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
tcappe@cs.ucl.ac.uk

Jurriaan Rot
University College London, United Kingdom
Radboud University, Nijmegen, 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 algebras for a fixed endofunctor 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

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.\footnote{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.} This impeded the application of abstract...
Tree Automata as Algebras: Minimisation and Determinisation

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 determination, 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 algorithms for minimisation, determinisation, and equivalence. We embrace the categorical view on automata due to Arbib and Manes [8] to study bottom-up 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 η: Id → T and a multiplication μ: T^2 → T, which satisfy the compatibility laws μ ◦ η_T = id_T = μ ◦ Tη and μ ◦ μ_T = μ ◦ Tμ.

Example 2.1. The triple (P_I, {−}, ∪) is a monad on Set, where P_I is the finite powerset functor , {−} is the singleton operation, and ∪ is union of sets. Another example is the multiplicity monad (M_F, e, m) for a field F, where M_F is the functor sending a set X to M_F X = {ϕ: X → F | ϕ 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) = \(\sum_i s_i \varphi_i(x)\), where \(\cdot\) is the field multiplication.

Algebras and Varietors. We fix a functor Σ: C → C and write Alg(Σ) for the category of Σ-algebras. Throughout this paper, we assume that Σ is a varietor [6], i.e., that the forgetful functor U: Alg(Σ) → C admits a left adjoint F: C → Alg(Σ). The varietor Σ induces a monad (Σ^o, η, μ) on C, where Σ^o = UF. Given an object X of C, we refer to FX = (Σ^o X, α_X) as the free Σ-algebra over X. The free Σ-algebra satisfies the following: for every Σ-algebra Q and every morphism x: X → UQ of C, there is a unique Σ-algebra morphism x^#: (Σ^o X, α_X) → Q with U(x^#) ◦ η_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 := id | A | P_1 × P_2 | \(\prod_{i∈ I} P_i\) 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 \((Q, \delta, i, o)\) such that \((Q, \delta)\) is a Σ-algebra and \(i: I \to Q\) and \(o: Q \to O\) 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 \((Q, \delta, i, o)\) to an automaton \((Q', \delta', i', o')\) is a Σ-algebra homomorphism \(h: Q \to Q'\) (i.e., \(h \circ \Sigma h = h \circ \delta\)) 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 \(L: \Sigma^* I \to O\). In the context of an automaton \(A = (Q, \delta, i, o)\), we can think of \(U(i^2): \Sigma^* I \to Q\), 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 \(rch_A\) (or \(rch\) if \(A\) is obvious) for \(U(i^2)\). The language of \(A\) is the morphism \(L(A): \Sigma^* I \to O\) in \(C\) given by \(L(A) = o \circ rch_A\).

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

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

\[
\delta_{\gamma}(t) = i(\ell) \quad (\ell \in I) \\
\delta_{\gamma}(t_1, \ldots, t_k) = \delta_{\gamma}(\delta_{\gamma}(t_1), \ldots, \delta_{\gamma}(t_k))
\]

In other words, \(L(A)\) contains the trees that evaluate to a final state. The map \(\hat{\delta}\) above is the transpose \(i^2\) in the relevant adjunction between \(\text{Set}\) and \(\text{Alg}(\Sigma)\), where the left adjoint sends a set \(I\) to the \(\Sigma\)-algebra with carrier \(T(I)\) and the obvious structure map.

3.1 Nominal tree automata

To show the versatility of our definition, we instantiate it in the category \(\text{Nom}\) 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 \(A\) be a countable set of atoms, and let \(\text{Sym}(A)\) be the associated symmetry group, consisting of all permutations on \(A\). A nominal set is a pair \((X, \cdot)\) of a set \(X\) and a function \(\cdot: \text{Sym}(A) \times X \to X\) forming a left action of \(\text{Sym}(A)\) on \(X\).
Tree Automata as Algebras: Minimisation and Determinisation

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 \( \pi \in \text{Sym}(\mathbb{A}) \), if \( \pi \) is equal to \( \text{id}_A \) when restricted to \( A \), then \( \pi \cdot x = x \). The minimal such \( A \) is denoted \( \text{supp}(x) \), and can be understood as the set of “free” names of \( x \). Given \( x \in X \), its orbit is the set \( \{ \pi \cdot x \mid \pi \in \text{Sym}(\mathbb{A}) \} \). We say that a nominal set \( X \) is orbit-finite whenever it has finitely many orbits. An equivariant function \( f : (X, \cdot) \to (Y, \cdot) \) is a function \( X \to Y \) that respects permutations, i.e., \( f(\pi \cdot x) = \pi \cdot f(x) \).

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

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

\[
\begin{align*}
= & \; Q \text{ a nominal set;} \\
= & \; \delta \text{ consists of two equivariant functions } \delta_{\text{app}} : Q \times Q \to Q \text{ and } \delta_{\text{lambda}} : [\mathbb{A}]Q \to Q; \\
= & \; i : \mathbb{A} \to Q \text{ is an equivariant function, selecting states for parsing variables;} \\
= & \; o : Q \to 2 \text{ is an equivariant characteristic function of final states, which implies that if a state is final, so are all the states in its orbit.}
\end{align*}
\]

The reachability function is the equivariant function given by

\[
\text{rch}_A(t) = \begin{cases} 
  i(t) & \text{if } t \in \mathbb{A} \\
  \delta_{\text{app}}(\text{rch}_A(t_1), \text{rch}_A(t_2)) & \text{if } t = (t_1, t_2) \\
  \delta_{\text{lambda}}(\langle a \rangle \text{rch}_A(t')) & \text{if } t = \langle a \rangle t'.
\end{cases}
\]

The most interesting case is the last one: in order to parse the \( \alpha \)-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 \( \alpha \)-conversion w.r.t. \( \langle a \rangle \). Note that \( L(A) \) is equivariant, i.e., invariant under permutations of atoms. Thus \( A \) recognises \( \lambda \)-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 \( \Sigma \) 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 \( \Sigma e \) 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 $q: X \to X'$ identified up to the equivalence, i.e., an equivalence class. We denote by $\text{Quot}(X)$ the class of all quotients of $X$. The underlying category $C$ is said to be cowellpowered if $\text{Quot}(X)$ is a set for every $X$. In that case, if $C$ is cocomplete, $\text{Quot}(X)$ forms a complete lattice, with the order given by $\leq$, and the least upper bound (join) given by cointersection. We denote by $\text{Quot}_C(X)$ 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 $\text{Set}$ is cocomplete and cowellpowered, and so is $\text{Nom}$ 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 $(Q, \delta)$ be a $\Sigma$-algebra. A quotient algebra is a $\Sigma$-algebra $(Q', \delta')$ together with a quotient $q: Q \twoheadrightarrow Q'$ in $\mathcal{E}$ that is an algebra homomorphism. Given a $\Sigma$-tree automaton $(Q, \delta, i, o)$, a quotient automaton is a $\Sigma$-tree automaton $(Q', \delta', i', o')$ together with a quotient $q: Q \twoheadrightarrow Q'$ in $\mathcal{E}$ that is a homomorphism of automata.

Definition 4.3 (Minimisation). The minimisation of a $\Sigma$-tree automaton $(Q, \delta, i, o)$ is a quotient automaton $(Q_m, \delta_m, i_m, o_m)$, $q: Q \twoheadrightarrow Q_m$, such that for any quotient automaton $(Q', \delta', i', o')$, $q': Q \twoheadrightarrow Q'$ of $(Q, \delta, i, o)$ there exists a (necessarily unique) automaton homomorphism $h: Q' \rightarrow Q_m$ 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 $q: Q \twoheadrightarrow Q_m$ as the minimisation if there exist $\delta_m, i_m, o_m$ turning $(Q_m, \delta_m, i_m, o_m), q$ into the minimisation of $(Q, \delta, i, o)$.

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

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 $\Sigma$ admits minimisation of reachable automata if every reachable automaton over $\Sigma$ has a minimisation.

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

Lemma 4.6. The functor $\Sigma$ admits minimisation of reachable automata if and only if a minimal automaton exists for every language over $\Sigma$. In that case, if $A$ is reachable, then the minimisation of $A$ 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 $A$ is given by the minimal automaton accepting $\mathcal{L}(A)$. 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 \((Q, \delta, i, o)\) 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 \(\text{Quot}_E(Q)\), which is a complete lattice by Assumption 4.1.

\begin{definition}
Let \(f : \Sigma X \to Y\) be a morphism. The \((E)\)-cobase of \(f\) (if it exists) is the greatest quotient \(q \in \text{Quot}_E(X)\) such that there exists a morphism \(g \circ \Sigma q = f\).
\end{definition}

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 coinsertions.

\begin{theorem}[Existence of cobases]
Suppose \(\Sigma : C \to C\) preserves \(E\)-coinsertions. Then every map \(f : \Sigma X \to Y\) has an \(E\)-cobase, given by the coinsertion
\[
\bigvee \{q \in \text{Quot}_E(X) \mid \exists g. g \circ \Sigma q = f\}.
\]
\end{theorem}

\begin{proof}
For \(E\) the class of all epis, the dual is shown in [10, 46]. The proof goes through in the current, more general setting, using that \(E\) is closed under coinsertions.
\end{proof}

\begin{remark}
A \(\text{Set}\) functor preserves coinsertions iff it is finitary [6]. In particular, this is the case for polynomial functors. For \(\text{Nom}\) functors we can use that, in general, a functor preserves coinsertions if it is finitary and preserves reflexive coequalisers [6]. These conditions hold for polynomial \(\text{Nom}\) 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 \(\Sigma\)-algebra \((Q, \delta)\) and a quotient \(q: Q \to Q' \in \text{Quot}_E(Q)\), define the quotient \(\Theta_{\delta}(q): Q \to \Theta_{\delta}(Q')\) as the cobase of \(q \circ \delta\). This defines a monotone operator \(\Theta_{\delta} : \text{Quot}_E(Q) \to \text{Quot}_E(Q)\) that has the following important property (see [10, 46]):

\begin{lemma}
Suppose \(\Sigma\) preserves \(E\)-coinsertions. For any \(\Sigma\)-algebra \((Q, \delta)\), a quotient \(q: Q \to Q'\) in \(\text{Quot}_E(Q)\) satisfies \(q \leq \Theta_{\delta}(q)\) iff there is an algebra structure \(\delta': \Sigma Q' \to Q'\) turning \(q\) into an algebra homomorphism.
\end{lemma}

The operator \(\Theta_{\delta}\) allows us to quotient the transition structure of the automaton. In order to obtain the minimal automaton, we incorporate the output map \(o: Q \to O\) into the construction of a monotone operator based on \(\Theta_{\delta}\). For technical convenience, we assume that this map is an element of \(\text{Quot}_E(Q)\). The relevant monotone operator for minimisation is \(\Theta_{\delta} \land o\) (where the meet \(\land\) is taken pointwise in \(\text{Quot}_E(Q)\)).

\begin{example}
Let \(\Sigma : \text{Set} \to \text{Set}\) be a polynomial functor induced by signature \(\Gamma\). We first spell out what the cobase means concretely in this case and then study the operator \(\Theta_{\delta}\) in more detail. Since \(\Sigma\) is an endofunctor on \(\text{Set}\), the cobase of a map \(f : \Sigma X \to Y\) is the largest quotient \(q \in \text{Quot}_E(X)\) such that for all \(t, t' \in \Sigma X:\)
\[
\text{if } \Sigma q(t) = \Sigma q(t'), \text{ then } f(t) = f(t').
\]
\end{example}

\footnote{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 \( k = \text{ar}(\gamma) \), and any \( x_1, \ldots, x_k, y_1, \ldots, y_k \), we have that
\[
q(x_1) = q(y_1) \quad \ldots \quad q(x_k) = q(y_k)
\]
\[
f(\kappa_\gamma(x_1, \ldots, x_k)) = f(\kappa_\gamma(y_1, \ldots, y_k))
\]
or equivalently that for all \( x_1, \ldots, x_k \) and \( x'_i \) with \( 1 \leq i \leq k \) we have
\[
q(x_i) = q(x'_i)
\]
\[
f(\kappa_\gamma(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_k)) = f(\kappa_\gamma(x_1, \ldots, x_{i-1}, x'_i, x_{i+1}, \ldots, x_k))
\]
Suppose \( (Q, \delta, i, o) \) is an automaton. For \( q \in \text{Quot}(Q) \), we have \( q \leq \Theta_\delta(q) \land o \) iff
\begin{itemize}
  \item for all \( x, x' \in Q \): if \( q(x) = q(x') \), then \( o(x) = o(x') \); and
  \item for all \( \gamma \in \Gamma \) with \( k = \text{ar}(\gamma) \), and \( x_1, \ldots, x_k \) and \( x'_i \) with \( 1 \leq i \leq k \) we have
\end{itemize}
\[
q(x_i) = q(x'_i)
\]
\[
f(\kappa_\gamma(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_k)) = f(\kappa_\gamma(x_1, \ldots, x_{i-1}, x'_i, x_{i+1}, \ldots, x_k))
\]
A partition \( q \) with the above two properties is known as a forward bisimulation [25].

\textbf{Theorem 4.12.} Suppose \( \Sigma \) preserves \( \mathcal{E} \)-cointersections. Let \((Q, \delta, i, o)\) be an automaton, where \( o \in \text{Quot}_\mathcal{E}(Q) \). Then \( \text{gfp}(\Theta_\delta \land o) \) is the minimisation of \((Q, \delta, i, o)\).

\textbf{Proof.} Denote the quotient \( \text{gfp}(\Theta_\delta \land o) \) by \( q_m : Q \rightarrow Q_m \). Thus \( q_m \leq \Theta_\delta(q_m) \) and \( q_m \leq o \), hence (using Lemma 4.10) there exist \( \delta_m, o_m \) turning \( q_m \) into an automaton homomorphism from \((Q, \delta, i, o)\) to \((Q_m, \delta_m, q_m \circ i, o_m)\). We show that this is the minimisation of \((Q, \delta, i, o)\).

To this end, let \((Q', \delta', i', o')\), \( q' : Q \rightarrow Q' \) be a quotient automaton of \((Q, \delta, i, o)\). By Lemma 4.10 we get \( q' \leq \Theta_\delta(q') \), and since \( o' \circ o = o \) we have \( q' \leq o \), hence \( q' \leq \Theta_\delta(q') \land o \). Thus \( q' \leq \text{gfp}(\Theta_\delta \land o) \), i.e., there is a quotient \( h : Q' \rightarrow Q_m \) 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 \( \Sigma \) 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 \((Q, \delta, i, o)\) 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 \( \Theta_\delta \land o \), 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 \((Q, \delta, i, o)\) by iterating \( \Theta_\delta \land o \), starting from the top element \( \top \) of the lattice \( \text{Quot}_\mathcal{E}(Q) \). 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 \( \Theta_\delta \land o \) 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 \((Q, \delta, i, o)\) is called simple if for every quotient automaton \((Q', \delta', i', o')\) the associated quotient \(q: Q \rightarrow Q'\) 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 \(\Sigma\) preserves \(E\)-cointersections.

Proposition 4.14. Suppose \(\Sigma\) preserves \(E\)-cointersections. Let \((Q, \delta, i, o)\) be an automaton with \(o \in E\), and let \((Q', \delta', i', o')\), \(q: Q \rightarrow Q'\) be a quotient automaton. Then \((Q', \delta', i', o')\) is simple if and only if it is the minimisation of \((Q, \delta, i, o)\).

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

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

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

Corollary 4.15. If \(\Sigma\) preserves \(E\)-cointersections, then an automaton \(A = (Q, \delta, i, o)\) with \(o \in E\) is minimal if and only if it is simple and reachable.

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

Conversely, let \(A\) be simple and reachable. By reachability, \(A\) is a quotient automaton of \((\Sigma I, \alpha I, \eta I, L(A))\); also, \(L(A) = o \circ \text{rch} \in E\). Proposition 4.14 then tells us that \(A\) is the minimisation of \((\Sigma I, \alpha I, \eta I, L(A))\); by Lemma 4.5 we conclude that \(A\) 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 Adámk 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 \(\Sigma\). Instead, we focus on the monad \(\Sigma^o\) induced by its adjunction and generalise by fixing an arbitrary monad \((T, \eta, \mu)\) in \(C\). Let \(F \dashv U: C \Rightarrow \mathcal{EM}(T)\) be the adjunction with its category of (Eilenberg-Moore) algebras. Given a \(C\)-morphism \(f: X \rightarrow UY\) for \(X\) in \(C\) and \(Y\) in \(\mathcal{EM}(T)\), we write \(f^*: FX \rightarrow Y\) 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 \((Q, \delta, i, o)\), where \((Q, \delta)\) is a T-algebra and \(i : I \to Q\) and \(o : Q \to O\) are morphisms in \(C\). A homomorphism from \((Q, \delta, i, o)\) to \((Q', \delta', i', o')\) is a T-algebra homomorphism \(h : (Q, \delta) \to (Q', \delta')\) such that \(h \circ i = i'\) and \(o' \circ h = o\).

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

The \(\Sigma\)-tree automata defined in Section 3 are recovered using the following fact: the reachability map of a 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 \(E\) is only invoked to obtain the right notion of reachability.

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 \(E\) 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 \(i : B \to UC\) is such that \(U(i^\sharp)\) reflexively coequalises \(q_1, q_2 : A \to TB\) in \(C\), then \(i^\sharp\) reflexively coequalises \(q_1^\sharp, q_2^\sharp : FA \to FB\).

Before defining an abstract Nerode equivalence, we recall the classical definition for languages of words. Given a language \(L : A^* \rightarrow 2\), 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 \(Q \times A \rightarrow Q\) 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 \(p_1, p_2 : R \rightarrow A^* \cong 1 \times A^*\) are the projections, we note that \(R\) is defined to be the largest relation making the following diagram commute.

\[ \begin{array}{ccc}
R \times A^* & \xrightarrow{p_2 \times \text{id}} & 1 \times A^* \times A^* \\
\downarrow{p_1 \times \text{id}} & & \downarrow{\mu} \\
1 \times A^* \times A^* & \xrightarrow{\mu} & 1 \times A^* \\
\downarrow{L} & & \downarrow{L} \\
1 \times A^* & \xrightarrow{L} & 2
\end{array} \]

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 \(L : TI \rightarrow O\) and an object \(R\) with morphisms \(p_1, p_2 : R \rightarrow TI\), 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 \(q_1, q_2 : S \rightarrow TI\)

\(^3\) 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 \(E\) 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 \( u: S \to R \) 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 \( L: A^* \to 2 \), the Nerode equivalence as defined above is then the largest relation \( R \subseteq A^* \times A^* \) such that

\[
\begin{align*}
n \in \mathbb{N} & \quad (u_1, v_1), \ldots, (u_n, v_n) \in R \\
L(u_1 \cdots u_n) & = L(v_1 \cdots v_n).
\end{align*}
\]

Equivalently, \( R \) is the largest relation such that

\[
(u, v) \in R \quad w, x \in A^* \\
L(wux) = L(wvx),
\]

which is precisely the syntactic congruence of the language.

We can show that the Nerode equivalence in \( \text{Set} \) 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 \( 1_x: 1 \to X \) the constant \( x \) function, assuming no ambiguity of the set involved.

▶ Proposition 5.7. For \( C = \text{Set} \) and \( T \) any finitary monad, every language \( L: TI \to O \) has a Nerode equivalence given by

\[
R = \{(u, v) \in TI \times TI | L \circ \mu \circ T[\text{id}_{TI}, 1_u] = L \circ \mu \circ T[\text{id}_{TI}, 1_v]: T(TI + 1) \to O\}
\]

with the corresponding projections \( p_1, p_2: R \to TI \).

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 \( \mu \) have the same value under \( L \). A context is an element of \( T(TI + 1) \), where \( 1 = \{\square\} \) 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 \( \square \), are used in algorithms for minimisation [25] and learning [44, 19]. Unfortunately, the characterisation of Proposition 5.7 does not directly extend to \( \text{Nom} \), 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 \( L: TI \to O \) that has a Nerode equivalence, we use the equivalence to quotient the \( T \)-automaton \((FI, \eta, L)\). 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 \( u: TR \to R \) 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 \( L : T I \to O \) be the language with Nerode equivalence \((R, p_1, p_2)\) and \( c : T \to M \) 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 \((T I, \mu)\) that have a common section. Since \( T \) preserves reflexive coequalisers, they are lifted by \( U \), and we have a morphism \( m : T M \to M \) making \((M, m)\) a \( T \)-algebra such that \( c \) is a \( T \)-algebra homomorphism \((T I, \mu) \to (M, m)\).

Since the diagram below on the left commutes and \( c \) coequalises \( p_1 \) and \( p_2 \), there is a unique morphism \( o \) rendering the diagram on the right commutative.

Choosing \( i_M = c \circ \eta : I \to M \), 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 \( A = (Q, \delta, i, o) \) such that \( \mathcal{L}(A) = L \). Reachability amounts to the reachability map \( \text{rch} : T I \to UQ \) being the reflexive coequaliser of a pair of morphisms \( q_1, q_2 : S \to T I \). From commutativity of

we obtain by the Nerode equivalence property a unique morphism \( v : S \to R \) 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 \( \text{rch} = U(i^\sharp) \) is the reflexive coequaliser of \( q_1 \) and \( q_2 \), \( i^\sharp \) is the reflexive coequaliser of \( q_1^\sharp \) and

---

CALCO 2019
\\(q^2_2\) by Lemma 5.4. We then obtain a unique \(T\)-algebra homomorphism \(h: (Q, \delta) \rightarrow (M, m)\) making the diagram below on the left commute.

\[
\begin{array}{ccc}
FI & \xrightarrow{i^1} & Q \\
\downarrow h & & \downarrow \circ h \\
M & \xrightarrow{i^*_M} & (M, m)
\end{array}
\]

\[
\begin{array}{ccc}
TI & \xrightarrow{rch} & Q \\
\downarrow \circ h & & \downarrow h \\
M & \xrightarrow{\circ M} & (M, m)
\end{array}
\]

\[
\begin{array}{ccc}
& & \circ L(T) \\
\downarrow \circ h & & \downarrow \circ h \\
& & \circ M
\end{array}
\]

From commutativity of the other diagrams we find \(\circ M \circ h = \circ o\) (using that \(rch\) is epi) and \(h \circ i = i_M\). Thus, \(h\) is a \(T\)-automaton homomorphism \(A \rightarrow M\). To see that it is unique, note that any \(T\)-automaton homomorphism \(h': A \rightarrow M\) is a \(T\)-algebra homomorphism \((Q, \delta) \rightarrow (M, m)\) such that \(h' \circ i = i_M\). It is then not hard to see that \(h' \circ i^3 = (h' \circ i)^3 = i^*_M\).

We conclude that \(h' = h\) by the uniqueness property of \(h\) satisfying \(h \circ i^3 = i^*_M\).

\[\triangleright\]

\textbf{Remark 5.10.} We briefly discuss the conditions of the above theorem in the specific case of \(C = \text{Set}\) with \(T\) a finitary monad. This includes the setting of tree automata in \(\text{Set}\), as a monad on \(\text{Set}\) is finitary if and only if \(\text{EM}(T)\) 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 \(\text{Set}\) 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.

\textbf{Lemma 5.11.} If \(q_1, q_2: A \rightarrow TB\) is a reflexive pair in \(C\), then so is \((q_1^1, q_2^2)\) in \(\text{EM}(T)\).

\textbf{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.

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

\[
\begin{array}{ccc}
TK & \xrightarrow{T_{p_2}} & TTI \\
\downarrow Trch & & \downarrow \mu \\
TM & \xrightarrow{rch} & M
\end{array}
\]

\[
\begin{array}{ccc}
T_{p_1} & \xrightarrow{T_{rch}} & TI \\
\downarrow \delta_M & & \downarrow \circ L(M) \\
TTI & \xrightarrow{rch} & O
\end{array}
\]

\[
\begin{array}{ccc}
TS & \xrightarrow{T_{q_2}} & TTI \\
\downarrow T_{q_1} & & \downarrow \mu \\
TI & \xrightarrow{\circ L(M)} & O
\end{array}
\]

\[
\begin{array}{ccc}
TTI & \xrightarrow{\circ L(M)} & O
\end{array}
\]

Now if \(S\) with \(q_1, q_2: S \rightarrow TI\) is any reflexive pair making the diagram on the right commute, we let \(c: TI \rightarrow Q\) be the coequaliser of \(U(q^1_1)\) and \(U(q^2_2)\), 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 \(\delta: TQ \rightarrow Q\) making \(c: FI \rightarrow (Q, \delta)\) a \(T\)-algebra homomorphism that is the coequaliser of \(q^1_1\) and \(q^2_2\). We also have \(L(M) \circ U(q^1_1) = L(M) \circ U(q^2_2)\) by commutativity of the diagram on the right, so with \(c\) coequalising \(U(q^1_1)\) and \(U(q^2_2)\) there is a unique morphism \(o: Q \rightarrow O\) such that \(o \circ c = L(M)\). Setting \(i = c \circ \eta_I\), we have a
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 \((S, \eta, \mu)\) has an associated *Kleisli category* \( \mathcal{K}(S) \), whose objects are those of \( C \) and whose morphisms \( X \to Y \) are morphisms \( X \to SY \) in \( C \). Given such morphisms \( f: X \to Y \) and \( g: Y \to Z \), their (Kleisli) composition \( g \circ f \) is defined as \( \mu_Y \circ Sg \circ f \) in \( C \). The *Kleisli adjunction* \( J \dashv V : C \rightleftharpoons \mathcal{K}(S) \) is given by \( JX = X \), \( J(f \circ g) = \eta_Y \circ f \circ Vf \) and \( V(\mu_Y \circ SY) = SY \). Given\( \mu_Y \circ SY \). Given\( \mu_Y \circ SY \).

**Example 6.1.** The category \( \mathcal{K}(\mathcal{P}_1) \) has morphisms \( X \to \mathcal{P}_1Y \), 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 \( \mathcal{P}_1X \) and a relation \( R: X \to Y \) to the function

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

The category \( \mathcal{K}(\mathcal{M}_2) \) has morphisms \( X \to \mathcal{M}_2Y \) 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 \( f: X \to Y \) to the matrix \( Jf[x, f(x)] = 1 \), for \( x \in X \), and 0 elsewhere, and the right adjoint \( V \) maps a matrix \( X \to Y \) to the corresponding linear function \( \mathcal{M}_2X \to \mathcal{M}_2Y \).

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

\[
\begin{array}{ccc}
\Sigma & \Rightarrow & \Sigma \\
J \downarrow & & \downarrow J \\
\mathcal{K}(S) & \xrightarrow{\Sigma} & \mathcal{K}(S)
\end{array}
\]

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

In [23, Lemma 2.4] it is shown that a canonical distributive law in \( \text{Set} \) always exists in case \( \Sigma \) is polynomial and \( S \) is a commutative monad [31].
6:14 Tree Automata as Algebras: Minimisation and Determinisation

- **Example 6.2.** For Σ a polynomial Set endofunctor, the canonical distributive law λ: ΣP↑ = ΣP↑Σ can be directly defined as follows: λ Σ P (u,v) = {v ∈ ΣX | {v, u} ∈ img(ΣP↑1, ΣP↑2)}, where p^1 and p^2 are the left and right projections of the membership relation ∈ X ⊆ X × P↑X.

The multiplicity monad (M^2, e, m) admits a distributive law λ: ΣM^2 = ΣM^2Σ, inductively defined as follows, where ⊕ is the Kronecker product:

\[
λ^{id} = id_{M^2} \quad λ^A = e_A \quad λ^{Σ_1 × Σ_2} = ⊕ o (λ^{Σ_1} × λ^{Σ_2}) \quad λ^{I_i ∈ Σ_i} = [M^2(κ_i) o λ^{Σ_i}]_{i ∈ I}
\]

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

- **Definition 6.3 ((Σ, S)-tree automaton).** Given a monad S, and \( \hat{Σ}: \lambda(S) → S(Σ) \) extending a functor Σ on C, a (Σ, S)-tree automaton is a \( \hat{Σ} \)-tree automaton, i.e., a tuple (Q, δ, i, o), where (Q, δ) is a \( \hat{Σ} \)-algebra and i: I → Q and o: Q → O are morphisms in \( S(Σ) \).

- **Example 6.4.**

1. Let Γ be a signature, and let Σ be its signature functor. Then the extension of Σ to \( \lambda(\mathcal{P}) \) is obtained via the distributive law of Example 6.2, namely ΣX = X and

\[
\hat{Σ}(f: X → Y)(κ_γ(x_1, \ldots, x_{ar(γ)})) = (κ_γ(y_1, \ldots, y_{ar(γ)})) \mid y_j \in f(x_j) \text{ for } 1 ≤ j ≤ ar(γ)
\]

for γ ∈ Γ. Non-deterministic tree automata are (Σ, P)-tree automata (Q, δ, i, o) with O = 1, the singleton set. In fact, we have that δ: \( \prod_{γ ∈ Γ} Q^{ar(γ)} → Q \) is a family of relations δ_γ ⊆ Q^{ar(γ)} × Q; by the same token, i ⊆ I × Q relates the frontier alphabet with (possibly several) states, and o ⊆ Q × 1 ⊆ 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 \( \lambda(\mathcal{M}) \) is obtained via the distributive law of Example 6.2. Explicitly, \( \hat{Σ}X = X \hat{Σ} \) and \( \hat{Σ}(f: X → Y) \) maps κ_γ(x_1, \ldots, x_{ar(γ)}) to the vector \([κ_γ(v_j)]_{j ∈ Y^{ar(γ)}}\) such that v_j is the z-th component of f(x_1) ⊗ ⋯ ⊗ f(x_{ar(γ)}), for γ ∈ Γ. A multiplicity tree automaton [29] is a (Σ, M^2)-tree automaton (Q, δ, i, o) with I = O = 1. In fact, we have that δ_γ is the transition matrix Q^{ar(γ)} → Q in \( \mathbb{F}^{Q^{ar(γ)} × Q} \); similarly, i: 1 → Q is the initial weight vector in \( \mathbb{F}^{1 × Q} \), and o: Q → 1 is the final weight vector in \( \mathbb{F}^{Q × 1} \). Intuitively, δ_γ maps an ar(γ)-tuple of elements of Q to a linear combination over Q. We note that we can go beyond fields and consider (Σ, M^2)-tree automata for a semiring S, encompassing weighted tree automata [17].

3. The nondeterministic version of the automata defined in Section 3.1 can be obtained via the monad \( \mathcal{P}_ω: \text{Nom} → \text{Nom} \), mapping a nominal set to the nominal set of its finitely-supported, orbit-finite subsets. This is analogous to non-deterministic nominal automata [13]. We note that \( \lambda(\mathcal{P}_ω) \) is precisely the category of nominal sets and (orbit-finitely branching) equivariant relations and that \( \Sigma_λ \) extends to relations just as a set endofunctor – the distributive law is defined as the one for \( \mathcal{P}_t \) of Example 6.2, where all the maps are equivariant. A nondeterministic nominal (Sigma, P)-tree automaton is a (Sigma, M^2)-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, δ_{ambida} ⊆ [A]Q × Q, and o is an equivariant subset of Q.

Now we 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 \( \text{Alg}(Σ) \) and \( \text{Alg}(\hat{Σ}) \) 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 $\Sigma : C \to C$ be a functor, $S : C \to C$ a monad, and $\widehat{\Sigma} : \text{Alg}(S) \to \text{Alg}(\Sigma)$ an extension of $\Sigma$. Let $\lambda : \Sigma S \Rightarrow SS\Sigma$ be the corresponding distributive law. Then the Kleisli adjunction $J \dashv V : C \leftrightarrows \text{Alg}(\Sigma)$ lifts to an adjunction in $\text{Alg}(\Sigma)$:

\[
\begin{array}{ccc}
\text{Alg}(\Sigma) & \xrightarrow{\eta} & \text{Alg}(\widehat{\Sigma}) \\
\downarrow J & & \downarrow V \\
C & \xleftarrow{\delta} & \text{Kl}(S)
\end{array}
\]

$J(Q, \delta : \Sigma Q \to Q) = (Q, J(\delta) : \widehat{\Sigma}Q \to Q)$

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

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

\[
\begin{array}{ccc}
\Sigma^\circ X & \xrightarrow{\alpha} & \Sigma^\circ I \\
\Sigma Q & \xrightarrow{\text{rch}} & SQ \\
\Sigma^\circ Q & \xrightarrow{\widehat{\Sigma} \text{rch}} & \widehat{\Sigma} Q
\end{array}
\]

\[
\begin{array}{ccc}
\Sigma^\circ I & \xrightarrow{\eta} & I \\
\widehat{\Sigma} Q & \xrightarrow{\widehat{\Sigma} \text{rch}} & Q
\end{array}
\] (2)

Note that the adjoint transpose of $\text{rch}$ is the same morphism, seen in $\text{Kl}(S)$. The functor $V$ can be viewed as a determinisation construction for $(\Sigma, S)$-tree automata.

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

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

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

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 $(\Sigma, P_f)$-tree automaton $(Q, \delta, i, o)$ is

\[
\overline{\delta}(x_1, \ldots, X_k) = \bigcup_{x_1 \in X_1, \ldots, x_k \in X_k} \delta_\gamma(x_1, \ldots, x_k)
\]

for $\gamma \in \Gamma$ with $k = \text{ar}(\gamma)$, and $X_1, \ldots, X_k, X$ 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

\[
\text{rch}(t) = \begin{cases} 
\bar{i}(t) & \text{if } t \in I \\
\bigcup_{x_1 \in \text{rch}(t_1)} \delta_{\gamma}(x_1, \ldots, x_k) & \text{if } t = (\gamma, x_1, \ldots, x_k), k = \text{ar}(\gamma).
\end{cases}
\]

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

\[
L(Q, \delta, i, o)(t) = \bigcup_{s \in \text{rch}(t)} o(s).
\]
That is: a tree $t$ is accepted by $(Q, \delta, i, o)$ whenever there is a final state among those reached by parsing $t$.

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

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

where $\gamma \in \Gamma$ and $\text{ar}(\gamma) = k$; 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 $\varphi$ over $k$-tuples of states via the distributive law (defined as the Kronecker product, see Example 6.2), i.e., $\varphi(q_1, \ldots, q_k) = \varphi_1(q_1) \cdots \varphi_k(q_k)$, for $q_1, \ldots, q_k \in Q$. The result is then multiplied by the matrix $\delta_\gamma$ to compute the successor vector.

Similarly, the reachability function becomes

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

Hence we obtain the language $L(Q, \delta, i, o)(t) = \text{rch}(t) \bullet o$, 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,

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

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

## 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. Parosh Aziz Abdulla, Johanna Högberg, and Lisa Kaati. Bisimulation Minimization of Tree Automata. *Int. J. Found. Comput. Sci.*, 18(4):699–713, 2007. doi:10.1142/S0129054107004929.
2. Jiří Adámek. Free algebras and automata realizations in the language of categories. *Commentationes Mathematicae Universitatis Carolinae*, 15(4):589–602, 1974. URL: http://eudml.org/doc/16649.
6:18 Tree Automata as Algebras: Minimisation and Determinisation

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

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

We need the following additional lemmas in the proofs below.

**A Proofs for Section 5**

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

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

We need the following additional lemmas in the proofs below.
Lemma A.1. If \( f : A \rightarrow B \) and \( h : A \rightarrow C \) in \( \mathcal{EM}(T) \) are such that there exists \( g : UB \rightarrow UC \) in \( C \) with \( g \circ f = h \) and \( Tf \) is an epi, then \( g \) is a \( T \)-algebra homomorphism \( B \rightarrow C \).

Proof. Let \( \alpha : TA \rightarrow A \), \( \beta : TB \rightarrow B \), and \( \gamma : TC \rightarrow C \) be the respective \( T \)-algebra structures on \( A \), \( B \), and \( C \). By commutativity of

\[
\begin{array}{c}
TA \xrightarrow{Tf} TB \\
\downarrow \alpha \quad \downarrow Tf \\
A \xrightarrow{h} TC \\
\downarrow f \quad \downarrow \gamma \\
B \xrightarrow{\beta} C
\end{array}
\]

and \( Tf \) being an epi, we directly conclude that \( g \) is a \( T \)-algebra homomorphism \( B \rightarrow C \).

Lemma A.2. Given a language \( L \) and \( q_1, q_2 : S \rightarrow T \) making the diagram below on the left commute, the diagram on the right commutes.

\[
\begin{array}{ccc}
TS & \xrightarrow{Tq_2} & TTI \\
\downarrow Tq_1 & \mu & \downarrow L \\
TI & \xrightarrow{\mu} & O
\end{array}
\quad
\begin{array}{ccc}
TTS & \xrightarrow{TTq_2} & TTTI \\
\downarrow TTq_1 & \mu & \downarrow L \\
TTTI & \xrightarrow{\mu} & O
\end{array}
\]

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

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

\[
\begin{array}{ccc}
TTS & \xrightarrow{T\mu} & TTTI \\
\downarrow TTq_1 & \mu & \downarrow L \\
TTTI & \xrightarrow{\mu} & O
\end{array}
\quad
\begin{array}{ccc}
\circ \\
TS & \xrightarrow{Tq_2} & TTI \\
\downarrow Tq_1 & \mu & \downarrow L \\
TTTI & \xrightarrow{\mu} & O
\end{array}
\]

As for reflexivity, \((\mu \circ Tq_1, \mu \circ Tq_2)\) is the composition of the reflexive pairs \((\mu_1, \mu_1)\) and \((Tq_1, Tq_2)\).

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

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

\[
Uf \circ q_1 = Uf \circ U(q_1^\#) \circ \eta_A = Uf \circ U(q_2^\#) \circ \eta_A = Uf \circ q_2,
\]

which because \( U(i^\#) \) coequalises \( q_1 \) and \( q_2 \) yields a unique function \( u : UC \rightarrow UZ \) such that \( u \circ rch = f \). Remains to show that \( u \) is a \( T \)-algebra homomorphism. Note that since \( U(i^\#) \) is a reflexive coequaliser, \( TU(i^\#) \) is an epi by assumption on \( T \). Precomposing \( u \) with \( U(i^\#) \) yields the \( T \)-algebra homomorphism \( f \), so by \( TU(i^\#) \) being an epi and Lemma A.1 we conclude \( u \) is a \( T \)-algebra homomorphism \( C \rightarrow Z \). Reflexivity of the pair follows from Lemma 5.11.
Proposition 5.7. For \( C = \text{Set} \) and \( T \) any finitary monad, every language \( L: TI \to O \) has a Nerode equivalence given by

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

with the corresponding projections \( p_1, p_2: R \to T_I \).

Proof. For each subset \( X \subseteq R \), we define \( p_X: R \to T_I \) by

\[
p_X(r) = \begin{cases} p_1(r) & \text{if } r \not\in X \\ p_2(r) & \text{if } r \in X. \end{cases}
\]

We have \( Tp_1 = Tp_0 \) by definition. Consider any \( t \in TR \) and let a finite \( E \subseteq R \) with inclusion map \( e: E \to R \) 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

\[
(L \circ \mu \circ Tp_0)(t) = (L \circ \mu \circ Tp_E)(t). \tag{3}
\]

The case where \( E = \emptyset \) is clear, so assume \( E = E' \cup \{z\} \) with \( z \not\in E' \) and (3) holds when \( E' \) is substituted for \( E \). We fix the singleton \( 1 = \{\square\} \) and define \( d: R \to TI + 1 \) by

\[
d(r) = \begin{cases} (\kappa_1 \circ p_1)(r) & \text{if } r \not\in E \\ (\kappa_1 \circ p_2)(r) & \text{if } r \in E' \\ \kappa_2(\square) & \text{if } r = z, \end{cases}
\]

where \( \kappa_1 \) and \( \kappa_2 \) are the coproduct injections. By this definition, we have \( [\text{id}_{TI}, 1_{p_1(z)}] \circ d = p_{E'} \) and \( [\text{id}_{TI}, 1_{p_2(z)}] \circ d = p_E \), so

\[
(L \circ \mu \circ Tp_0)(t) = (L \circ \mu \circ Tp_{E'})(t) \tag{induction hypothesis}
\]

\[
= (L \circ \mu \circ T([\text{id}_{TI}, 1_{p_1(z)}] \circ d))(t) 
= (L \circ \mu \circ T([\text{id}_{TI}, 1_{p_2(z)}] \circ d))(t) \tag{definition of R}
= (L \circ \mu \circ Tp_E)(t),
\]

thus concluding the proof of (3). Now \( Tp_1 = Tp_0 \) 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 \( L \circ \mu \circ Tp_1(t) = (L \circ \mu \circ Tp_0)(t) = (L \circ \mu \circ Tp_E)(t) = (L \circ \mu \circ Tp_2)(t) \).

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 \( q_1, q_2: S \to TI \) making

\[
\begin{array}{c}
\begin{array}{c}
TS \\
Tq_1
\end{array}
\end{array}
\xrightarrow{Tq_2}
\begin{array}{c}
\begin{array}{c}
TTI \\
T\mu
\end{array}
\end{array}
\xrightarrow{T\mu}
\begin{array}{c}
\begin{array}{c}
TI \\
T\kappa
\end{array}
\end{array}
\xrightarrow{T\kappa}
\begin{array}{c}
\begin{array}{c}
O \\
T_1
\end{array}
\end{array}
\xrightarrow{T_1}
\end{array}
\tag{4}
\]

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

\[
L \circ \mu \circ T[\text{id}_{TI}, 1_{q_k(s)}] = L \circ \mu \circ T[\text{id}_{TI}, q_k \circ 1_s] 
= L \circ \mu \circ T[q_k \circ j, q_k \circ 1_s] \tag{section}
= L \circ \mu \circ Tq_k \circ T[j, 1_s],
\]
and therefore $L \circ \mu \circ T[\id_{TI}, 1_{\eta(I)}] = L \circ \mu \circ T[\id_{TI}, 1_{\eta(T)}]$ follows from (4). By definition, $u$ is the unique map making the diagram below commute.

$\begin{align*}
TI & \xrightarrow{u} R \\
& \downarrow u \\
& \downarrow p_2 \\
& TI
\end{align*}$

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

**Proof.** Let $L: TI \rightarrow O$ 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 $(\id_{TI}, \id_{TI})$ 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 $r: TR \rightarrow R$ making the diagram below commute.

$\begin{align*}
TTTI & \leftarrow TTp_1 \\
& \downarrow T\mu \\
TR & \xrightarrow{T\mu} TTI \\
& \downarrow \mu \\
TI & \xrightarrow{\eta} RI \\
& \downarrow \eta \\
& \downarrow p_2 \\
& TI
\end{align*}$

(5)

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

$\begin{align*}
TTTI & \leftarrow TTp_1 \\
& \downarrow T\mu \\
TR & \xrightarrow{T\mu} TTI \\
& \downarrow \mu \\
TI & \xrightarrow{\eta} RI \\
& \downarrow \eta \\
& \downarrow p_2 \\
& TI
\end{align*}$

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 $r \circ \eta$ 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 $([\eta_I, \id_{TI}], [\eta_I, \id_{TI}])$ is a reflexive pair trivially satisfying the Nerode equivalence condition. Thus, we obtain by the Nerode equivalence property a unique morphism $u: I + TI \rightarrow R$ making

$\begin{align*}
TI & \xleftarrow{p_1} R \\
& \downarrow u \\
& \downarrow p_2 \\
& TI
\end{align*}$
commute. Then for \( k \in \{1, 2\} \),
\[
(p_k \circ (u \circ \kappa_1)^\sharp = (p_k \circ u \circ \kappa_1)^\sharp = (p_k \circ [\eta_I, \text{id}_{T^I}])^\sharp = \eta_I^\sharp = \text{id}_{(T^I, u)}).
\]

\( \blacktriangleleft \)

\( \blacktriangleright \) **Lemma 5.11.** If \( q_1, q_2 : A \rightarrow TB \) is a reflexive pair in \( C \), then so is \( (q_1^\sharp, q_2^\sharp) \) in \( EM(T) \).

**Proof.** Assume \( j : TB \rightarrow A \) is the common section of \( q_1 \) and \( q_2 \). Then, for \( k \in \{1, 2\} \),
\[
q_k^\sharp \circ (\eta_A \circ j \circ \eta_B)^\sharp = U((U(q_k^\sharp) \circ \eta_A \circ j \circ \eta_B)^\sharp) = U((q_k \circ j \circ \eta_B)^\sharp) = U(\eta_B^\sharp) = \text{id}_{TB}.
\]

\( \blacktriangleleft \)