathcal{A}$ is given by the minimal automaton accepting $\mathcal{L}(\mathcal{A})$. The second statement holds by uniqueness of minimisations. ◀

4.1 Minimisation via the cobase

We show how to compute the minimisation of a given automaton (Q, δ, i, o) using the so-called *cobase* [11]. This is the dual of the base, which is used in [10, 46] for reachability of coalgebras. The cobase allows us to characterise the minimisation as the greatest fixed point of a certain monotone operator on $\mathsf{Quot}_{\mathcal{E}}(Q)$, which is a complete lattice by Assumption 4.1.

▶ **Definition 4.7.** *Let $f\colon \Sigma X \to Y$ be a morphism. The* ($\mathcal{E}$)-cobase *of f (if it exists) is the greatest quotient $q \in \mathsf{Quot}_{\mathcal{E}}(X)$ such that there exists a morphism g with $g \circ \Sigma q = f$.*

A concrete instance of the cobase will be given below in Example 4.11. The cobase can be computed as the join of all quotients satisfying the relevant condition, provided that the functor preserves cointersections.

▶ **Theorem 4.8** (Existence of cobases)**.** *Suppose $\Sigma\colon \mathsf{C} \to \mathsf{C}$ preserves $\mathcal{E}$-cointersections. Then every map $f\colon \Sigma X \to Y$ has an $\mathcal{E}$-cobase, given by the cointersection*

$$\bigvee \{q \in \mathsf{Quot}_{\mathcal{E}}(X) \mid \exists g.\, g \circ \Sigma q = f\}\,.$$

Proof. For $\mathcal{E}$ the class of all epis, the dual is shown in [10, 46]. The proof goes through in the current, more general setting, using that $\mathcal{E}$ is closed under cointersections. ◀

▶ Remark 4.9. A Set functor preserves cointersections iff it is finitary [6]. In particular, this is the case for polynomial functors. For Nom functors we can use that, in general, a functor preserves cointersections if it is finitary and preserves reflexive coequalisers [6]. These conditions hold for polynomial Nom functors introduced in Section 3.1, because they preserve sifted colimits [34], which include filtered colimits and reflexive coequalisers.

We now define an operator on quotients of the state space of an automaton, which characterises the minimisation of an automaton and gives a way of computing it. To this end, given a Σ-algebra (Q, δ) and a quotient $q\colon Q \twoheadrightarrow Q' \in \mathsf{Quot}_{\mathcal{E}}(Q)$, define the quotient $\Theta_\delta(q)\colon Q \twoheadrightarrow \Theta_\delta(Q')$ as the cobase of $q \circ \delta$. This defines a monotone operator $\Theta_\delta\colon \mathsf{Quot}_{\mathcal{E}}(Q) \to \mathsf{Quot}_{\mathcal{E}}(Q)$ that has the following important property (see [10, 46]):

▶ **Lemma 4.10.** *Suppose Σ preserves $\mathcal{E}$-cointersections. For any Σ-algebra (Q, δ), a quotient $q\colon Q \twoheadrightarrow Q'$ in $\mathsf{Quot}_{\mathcal{E}}(Q)$ satisfies $q \leq \Theta_\delta(q)$ iff there is an algebra structure $\delta'\colon \Sigma Q' \to Q'$ turning q into an algebra homomorphism.*

The operator Θ_δ allows us to quotient the transition structure of the automaton. In order to obtain the minimal automaton, we incorporate the output map $o\colon Q \to O$ into the construction of a monotone operator based on Θ_δ. For technical convenience, we assume that this map is an element of $\mathsf{Quot}_{\mathcal{E}}(Q)$.[2] The relevant monotone operator for minimisation is $\Theta_\delta \wedge o$ (where the meet $\wedge$ is taken pointwise in $\mathsf{Quot}_{\mathcal{E}}(Q)$).

▶ **Example 4.11.** Let $\Sigma\colon \mathsf{Set} \to \mathsf{Set}$ be a polynomial functor induced by signature Γ. We first spell out what the cobase means concretely in this case and then study the operator Θ_δ in more detail. Since Σ is an endofunctor on Set, the cobase of a map $f\colon \Sigma X \to Y$ is the largest quotient $q \in \mathsf{Quot}_{\mathcal{E}}(X)$ such that for all $t, t' \in \Sigma X$:

$$\text{if } \Sigma q(t) = \Sigma q(t'), \text{ then } f(t) = f(t')\,.$$

[2] This is not a real restriction: one can just pre-process the automaton by factorising o, i.e., keeping only those outputs actually occurring in the automaton.

This means that for every $\gamma \in \Gamma$ with $k = \mathsf{ar}(\gamma)$, and any $x_1, \dots, x_k, y_1, \dots y_k$, we have that

$$\frac{q(x_1) = q(y_1) \qquad \dots \qquad q(x_k) = q(y_k)}{f(\kappa_\gamma(x_1, \dots, x_k)) = f(\kappa_\gamma(y_1, \dots, y_k))}$$

or equivalently that for all $x_1, \dots, x_k$ and x_i' with $1 \leq i \leq k$ we have

$$\frac{q(x_i) = q(x_i')}{f(\kappa_\gamma(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_k)) = f(\kappa_\gamma(x_1, \dots, x_{i-1}, x_i', x_{i+1}, \dots, x_k))}$$

Suppose (Q, δ, i, o) is an automaton. For $q \in \mathsf{Quot}(Q)$, we have $q \leq \Theta_\delta(q) \wedge o$ iff

- for all $x, x' \in Q$: if $q(x) = q(x')$, then $o(x) = o(x')$; and
- for all $\gamma \in \Gamma$ with $k = \mathsf{ar}(\gamma)$, and $x_1, \dots, x_k$ and x_i' with $1 \leq i \leq k$ we have

$$\frac{q(x_i) = q(x_i')}{q(\delta_\gamma(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_k)) = q(\delta_\gamma(x_1, \dots, x_{i-1}, x_i', x_{i+1}, \dots, x_k))}$$

A partition q with the above two properties is known as a *forward bisimulation* [25].

▶ **Theorem 4.12.** *Suppose Σ preserves $\mathcal{E}$-cointersections. Let (Q, δ, i, o) be an automaton, where $o \in \mathsf{Quot}_{\mathcal{E}}(Q)$. Then $\mathsf{gfp}(\Theta_\delta \wedge o)$ is the minimisation of (Q, δ, i, o).*

Proof. Denote the quotient $\mathsf{gfp}(\Theta_\delta \wedge o)$ by $q_m \colon Q \twoheadrightarrow Q_m$. Thus $q_m \leq \Theta_\delta(q_m)$ and $q_m \leq o$, hence (using Lemma 4.10) there exist δ_m, o_m turning q_m into an automaton homomorphism from (Q, δ, i, o) to $(Q_m, \delta_m, q_m \circ i, o_m)$. We show that this is the minimisation of (Q, δ, i, o).

To this end, let (Q', δ', i', o'), $q' \colon Q \twoheadrightarrow Q'$ be a quotient automaton of (Q, δ, i, o). By Lemma 4.10 we get $q' \leq \Theta_\delta(q')$, and since $o' \circ q' = o$ we have $q' \leq o$, hence $q' \leq \Theta_\delta(q') \wedge o$. Thus $q' \leq \mathsf{gfp}(\Theta_\delta \wedge o)$, i.e., there is a quotient $h \colon Q' \twoheadrightarrow Q_m$ such that $h \circ q' = q_m$. It only remains to show that h is a homomorphism of automata. First, since $q' \in \mathcal{E}$ and Σ preserves $\mathcal{E}$-cointersections, $\Sigma q'$ is an epimorphism. Combined with the fact that q' and q_m are algebra homomorphisms and that $h \circ q' = q_m$, it easily follows that h is an algebra homomorphism. To see that it preserves the output, we have $o_m \circ h \circ q' = o_m \circ q_m = o = o' \circ q'$; hence, since q' is epic, we get $o_m \circ h = o'$. For preservation of the input, we have $h \circ i' = h \circ q' \circ i = q_m \circ i$, where the first step holds because q' is a homomorphism of automata. ◀

The above characterisation of minimisation of an automaton (Q, δ, i, o) gives us two ways of constructing it by standard lattice-theoretic computations. First, via the Knaster-Tarski theorem, we obtain it as the join of all post-fixed points of $\Theta_\delta \wedge o$, which, by Lemma 4.10, amounts to the join of all quotient algebras respecting the output map o. That corresponds to the construction in [6]. Second, and perhaps most interestingly, we obtain the minimisation of (Q, δ, i, o) by iterating $\Theta_\delta \wedge o$, starting from the top element $\top$ of the lattice $\mathsf{Quot}_{\mathcal{E}}(Q)$. The latter construction is analogous to the classical partition refinement algorithm: Starting from $\top$ corresponds to identifying all states as equivalent (or in other words, starting from the coarsest equivalence class of states). Every iteration step of $\Theta_\delta \wedge o$ splits the states that can be distinguished successively by just outputs, trees of depth 1, trees of depth 2, etc. If the state space is finite, this construction terminates, yielding the minimisation of the original automaton by Theorem 4.12.

4.2 Simple automata

We defined an automaton to be minimal if it is reachable and satisfies a universal property w.r.t. reachable automata accepting the same language. It is also interesting to ask whether

there is another property that, together with reachability, implies minimality, but is not itself dependent on reachability [9]. Here we propose precisely such a condition.

▶ **Definition 4.13.** *An automaton (Q, δ, i, o) is called* simple *if for every quotient automaton (Q', δ', i', o') the associated quotient $q\colon Q \twoheadrightarrow Q'$ is an isomorphism.*

The result below asserts that minimal automata are precisely the automata that are simple and reachable. It can be seen as a refinement (and dual) of [10, Theorem 17], computing the reachable part of a coalgebra. One of the implications makes use of Theorem 4.12, so we assume that Σ preserves $\mathcal{E}$-cointersections.

▶ **Proposition 4.14.** *Suppose Σ preserves $\mathcal{E}$-cointersections. Let (Q, δ, i, o) be an automaton with $o \in \mathcal{E}$, and let (Q', δ', i', o'), $q\colon Q \twoheadrightarrow Q'$ be a quotient automaton. Then (Q', δ', i', o') is simple if and only if it is the minimisation of (Q, δ, i, o).*

Proof. By Theorem 4.12, the minimisation of (Q, δ, i, o) exists. We denote it by $q_m\colon Q \twoheadrightarrow Q_m$ and its associated automaton structure by $(Q_m, \delta_m, i_m, o_m)$.

Suppose (Q', δ', i', o') is simple. Since Q_m is the minimisation of Q and Q' is a quotient automaton of Q, there exists a homomorphism of automata $h\colon Q' \twoheadrightarrow Q_m$. Since Q' is simple, this homomorphism is an iso.

Conversely, suppose (Q', δ', i', o') is the minimisation of (Q, δ, i, o) and consider any quotient automaton Q'' of Q', witnessed by some $q'\colon Q' \twoheadrightarrow Q''$. Then Q'' is also a quotient automaton of Q, via $q' \circ q$. Because Q' is the minimisation of Q, there exists $k\colon Q'' \to Q'$ such that $k \circ q' \circ q = q$. Thus $k \circ q' = \mathsf{id}$, using that q is an epi. Since $q' \circ k \circ q' = q'$ and q' is an epi as well, we also have $q' \circ k = \mathsf{id}$. Hence q' is an iso, as needed. ◀

▶ **Corollary 4.15.** *If Σ preserves $\mathcal{E}$-cointersections, then an automaton $\mathcal{A} = (Q, \delta, i, o)$ with $o \in \mathcal{E}$ is minimal if and only if it is simple and reachable.*

Proof. First, suppose that $\mathcal{A}$ is minimal. By Lemma 4.5, we know that $\mathcal{A}$ is the minimisation (and, in particular, a quotient) of $(\Sigma^\diamond I, \alpha_I, \eta_I, \mathcal{L}(\mathcal{A}))$. In that case, $\mathcal{A}$ is reachable, and thus, since $o \in \mathcal{E}$, we have $\mathcal{L}(\mathcal{A}) \in \mathcal{E}$. By Proposition 4.14, we conclude that $\mathcal{A}$ is simple.

Conversely, let $\mathcal{A}$ be simple and reachable. By reachability, $\mathcal{A}$ is a quotient automaton of $(\Sigma^\diamond I, \alpha_I, \eta_I, \mathcal{L}(\mathcal{A}))$; also, $\mathcal{L}(\mathcal{A}) = o \circ \mathsf{rch} \in \mathcal{E}$. Proposition 4.14 then tells us that $\mathcal{A}$ is the minimisation of $(\Sigma^\diamond I, \alpha_I, \eta_I, \mathcal{L}(\mathcal{A}))$; by Lemma 4.5 we conclude that $\mathcal{A}$ is minimal. ◀

5 Nerode equivalence

We now show a generalised Nerode equivalence from which the minimal automaton can be constructed. Most of this section is based upon the work by Arbib and Manes [8], whose construction was further studied and refined by Anderson et al. [7] and Adámek and Trnková [6]. We make a significant improvement in generality by phrasing the central equivalence definition (Definition 5.5) in terms of an arbitrary monad, which unlike the previous cited work allows applications to algebras satisfying a fixed set of equations. A monad generalisation of the Myhill-Nerode theorem appears in [12], which confines itself to categories of sorted sets and does not characterise the equivalence as an object.

The abstract construction in this section does not require the varietor Σ. Instead, we focus on the monad $\Sigma^\diamond$ induced by its adjunction and generalise by fixing an arbitrary monad (T, η, μ) in C. Let $F \dashv U\colon \mathsf{C} \leftrightarrows \mathcal{EM}(T)$ be the adjunction with its category of (Eilenberg-Moore) algebras. Given a C-morphism $f\colon X \to UY$ for X in C and Y in $\mathcal{EM}(T)$, we write $f^\sharp\colon FX \to Y$ for its adjoint transpose. We can then use a generalised version of the automata defined in Section 3.

▶ **Definition 5.1** (T-automaton). *A* T-automaton *is a tuple* (Q, δ, i, o)*, where* (Q, δ) *is a* T*-algebra and* $i\colon I \to Q$ *and* $o\colon Q \to O$ *are morphisms in* C*. A* homomorphism *from* (Q, δ, i, o) *to* (Q', δ', i', o') *is a* T*-algebra homomorphism* $h\colon (Q, \delta) \to (Q', \delta')$ *such that* $h \circ i = i'$ *and* $o' \circ h = o$*.*

The reachability map of a T-automaton $\mathcal{A} = (Q, \delta, i, o)$ is given by $\mathsf{rch}_{\mathcal{A}} = U(i^\sharp)\colon TI \to Q$ and is therefore the unique T-algebra homomorphism $(TI, \mu) \to (Q, q)$ preserving initial states, taking $\eta_I\colon I \to TI$ to be the initial state selector of TI. The language of $\mathcal{A}$ is given by $\mathcal{L}(\mathcal{A}) = o \circ \mathsf{rch}_{\mathcal{A}}\colon TI \to O$.

The Σ-tree automata defined in Section 3 are recovered using the following fact: the category of Σ-algebras is isomorphic to $\mathcal{EM}(T)$ for T the free Σ-algebra monad $\Sigma^\diamond$.

▶ Remark 5.2. In Set, we may even add equations to the signature [36, Chapter VI.8, Theorem 1]. For Nom, this follows from the treatment of [34], giving a standard universal algebraic presentation of algebras over Nom. Indeed, results in this section apply to nominal tree automata, unless explicitly stated. For brevity we therefore focus on examples in Set.

▶ **Assumption 5.3.** In this section we will need the class $\mathcal{E}$ to be the reflexive regular epis.[3]

The next lemma will be used in proving our main theorems.

▶ **Lemma 5.4.** *Suppose* T *maps reflexive coequalisers to epimorphisms. If* $i\colon B \to UC$ *is such that* $U(i^\sharp)$ *reflexively coequalises* $q_1, q_2\colon A \to TB$ *in* C*, then* $i^\sharp$ *reflexively coequalises* $q_1^\sharp, q_2^\sharp\colon FA \to FB$*.*

Before defining an abstract Nerode equivalence, we recall the classical definition for languages of words. Given a language $L\colon A^* \to 2$, the equivalence $R \subseteq A^* \times A^*$ is defined as

$$R = \{(u, v) \in A^* \times A^* \mid \forall w \in A^*.\, L(uw) = L(vw)\}.$$

In this setting, $I = 1$ and $O = 2$. A function $Q \times A \to Q$ corresponds to an algebra for the monad $T = (-) \times A^*$, whose unit and multiplication are defined using the unit and multiplication of the monoid A^*. If $p_1, p_2\colon R \to A^* \cong 1 \times A^*$ are the projections, we note that R is defined to be the largest relation making the following diagram commute.

$$\begin{array}{ccccc} R \times A^* & & \xrightarrow{p_2 \times \mathsf{id}} & & 1 \times A^* \times A^* \\ & & & & \downarrow \mu \\ {\scriptstyle p_1 \times \mathsf{id}}\downarrow & & & & 1 \times A^* \\ & & & & \downarrow L \\ 1 \times A^* \times A^* & \xrightarrow{\mu} & 1 \times A^* & \xrightarrow{L} & 2 \end{array}$$

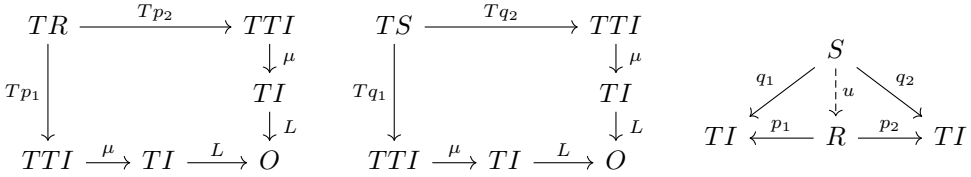
This leads to an abstract definition, using a limit[4] to generalise what it means to be maximal.

▶ **Definition 5.5** (Nerode equivalence). *Given a language* $L\colon TI \to O$ *and an object* R *with morphisms* $p_1, p_2\colon R \to TI$*, we say that* (R, p_1, p_2) *is the* Nerode equivalence *of* L *if the diagram below on the left commutes and for all objects* S *with a reflexive pair* $q_1, q_2\colon S \to TI$

[3] In a regular category, (reflexive regular epi, mono) forms a factorisation system in the same way (regular epi, mono) does, though one should note that the theory in this section does not actually need a factorisation system; the instantiation of $\mathcal{E}$ is only invoked to obtain the right notion of reachability.

[4] Note that it is not exactly a limit, as the defining property works with cones under T.

such that the diagram in the middle commutes there is a unique morphism $u\colon S \to R$ making the diagram on the right commute.



To show the versatility of our definition, we briefly explain a different example where the language is a set of words. This example cannot be recovered from the original definition by Arbib and Manes [8].

▶ **Example 5.6** (Syntactic congruence)**.** Let T be the free monoid or list monad $(-)^*$ so that $\mathcal{EM}(T)$ is the category of monoids, $I = A$, and $O = 2$. Given a language $L\colon A^* \to 2$, the Nerode equivalence as defined above is then the largest relation $R \subseteq A^* \times A^*$ such that

$$\frac{n \in \mathbb{N} \qquad (u_1, v_1), \ldots, (u_n, v_n) \in R}{L(u_1 \cdots u_n) = L(v_1 \cdots v_n)}.$$

Equivalently, R is the largest relation such that

$$\frac{(u, v) \in R \qquad w, x \in A^*}{L(wux) = L(wvx)},$$

which is precisely the *syntactic congruence* of the language.

We can show that the Nerode equivalence in Set exists, as long as the monad is finitary. To define it concretely, we use the following piece of notation. For any set X and $x \in X$, denote by $1_x\colon 1 \to X$ the constant x function, assuming no ambiguity of the set involved.

▶ **Proposition 5.7.** *For $\mathsf{C} = \mathsf{Set}$ and T any finitary monad, every language $L\colon TI \to O$ has a Nerode equivalence given by*

$$R = \{(u, v) \in TI \times TI \mid L \circ \mu \circ T[\mathsf{id}_{TI}, 1_u] = L \circ \mu \circ T[\mathsf{id}_{TI}, 1_v]\colon T(TI + 1) \to O\}$$

with the corresponding projections $p_1, p_2\colon R \to TI$.

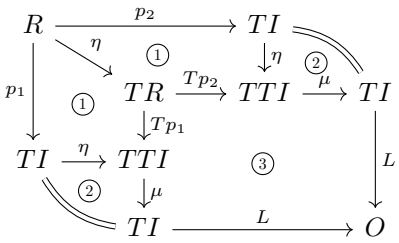
The definition of R above states that $u, v \in TI$ are related iff the elements of TI formed by putting either u or v in any *context* and then applying μ have the same value under L. A context is an element of $T(TI + 1)$, where $1 = \{\Box\}$ denotes a *hole* where either u or v can be plugged in. In the tree automata literature, such contexts, although restricted to contain a single instance of $\Box$, are used in algorithms for minimisation [25] and learning [44, 19]. Unfortunately, the characterisation of Proposition 5.7 does not directly extend to Nom, because the functions 1_x are not, in general, equivariant. We leave this for future work.

Below we show that, under a few mild assumptions, the abstract equivalence is in fact a congruence: it induces a T-automaton, which moreover is minimal. Intuitively, given a language $L\colon TI \to O$ that has a Nerode equivalence, we use the equivalence to quotient the T-automaton (FI, η, L). We first need a technical lemma.

▶ **Lemma 5.8.** *If C has coproducts, then for any Nerode equivalence (R, p_1, p_2) there exists a unique T-algebra structure $u\colon TR \to R$ making p_1 and p_2 T-algebra homomorphisms $(R, u) \to (TI, \mu)$ that have a common section.*

▶ **Theorem 5.9.** *If* C *has coproducts and reflexive coequalisers and* T *preserves reflexive coequalisers, then for every language that has a Nerode equivalence there exists a minimal* T*-automaton accepting it.*

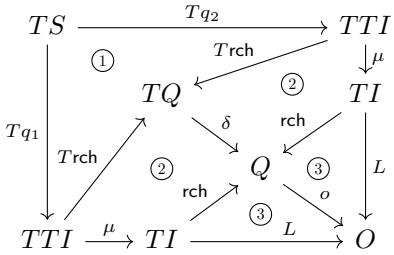
Proof. Let $L\colon TI \to O$ be the language with Nerode equivalence (R, p_1, p_2) and $c\colon T \to M$ the coequaliser of p_1 and p_2 in C. By Lemma 5.8 there exists a T-algebra structure on R making p_1 and p_2 T-algebra homomorphisms into (TI, μ) that have a common section. Since T preserves reflexive coequalisers, they are lifted by U, and we have a morphism $m\colon TM \to M$ making (M, m) a T-algebra such that c is a T-algebra homomorphism $(TI, \mu) \to (M, m)$. Since the diagram below on the left commutes and c coequalises p_1 and p_2, there is a unique morphism o_M rendering the diagram on the right commutative.



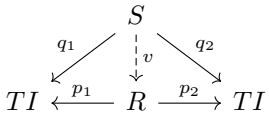
(1)

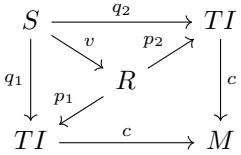
Choosing $i_M = c \circ \eta\colon I \to M$, we obtain a T-automaton $\mathcal{M} = (M, m, i_M, o_M)$. Note that $U(i_M^\sharp) = c$, so c is the reachability map of $\mathcal{M}$. Hence, we find that $\mathcal{L}(\mathcal{M}) = L$ by (1). The morphism c coequalises the reflexive pair (p_1, p_2) by definition, so $\mathcal{M}$ is reachable.

To see that $\mathcal{M}$ is minimal, consider any reachable T-automaton $\mathcal{A} = (Q, \delta, i, o)$ such that $\mathcal{L}(\mathcal{A}) = L$. Reachability amounts to the reachability map $\mathsf{rch}\colon TI \to UQ$ being the reflexive coequaliser of a pair of morphisms $q_1, q_2\colon S \to TI$. From commutativity of



we obtain by the Nerode equivalence property a unique morphism $v\colon S \to R$ making the diagram below on the left commute.



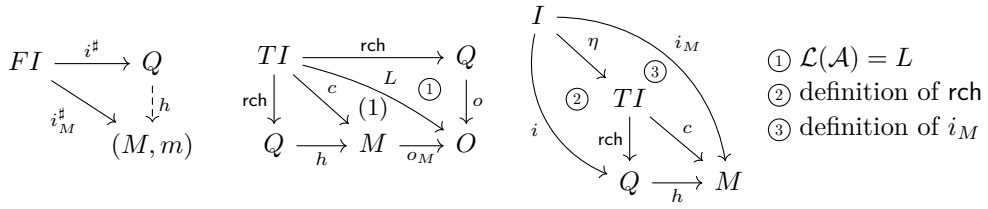


Extending this with c, the coequaliser of p_1 and p_2, gives the commutative diagram on the right. Recall that $U(i_M^\sharp) = c$. We now find

$$i_M^\sharp \circ q_1^\sharp = (U(i_M^\sharp) \circ q_1)^\sharp = (c \circ q_1)^\sharp = (c \circ q_2)^\sharp = (U(i_M^\sharp) \circ q_2)^\sharp = i_M^\sharp \circ q_2^\sharp.$$

Here the first and last equality apply a general naturality property of the adjunction. Since $\mathsf{rch} = U(i^\sharp)$ is the reflexive coequaliser of q_1 and q_2, $i^\sharp$ is the reflexive coequaliser of $q_1^\sharp$ and

$q_2^\sharp$ by Lemma 5.4. We then obtain a unique T-algebra homomorphism $h\colon (Q,\delta) \to (M,m)$ making the diagram below on the left commute.



From commutativity of the other diagrams we find $o_M \circ h = o$ (using that rch is epi) and $h \circ i = i_M$. Thus, h is a T-automaton homomorphism $\mathcal{A} \to \mathcal{M}$. To see that it is unique, note that any T-automaton homomorphism $h'\colon \mathcal{A} \to \mathcal{M}$ is a T-algebra homomorphism $(Q,\delta) \to (M,m)$ such that $h' \circ i = i_M$. It is then not hard to see that $h' \circ i^\sharp = (h' \circ i)^\sharp = i_M^\sharp$. We conclude that $h' = h$ by the uniqueness property of h satisfying $h \circ i^\sharp = i_M^\sharp$. ◀

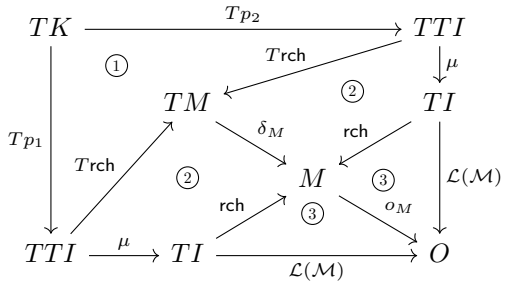
▶ Remark 5.10. We briefly discuss the conditions of the above theorem in the specific case of $\mathsf{C} = \mathsf{Set}$ with T a finitary monad. This includes the setting of tree automata in Set, as a monad on Set is finitary if and only if $\mathcal{EM}(T)$ is equivalent to the category of algebras for a signature modulo equations. Proposition 5.7 shows that all Nerode equivalences exist here. Furthermore, Lack and Rosický [35] observe that an endofunctor on Set is finitary if and only if it preserves sifted colimits, of which reflexive coequalisers form an instance.

To conclude this section we show that the converse of the previous theorem also holds, using the existence of kernel pairs rather than coproducts. We need a technical lemma first.

▶ **Lemma 5.11.** *If $q_1, q_2\colon A \to TB$ is a reflexive pair in C, then so is $(q_1^\sharp, q_2^\sharp)$ in $\mathcal{EM}(T)$.*

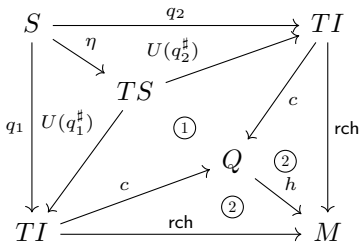
▶ **Theorem 5.12.** *If C has kernel pairs and reflexive coequalisers and T preserves reflexive coequalisers, then every language that has a minimal T-automaton has a Nerode equivalence.*

Proof. Let $\mathcal{M} = (M, \delta_M, i_M, o_M)$ be a minimal T-automaton and $p_1, p_2\colon K \to TI$ the kernel pair of its reachability map $\mathsf{rch}\colon FI \to M$. We claim that K together with p_1 and p_2 forms the Nerode equivalence of $\mathcal{L}(\mathcal{M})$. To see this, note that the diagram below on the left commutes.

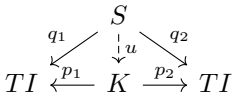


Now if S with $q_1, q_2\colon S \to TI$ is any reflexive pair making the diagram on the right commute, we let $c\colon TI \to Q$ be the coequaliser of $U(q_1^\sharp)$ and $U(q_2^\sharp)$, noting that this is a reflexive pair by Lemma 5.11. Then since T preserves reflexive coequalisers, they are lifted by U, meaning that there exists a unique T-algebra structure $\delta\colon TQ \to Q$ making $c\colon FI \to (Q,\delta)$ a T-algebra homomorphism that is the coequaliser of $q_1^\sharp$ and $q_2^\sharp$. We also have $\mathcal{L}(\mathcal{M}) \circ U(q_1^\sharp) = \mathcal{L}(\mathcal{M}) \circ U(q_2^\sharp)$ by commutativity of the diagram on the right, so with c coequalising $U(q_1^\sharp)$ and $U(q_2^\sharp)$ there is a unique morphism $o\colon Q \to O$ such that $o \circ c = \mathcal{L}(\mathcal{M})$. Setting $i = c \circ \eta_I$, we have a

T-automaton (Q, δ, i, o) with reachability map $U(i^\sharp) = c$ that accepts the language $\mathcal{L}(\mathcal{M})$. By $\mathcal{M}$ being minimal there exists a unique T-automaton homomorphism $h\colon (Q, \delta, i, o) \to \mathcal{M}$. Then the diagram below on the left commutes.



① c coequalises $U(q_1^\sharp)$ and $U(q_2^\sharp)$
② uniqueness of reachability maps



By p_1 and p_2 being the kernel pair of rch there exists a unique morphism $u\colon S \to K$ making the diagram on the right commute. ◀

6 Tree automata with side-effects

We extend tree automata with various side-effects, covering as examples non-deterministic, weighted, and non-deterministic nominal automata. The key insight is to view them as algebras in the *Kleisli category of a monad* S. We first recall some basic notions.

Kleisli category Every monad (S, η, μ) has an associated *Kleisli category* $\mathcal{K}\ell(S)$, whose objects are those of C and whose morphisms $X \kto Y$ are morphisms $X \to SX$ in C. Given such morphisms $f\colon X \kto Y$ and $g\colon Y \kto Z$, their (Kleisli) composition $g \circ f$ is defined as $\mu_Z \circ Sg \circ f$ in C. The *Kleisli adjunction* $J \dashv V : \mathsf{C} \leftrightarrows \mathcal{K}\ell(S)$ is given by $JX = X$, $J(f\colon X \to Y) = \eta_Y \circ f$ and $VY = SY$, $V(f\colon X \kto Y) = \mu_Y \circ Sf$.

▶ **Example 6.1.** The category $\mathcal{K}\ell(\mathcal{P}_\mathsf{f})$ has morphisms $X \to \mathcal{P}_\mathsf{f} Y$, which are finitely-branching relations, and Kleisli composition is relational composition. We have that J maps a function to its graph, and V maps X to $\mathcal{P}_\mathsf{f} X$ and a relation $R\colon X \kto Y$ to the function

$$\lambda U \subseteq X.\{y \mid \exists x \in U : (x, y) \in R\}.$$

The category $\mathcal{K}\ell(\mathcal{M}_\mathbb{F})$ has morphisms $X \to \mathcal{M}_\mathbb{F} Y$ that are matrices over $\mathbb{F}$ indexed by X and Y (equivalently, linear maps between the corresponding free vector spaces), and composition is matrix multiplication. The left adjoint J maps a function $f\colon X \to Y$ to the matrix $Jf[x, f(x)] = 1$, for $x \in X$, and 0 elsewhere, and the right adjoint V maps a matrix $X \kto Y$ to the corresponding linear function $\mathcal{M}_\mathbb{F} X \to \mathcal{M}_\mathbb{F} Y$.

Given an endofunctor Σ on C, a functor $\widehat{\Sigma}\colon \mathcal{K}\ell(S) \to \mathcal{K}\ell(S)$ is an *extension* of Σ if the following diagram commutes:

$$\begin{array}{ccc} \mathsf{C} & \xrightarrow{\Sigma} & \mathsf{C} \\ {\scriptstyle J}\downarrow & & \downarrow{\scriptstyle J} \\ \mathcal{K}\ell(S) & \xrightarrow{\widehat{\Sigma}} & \mathcal{K}\ell(S) \end{array}$$

Extensions are in bijective correspondence with *distributive laws* $\lambda\colon \Sigma S \Rightarrow S\Sigma$ [39], which are natural transformations satisfying certain axioms. Explicitly, we have $\widehat{\Sigma} X = \Sigma X$ and $f\colon X \kto Y$ (a morphism $X \to SY$ in C) is mapped to $\lambda_Y \circ \Sigma f\colon \Sigma X \to S\Sigma Y$, seen in $\mathcal{K}\ell(S)$.

In [23, Lemma 2.4] it is shown that a canonical distributive law in Set always exists in case Σ is polynomial and S is a commutative monad [31].

▶ **Example 6.2.** For Σ a polynomial Set endofunctor, the canonical distributive law $\lambda\colon \Sigma\mathcal{P}_\mathsf{f} \Rightarrow \mathcal{P}_\mathsf{f}\Sigma$ can be directly defined as follows: $\lambda_X(u) = \{v \in \Sigma X \mid (v,u) \in \mathsf{img}(\langle \Sigma p_1, \Sigma p_2\rangle)\}$, where p_1 and p_2 are the left and right projections of the membership relation $\in_X \subseteq X \times \mathcal{P}_\mathsf{f}X$.

The multiplicity monad $(\mathcal{M}_\mathbb{F}, e, m)$ admits a distributive law $\lambda\colon \Sigma\mathcal{M}_\mathbb{F} \Rightarrow \mathcal{M}_\mathbb{F}\Sigma$, inductively defined as follows, where $\otimes$ is the Kronecker product:

$$\lambda^{\mathsf{id}} = \mathsf{id}_{\mathcal{M}_\mathbb{F}} \qquad \lambda^{A} = e_A \qquad \lambda^{\Sigma_1\times\Sigma_2} = \otimes \circ (\lambda^{\Sigma_1} \times \lambda^{\Sigma_2}) \qquad \lambda^{\coprod_{i\in I}\Sigma_i} = [\mathcal{M}_\mathbb{F}(\kappa_i) \circ \lambda^{\Sigma_i}]_{i\in I}$$

We can now define our notion of tree automaton with side-effects.

▶ **Definition 6.3** ((Σ, S)-tree automaton)**.** *Given a monad S, and $\widehat{\Sigma}\colon \mathcal{K}\ell(S) \to \mathcal{K}\ell(S)$ extending a functor Σ on C, a* (Σ, S)-tree automaton *is a $\widehat{\Sigma}$-tree automaton, i.e., a tuple (Q, δ, i, o), where (Q, δ) is a $\widehat{\Sigma}$-algebra and $i\colon I \nrightarrow Q$ and $o\colon Q \nrightarrow O$ are morphisms in $\mathcal{K}\ell(S)$.*

▶ **Example 6.4.**

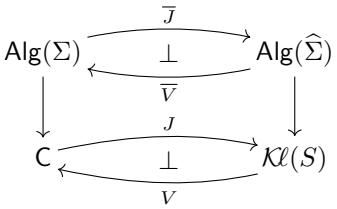
1. Let Γ be a signature, and let Σ be its signature functor. Then the extension of Σ to $\mathcal{K}\ell(\mathcal{P}_\mathsf{f})$ is obtained via the distributive law of Example 6.2, namely $\widehat{\Sigma}X = \Sigma X$ and
$$\widehat{\Sigma}(f\colon X \nrightarrow Y)(\kappa_\gamma(x_1,\dots,x_{\mathsf{ar}(\gamma)})) = \{(\kappa_\gamma(y_1,\dots,y_{\mathsf{ar}(\gamma)})) \mid y_j \in f(x_j) \text{ for } 1 \le j \le \mathsf{ar}(\gamma)\}$$
for $\gamma \in \Gamma$. Non-deterministic tree automata are $(\Sigma, \mathcal{P})$-tree automata (Q, δ, i, o) with $O = 1$, the singleton set. In fact, we have that $\delta\colon \coprod_{\gamma\in\Gamma} Q^{\mathsf{ar}(\gamma)} \nrightarrow Q$ is a family of relations $\delta_\gamma \subseteq Q^{\mathsf{ar}(\gamma)} \times Q$; by the same token, $i \subseteq I \times Q$ relates the frontier alphabet with (possibly several) states, and $o \subseteq Q \times 1 \cong Q$ is the set of final states.
2. Let Γ be a signature, and let Σ be its signature functor. The extension of Σ to $\mathcal{K}\ell(\mathcal{M}_\mathbb{F})$ is obtained via the distributive law of Example 6.2. Explicitly, $\widehat{\Sigma}X = \Sigma X$ and $\widehat{\Sigma}(f\colon X \nrightarrow Y)$ maps $\kappa_\gamma(x_1,\dots,x_{\mathsf{ar}(\gamma)})$ to the vector $[\kappa_\gamma(v_z)]_{z\in Y^{\mathsf{ar}(\gamma)}}$ such that v_z is the z-th component of $f(x_1) \otimes \cdots \otimes f(x_{\mathsf{ar}(\gamma)})$, for $\gamma \in \Gamma$. A multiplicity tree automaton [29] is a $(\Sigma, \mathcal{M}_\mathbb{F})$-tree automaton (Q, δ, i, o) with $I = O = 1$. In fact, we have that δ_γ is the *transition matrix* $Q^{\mathsf{ar}(\gamma)} \nrightarrow Q$ in $\mathbb{F}^{|Q|^{\mathsf{ar}(\gamma)}\times|Q|}$; similarly, $i\colon 1 \nrightarrow Q$ is the *initial weight vector* in $\mathbb{F}^{1\times|Q|}$, and $o\colon Q \nrightarrow 1$ is the *final weight vector* in $\mathbb{F}^{|Q|\times 1}$. Intuitively, δ_γ maps an $\mathsf{ar}(\gamma)$-tuple of elements of Q to a linear combination over Q. We note that we can go beyond fields and consider $(\Sigma, \mathcal{M}_\mathbb{S})$-tree automata for a semiring $\mathbb{S}$, encompassing *weighted* tree automata [17].
3. The nondeterministic version of the automata defined in Section 3.1 can be obtained via the monad $\mathcal{P}_\omega\colon \mathsf{Nom} \to \mathsf{Nom}$, mapping a nominal set to the nominal set of its finitely-supported, orbit-finite subsets.[5] This is analogous to non-deterministic nominal automata [13]. We note that $\mathcal{K}\ell(\mathcal{P}_\omega)$ is precisely the category of nominal sets and (orbit-finitely branching) equivariant relations and that Σ_λ extends to relations just as a set endofunctor—the distributive law is defined as the one for $\mathcal{P}_\mathsf{f}$ of Example 6.2, where all the maps are equivariant. A nondeterministic nominal Σ_λ-tree automaton is a $(\Sigma_\lambda, \mathcal{P}_\omega)$-tree automaton (Q, δ, i, o) with $O = 1$, the one-element nominal set with trivial group action. We have that i and the components of δ are equivariant relations. For instance, $\delta_{\mathsf{lambda}} \subseteq [\mathbb{A}]Q \times Q$, and o is an equivariant subset of Q.

We now study language semantics of (Σ, S)-tree automata. Languages are defined via free algebras (see Section 3). It turns out that the free algebras in $\mathsf{Alg}(\Sigma)$ and $\mathsf{Alg}(\widehat{\Sigma})$ are closely related. To see this, we use the following result, which follows from [24, Theorem 2.14].

[5] This is the finitary version of the powerset functor in Nom.

▶ **Lemma 6.5.** *Let $\Sigma\colon \mathsf{C} \to \mathsf{C}$ be a functor, $S\colon \mathsf{C} \to \mathsf{C}$ a monad, and $\widehat{\Sigma}\colon \mathcal{K}\ell(S) \to \mathcal{K}\ell(S)$ an extension of Σ. Let $\lambda\colon \Sigma S \Rightarrow S\Sigma$ be the corresponding distributive law.*

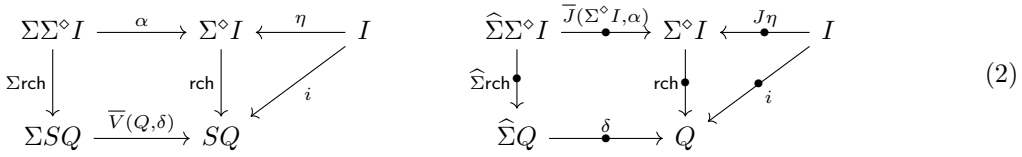
Then the Kleisli adjunction $J \dashv V : \mathsf{C} \leftrightarrows \mathcal{K}\ell(S)$ lifts to an adjunction in: $\overline{J} \dashv \overline{V}$



$$\overline{J}(Q, \delta\colon \Sigma Q \to Q) = (Q, J(\delta)\colon \widehat{\Sigma} Q \nrightarrow Q)$$

$$\overline{V}(Q, \gamma\colon \widehat{\Sigma} Q \nrightarrow Q) = (SQ, V(\gamma) \circ \lambda_Q\colon \Sigma SQ \to SQ)$$

From Lemma 6.5, and using that free algebras can be obtained as colimits of transfinite sequences [2, 28], it follows that any free Σ-algebra $(\Sigma^\diamond X, \alpha_X)$ is mapped by the left adjoint $\overline{J}$ to a free $\widehat{\Sigma}$-algebra with the same carrier. Concretely, given a $\widehat{\Sigma}$-algebra (Q, δ) and a morphism $i\colon I \nrightarrow Q$, a (unique) morphism $\mathsf{rch}\colon \Sigma^\diamond I \to SQ$ makes the diagram on the left commute in C iff it makes the diagram on the right commute in $\mathcal{K}\ell(S)$:



(2)

Note that the adjoint transpose of rch is the same morphism, seen in $\mathcal{K}\ell(S)$. The functor $\overline{V}$ can be viewed as a *determinisation* construction for (Σ, S)-tree automata.

▶ **Definition 6.6.** *Given an (Σ, S)-tree automaton (Q, δ, i, o), let $\overline{\delta}\colon \Sigma S(Q) \to S(Q)$ be the algebra structure of $\overline{V}(Q, \delta)$, i.e., $(S(Q), \overline{\delta}) = \overline{V}(Q, \delta)$, and $\overline{o} = V(o)$. The Σ-tree automaton $(SQ, \overline{\delta}, i, \overline{o})$ is called the* determinisation *of (Q, δ, i, o).*

The following shows correctness of this determinisation construction, using the correspondence in (2), and provides a concrete description of the language semantics of (Σ, S)-tree automata.

▶ **Corollary 6.7.** *Let (Q, δ, i, o) be a (Σ, S)-tree automaton. Then $\mathcal{L}(SQ, \overline{\delta}, i, \overline{o}) = \mathcal{L}(Q, \delta, i, o)$.*

We conclude this section with some example instantiations of determinisation, and of how they can be used to compute languages.

▶ **Example 6.8.**

1. The determinisation of a $(\Sigma, \mathcal{P}_\mathsf{f})$-tree automaton (Q, δ, i, o) is

$$\overline{\delta}_\gamma(X_1, \dots, X_k) = \bigcup_{x_1 \in X_1, \dots, x_k \in X_k} \delta_\gamma(x_1, \dots, x_k) \qquad \overline{o}(X) = \bigcup_{x \in X} o(x)$$

for $\gamma \in \Gamma$ with $k = \mathsf{ar}(\gamma)$, and $X_1, \dots, X_k, X$ finite subsets of Q. This definition precisely corresponds to the usual determinisation of bottom-up tree automata (see e.g. [22]). The reachability function is then given by

$$\mathsf{rch}(t) = \begin{cases} i(t) & \text{if } t \in I \\ \bigcup_{\substack{x_1 \in \mathsf{rch}(t_1) \\ \dots \\ x_k \in \mathsf{rch}(t_k)}} \delta_\gamma(x_1, \dots, x_k) & \text{if } t = (\gamma, x_1, \dots, x_k), k = \mathsf{ar}(\gamma). \end{cases}$$

Using Corollary 6.7, we have that the language of (Q, δ, i, o) is

$$\mathcal{L}(Q, \delta, i, o)(t) = \bigcup_{s \in \mathsf{rch}(t)} o(s).$$

That is: a tree t is accepted by (Q, δ, i, o) whenever there is a final state among those reached by parsing t.

2. The determinisation of a $(\Sigma, \mathcal{M}_{\mathbb{F}})$-tree automaton is given by

$$\overline{\delta}_\gamma(\varphi_1, \dots, \varphi_k) = (\varphi_1 \otimes \cdots \otimes \varphi_k) \bullet \delta_\gamma \qquad \overline{o}(\varphi) = \varphi \bullet o$$

where $\gamma \in \Gamma$ and $\mathsf{ar}(\gamma) = k$; also, $\otimes$ is the Kronecker product and $\bullet$ is matrix multiplication. Explicitly, $\overline{\delta}_\gamma$ takes a k-tuple of vectors over Q and turns it into a vector φ over k-tuples of states via the distributive law (defined as the Kronecker product, see Example 6.2), i.e., $\varphi(q_1, \dots, q_k) = \varphi_1(q_1) \cdots \varphi_k(q_k)$, for $q_1, \dots, q_k \in Q$. The result is then multiplied by the matrix δ_γ to compute the successor vector.
Similarly, the reachability function becomes

$$\mathsf{rch}(t) = \begin{cases} i(t) & \text{if } t \in I \\ (\mathsf{rch}(t_1) \otimes \cdots \otimes \mathsf{rch}(t_k)) \bullet \delta_\gamma & \text{if } t = (\gamma, t_1, \dots, t_k), k = \mathsf{ar}(\gamma). \end{cases}$$

Hence we obtain the language $\mathcal{L}(Q, \delta, i, o)(t) = \mathsf{rch}(t) \bullet o$, which corresponds to the language semantics given in [29].

3. The case of $(\Sigma_\lambda, \mathcal{P}_\omega)$-tree automata is completely analogous to point 1. For instance,

$$\overline{\delta}_{\mathsf{lambda}}(X) = \bigcup_{x \in X} \delta_{\mathsf{lambda}}(x)$$

for X a finitely supported orbit-finite subset of $[\mathbb{A}]Q$.

7 Future work

The algorithmic side of the iterative minimisation construction presented in Section 4 is left open. For classical tree automata there exist sophisticated variants of partition refinement [25, 1], akin to Hopcroft's classical algorithm. A generalisation to the current algebraic setting is an interesting direction of research, for which a natural starting point would be to try and integrate in our setting the efficient coalgebraic algorithm presented in [18].

Further, we characterised the minimal automaton as the greatest fixed point of a monotone function, recovering the notion of forward bisimulations as its post-fixed points (although it is perhaps more natural to think of these as congruences). This characterisation suggests an integration with up-to techniques [42, 14, 15], which have, to the best of our knowledge, not been applied to tree automata. In particular, we are interested in applying these algorithms to decide equivalence of series-parallel rational and series-rational expressions [38].

Since completeness of Kleene Algebra is connected to minimality of deterministic finite automata [33], we wonder whether a completeness proof can be recovered using automata as presented in this paper. In particular, our abstract framework might allow us to transpose such a proof to settings such as Bi-Kleene Algebra [37] or Concurrent Kleene Algebra [27].

References

1 P. A. Abdulla, J. Högberg, and L. Kaati. Bisimulation minimization of tree automata. *Int. J. Found. Comput. Sci.*, 18(4):699–713, 2007.

2 J. Adámek. Free algebras and automata realizations in the language of categories. *Commentationes Mathematicae Universitatis Carolinae*, 15(4):589–602, 1974.

3 J. Adámek. Realization theory for automata in categories. *J. Pure and Appl. Algebra*, 9(2):281 – 296, 1977.
4 J. Adámek, F. Bonchi, M. Hülsbusch, B. König, S. Milius, and A. Silva. A coalgebraic perspective on minimization and determinization. In *FoSSaCS*, pages 58–73, 2012.
5 J. Adámek, H. Herrlich, and G. Strecker. *Abstract and concrete categories: The joy of cats*, volume 17 of *Reprints in Theory and Applications of Categories*. TAC, 2006.
6 J. Adámek and V. Trnková. *Automata and algebras in categories*. Kluwer, 1989.
7 B. D. Anderson, M. A. Arbib, and E. G. Manes. *Foundations of system theory: finitary and infinitary conditions*, volume 115 of *Lecture Notes in Econ. and Math. Syst.* Springer, 1976.
8 M. A. Arbib and E. G. Manes. Machines in a category: An expository introduction. *SIAM review*, 16(2):163–192, 1974.
9 M. A. Arbib and E. G. Manes. Adjoint machines, state-behavior machines, and duality. *Journal of Pure and Applied Algebra*, 6(3):313–344, 1975.
10 S. Barlocco, C. Kupke, and J. Rot. Coalgebra learning via duality. In *FoSSaCS*, pages 62–79, 2019.
11 A. Blok. Interaction, observation and denotation. Master's thesis, ILLC Amsterdam, 2012.
12 M. Bojańczyk. Recognisable languages over monads. In *DLT*, pages 1–13. Springer, 2015.
13 M. Bojańczyk, B. Klin, and S. Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10(3), 2014.
14 F. Bonchi, D. Petrisan, D. Pous, and J. Rot. A general account of coinduction up-to. *Acta Inf.*, 54(2):127–190, 2017.
15 F. Bonchi and D. Pous. Hacking nondeterminism with induction and coinduction. *Commun. ACM*, 58(2):87–95, 2015.
16 F. Borceux. *Handbook of Categorical Algebra*, volume 1 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1994.
17 B. Borchardt and H. Vogler. Determinization of finite state weighted tree automata. *Journal of Automata, Languages and Combinatorics*, 8(3):417–463, 2003.
18 U. Dorsch, S. Milius, L. Schröder, and T. Wißmann. Efficient coalgebraic partition refinement. In *CONCUR*, pages 32:1–32:16, 2017.
19 F. Drewes and J. Högberg. Query learning of regular tree languages: How to avoid dead states. *Theory of Computing Systems*, 40(2):163–185, 2007.
20 M. P. Fiore. Discrete generalised polynomial functors (extended abstract). In *ICALP*, pages 214–226, 2012.
21 M. J. Gabbay and A. Mathijssen. Nominal (universal) algebra: Equational logic with names and binding. *J. Log. Comput.*, 19(6):1455–1508, 2009.
22 A. Habrard and J. Oncina. Learning multiplicity tree automata. In *ICGI*, volume 4201 of *Lecture Notes in Computer Science*, pages 268–280. Springer, 2006.
23 I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3(4), 2007.
24 C. Hermida and B. Jacobs. Structural induction and coinduction in a fibrational setting. *Inf. Comput.*, 145(2):107–152, 1998.
25 J. Högberg, A. Maletti, and J. May. Backward and forward bisimulation minimization of tree automata. *Theoretical Computer Science*, 410(37):3539–3552, 2009.
26 W. M. Holcombe. *Algebraic Automata Theory*. Cambridge University Press, 1982.
27 T. Kappé, P. Brunet, A. Silva, and F. Zanasi. Concurrent Kleene algebra: Free model and completeness. In *ESOP*, pages 856–882, 2018.
28 G. M. Kelly. A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on. *Bull. Austr. Math. Soc.*, 22(1):1–83, 1980.
29 S. Kiefer, I. Marusic, and J. Worrell. Minimisation of multiplicity tree automata. *Logical Methods in Computer Science*, 13(1), 2017.
30 B. Klin and J. Rot. Coalgebraic trace semantics via forgetful logics. *Logical Methods in Computer Science*, 12(4), 2016.

31 A. Kock. Monads on symmetric monoidal closed categories. *Arch. der Math.*, 21(1):1–10, 1970.
32 B. König and S. Küpper. Generic partition refinement algorithms for coalgebras and an instantiation to weighted automata. In *Theory Comput. Syst.*, pages 311–325, 2014.
33 D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Inf. Comput.*, 110(2):366–390, 1994.
34 A. Kurz and D. Petrisan. On universal algebra over nominal sets. *Mathematical Structures in Computer Science*, 20(2):285–318, 2010.
35 S. Lack and J. Rosickỳ. Notions of Lawvere theory. *Appl. Categ. Struct.*, 19(1):363–391, 2011.
36 S. M. Lane. *Categories for the working mathematician*, volume 5. Springer, 2013.
37 M. R. Laurence and G. Struth. Completeness theorems for bi-Kleene algebras and series-parallel rational pomset languages. In *RAMiCS*, pages 65–82, 2014.
38 K. Lodaya and P. Weil. Series-parallel languages and the bounded-width property. *Theoretical Computer Science*, 237(1):347–380, 2000.
39 P. S. Mulry. Lifting theorems for Kleisli categories. In *MFPS*, pages 304–319, 1993.
40 J. E. Pin. Mathematical foundations of automata theory. Version of March 13, 2019.
41 A. M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press, 2013.
42 D. Pous and D. Sangiorgi. *Advanced Topics in Bisimulation and Coinduction*, chapter Enhancements of the coinductive proof method. Cambridge University Press, 2011.
43 J. J. M. M. Rutten. Automata and coinduction (an exercise in coalgebra). In *CONCUR*, pages 194–218, 1998.
44 Y. Sakakibara. Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science*, 76(2-3):223–242, 1990.
45 A. Silva, F. Bonchi, M. M. Bonsangue, and J. J. M. M. Rutten. Generalizing determinization from automata to coalgebras. *Logical Methods in Computer Science*, 9(1), 2013.
46 T. Wißmann, S. Milius, S. Katsumata, and J. Dubut. A coalgebraic view on reachability. *arXiv e-prints*, Jan 2019. `https://arxiv.org/abs/1901.10717`.

A Proofs for Section 5

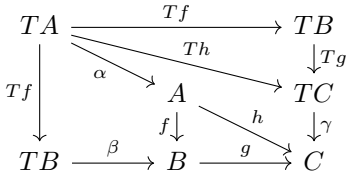
In the proofs below, we will use the following basic adjunction properties, in particular for the adjunction $F \dashv U \colon \mathsf{C} \leftrightarrows \mathcal{EM}(T)$ with adjoint transpose $(-)^\sharp$:

- The transpose $f^\sharp \colon FA \to B$ for $f \colon A \to UB$ in C can be defined as $f^\sharp = y \circ Tf$, where y is the T-algebra structure on Y.
- For all $f \colon X \to UY$ and $g \colon UY \to UZ$ in C we have $U(g^\sharp) \circ Tf = U((g \circ f)^\sharp)$.
- We have $U(\eta^\sharp_X) = \mathsf{id}_{TX}$.

We need the following additional lemmas in the proofs below.

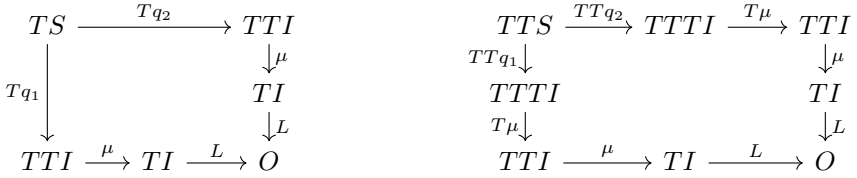
▶ **Lemma A.1.** *If $f \colon A \to B$ and $h \colon A \to C$ in $\mathcal{EM}(T)$ are such that there exists $g \colon UB \to UC$ in C with $g \circ f = h$ and Tf is an epi, then g is a T-algebra homomorphism $B \to C$.*

Proof. Let $\alpha \colon TA \to A$, $\beta \colon TB \to B$, and $\gamma \colon TC \to C$ be the respective T-algebra structures on A, B, and C. By commutativity of



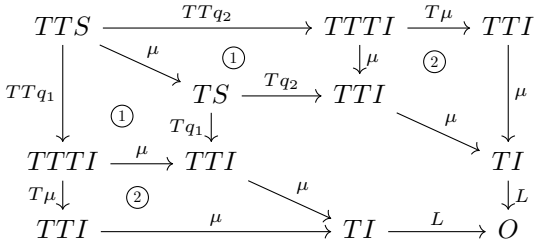
and Tf being an epi, we directly conclude that g is a T-algebra homomorphism $B \to C$. ◀

▶ **Lemma A.2.** *Given a language L and $q_1, q_2 \colon S \to TI$ making the diagram below on the left commute, the diagram on the right commutes.*



Furthermore, if (q_1, q_2) is a reflexive pair, then so is $(\mu_I \circ Tq_1, \mu_I \circ Tq_2)$.

Proof. We extend the assumption to the following commutative diagram.



① naturality of μ
② monad law

As for reflexivity, $(\mu_I \circ Tq_1, \mu_I \circ Tq_2)$ is the composition of the reflexive pairs (μ_I, μ_I) and (Tq_1, Tq_2). ◀

▶ **Lemma 5.4.** *Suppose T maps reflexive coequalisers to epimorphisms. If $i \colon B \to UC$ is such that $U(i^\sharp)$ reflexively coequalises $q_1, q_2 \colon A \to TB$ in C, then $i^\sharp$ reflexively coequalises $q_1^\sharp, q_2^\sharp \colon FA \to FB$.*

Proof. For $k \in \{1,2\}$ we have $U(i^\sharp \circ q_k^\sharp) \circ \eta_A = \mathsf{rch} \circ U(q_k^\sharp) \circ \eta_A = \mathsf{rch} \circ q_k$, so by $U(i^\sharp)$ coequalising q_1 and q_2 we have $i^\sharp \circ q_1^\sharp = i^\sharp \circ q_2^\sharp$. If a T-algebra homomorphism $f\colon TB \to Z$ is such that $f \circ q_1^\sharp = f \circ q_2^\sharp$, then

$$Uf \circ q_1 = Uf \circ U(q_1^\sharp) \circ \eta_A = Uf \circ U(q_2^\sharp) \circ \eta_A = Uf \circ q_2,$$

which because $U(i^\sharp)$ coequalises q_1 and q_2 yields a unique function $u\colon UC \to UZ$ such that $u \circ \mathsf{rch} = f$. Remains to show that u is a T-algebra homomorphism. Note that since $U(i^\sharp)$ is a reflexive coequaliser, $TU(i^\sharp)$ is an epi by assumption on T. Precomposing u with $U(i^\sharp)$ yields the T-algebra homomorphism f, so by $TU(i^\sharp)$ being an epi and Lemma A.1 we conclude u is a T-algebra homomorphism $C \to Z$. Reflexivity of the pair follows from Lemma 5.11. ◀

▶ **Proposition 5.7.** *For* $\mathsf{C} = \mathsf{Set}$ *and* T *any finitary monad, every language* $L\colon TI \to O$ *has a Nerode equivalence given by*

$$R = \{(u,v) \in TI \times TI \mid L \circ \mu \circ T[\mathsf{id}_{TI}, 1_u] = L \circ \mu \circ T[\mathsf{id}_{TI}, 1_v]\colon T(TI+1) \to O\}$$

with the corresponding projections $p_1, p_2\colon R \to TI$.

Proof. For each subset $X \subseteq R$, we define $p_X\colon R \to TI$ by

$$p_X(r) = \begin{cases} p_1(r) & \text{if } r \notin X \\ p_2(r) & \text{if } r \in X. \end{cases}$$

We have $Tp_1 = Tp_\emptyset$ by definition. Consider any $t \in TR$ and let a finite $E \subseteq R$ with inclusion map $e\colon E \to R$ and $t' \in TE$ be such that $T(e)(t') = t$. These exist because T is finitary. We will show by induction on E that

$$(L \circ \mu \circ Tp_\emptyset)(t) = (L \circ \mu \circ Tp_E)(t). \tag{3}$$

The case where $E = \emptyset$ is clear, so assume $E = E' \cup \{z\}$ with $z \notin E'$ and (3) holds when E' is substituted for E. We fix the singleton $1 = \{\Box\}$ and define $d\colon R \to TI + 1$ by

$$d(r) = \begin{cases} (\kappa_1 \circ p_1)(r) & \text{if } r \notin E \\ (\kappa_1 \circ p_2)(r) & \text{if } r \in E' \\ \kappa_2(\Box) & \text{if } r = z, \end{cases}$$

where κ_1 and κ_2 are the coproduct injections. By this definition, we have $[\mathsf{id}_{TI}, 1_{p_1(z)}] \circ d = p_{E'}$ and $[\mathsf{id}_{TI}, 1_{p_2(z)}] \circ d = p_E$, so

$$\begin{aligned}
(L \circ \mu \circ Tp_\emptyset)(t) &= (L \circ \mu \circ Tp_{E'})(t) && \text{(induction hypothesis)} \\
&= (L \circ \mu \circ T([\mathsf{id}_{TI}, 1_{p_1(z)}] \circ d))(t) \\
&= (L \circ \mu \circ T([\mathsf{id}_{TI}, 1_{p_2(z)}] \circ d))(t) && \text{(definition of } R) \\
&= (L \circ \mu \circ Tp_E)(t),
\end{aligned}$$

thus concluding the proof of (3). Now $Tp_1 = Tp_\emptyset$ by definition and

$$Tp_E(t) = T(p_E \circ e)(t') = T(p_2 \circ e)(t') = Tp_2(t),$$

from which we find that $(L \circ \mu \circ Tp_1)(t) = (L \circ \mu \circ Tp_\emptyset)(t) = (L \circ \mu \circ Tp_E)(t) = (L \circ \mu \circ Tp_2)(t)$. As this argument works for any $t \in TR$, we have $L \circ \mu \circ Tp_1 = L \circ \mu \circ Tp_2$.

Now consider any set S with $q_1, q_2 \colon S \to TI$ making

$$\begin{array}{ccccc} TS & \xrightarrow{Tq_2} & TTI & & \\ {\scriptstyle Tq_1}\downarrow & & \downarrow{\scriptstyle \mu} & & \\ TTI & \xrightarrow{\mu} & TI & \xrightarrow{L} & O \end{array} \tag{4}$$

commute, and assume q_1 and q_2 have a common section $j \colon TI \to S$. We define $u \colon S \to R$ by $u(s) = (q_1(s), q_2(s))$. To see that this is indeed an element of R, note that for $k \in \{1, 2\}$,

$$\begin{aligned} L \circ \mu \circ T[\mathsf{id}_{TI}, 1_{q_k(s)}] &= L \circ \mu \circ T[\mathsf{id}_{TI}, q_k \circ 1_s] \\ &= L \circ \mu \circ T[q_k \circ j, q_k \circ 1_s] && \text{(section)} \\ &= L \circ \mu \circ Tq_k \circ T[j, 1_s], \end{aligned}$$

and therefore $L \circ \mu \circ T[\mathsf{id}_{TI}, 1_{q_1(s)}] = L \circ \mu \circ T[\mathsf{id}_{TI}, 1_{q_2(s)}]$ follows from (4). By definition, u is the unique map making the diagram below commute.

$$\begin{array}{ccccc} & & S & & \\ & {\scriptstyle q_1}\swarrow & \downarrow{\scriptstyle u} & \searrow{\scriptstyle q_2} & \\ TI & \xleftarrow{p_1} & R & \xrightarrow{p_2} & TI \end{array}$$

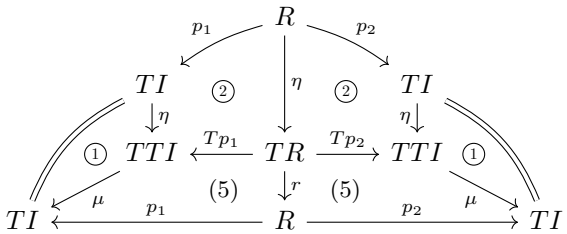
◀

▶ **Lemma 5.8.** *If* C *has coproducts, then for any Nerode equivalence* (R, p_1, p_2) *there exists a unique* T*-algebra structure* $u \colon TR \to R$ *making* p_1 *and* p_2 T*-algebra homomorphisms* $(R, u) \to (TI, \mu)$ *that have a common section.*

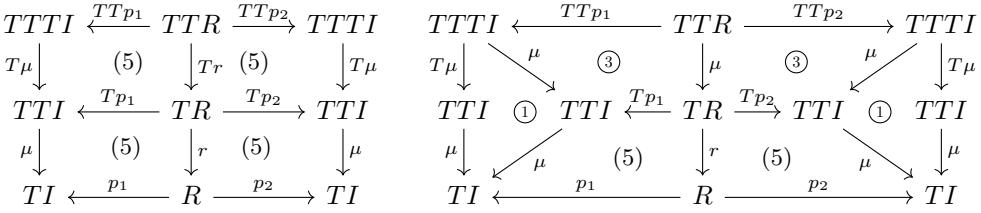
Proof. Let $L \colon TI \to O$ be a language with Nerode equivalence (R, p_1, p_2). Then (p_1, p_2) is a reflexive pair by the Nerode equivalence property, since $(\mathsf{id}_{TI}, \mathsf{id}_{TI})$ is a reflexive pair trivially satisfying the Nerode equivalence condition. We apply Lemma A.2 to obtain from the Nerode equivalence property a unique morphism $r \colon TR \to R$ making the diagram below commute.

$$\begin{array}{ccccc} TTI & \xleftarrow{Tp_1} & TR & \xrightarrow{Tp_2} & TTI \\ \downarrow{\scriptstyle \mu} & & \downarrow{\scriptstyle r} & & \downarrow{\scriptstyle \mu} \\ TTI & \xleftarrow{p_1} & R & \xrightarrow{p_2} & TI \end{array} \tag{5}$$

We need to show that (R, r) is a T-algebra. The first commutative diagram below shows that $r \circ \eta_R$ preserves p_1 and p_2, so since id_R also does this we must have $r \circ \eta_R = \mathsf{id}_R$ by the uniqueness property of the Nerode equivalence.

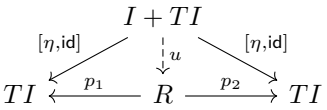


① monad law
② naturality of η
③ naturality of μ



As for the other two, we use a double application of Lemma A.2 to see that the pair $(\mu \circ T\mu \circ TTp_1, \mu \circ T\mu \circ TTp_2)$ satisfies the Nerode equivalence conditions. Commutativity of the two diagrams then shows that both $r \circ Tr$ and $r \circ \mu$ are the unique map commuting with the pairs $(\mu \circ T\mu \circ TTp_1, \mu \circ T\mu \circ TTp_2)$ and (p_1, p_2), so they must be equal and (TR, r) is a T-algebra.

It remains to show that p_1 and p_2 have a common section in $\mathcal{EM}(T)$. To this end, note that $([\eta_I, \mathsf{id}_{TI}], [\eta_I, \mathsf{id}_{TI}])$ is a reflexive pair trivially satisfying the Nerode equivalence condition. Thus, we obtain by the Nerode equivalence property a unique morphism $u \colon I + TI \to R$ making



commute. Then for $k \in \{1, 2\}$,

$$p_k \circ (u \circ \kappa_1)^\sharp = (p_k \circ u \circ \kappa_1)^\sharp = (p_k \circ [\eta_I, \mathsf{id}_{TI}])^\sharp = \eta_I^\sharp = \mathsf{id}_{(TI,\mu)}. \qquad \blacktriangleleft$$

▶ **Lemma 5.11.** *If $q_1, q_2 \colon A \to TB$ is a reflexive pair in C, then so is $(q_1^\sharp, q_2^\sharp)$ in $\mathcal{EM}(T)$.*

Proof. Assume $j \colon TB \to A$ is the common section of q_1 and q_2. Then, for $k \in \{1, 2\}$,

$$q_k^\sharp \circ (\eta_A \circ j \circ \eta_B)^\sharp = U((U(q_k^\sharp) \circ \eta_A \circ j \circ \eta_B)^\sharp) = U((q_k \circ j \circ \eta_B)^\sharp) = U(\eta_B^\sharp) = \mathsf{id}_{TB}. \qquad \blacktriangleleft$$