A COALGEBRAIC TAKE ON REGULAR AND ω-REGULAR BEHAVIOURS

TOMASZ BRENGOS

Abstract. We present a general coalgebraic setting in which we define finite and infinite behaviour with Büchi acceptance condition for systems whose type is a monad. The first part of the paper is devoted to presenting a construction of a monad suitable for modelling (in)finite behaviour. The second part of the paper focuses on presenting the concepts of a (coalgebraic) automaton and its (ω-) behaviour. We end the paper with coalgebraic Kleene-type theorems for (ω-) regular input. The framework is instantiated on non-deterministic (Büchi) automata, tree automata and probabilistic automata.

Contents

Key words and phrases. bisimulation, coalgebra, epsilon transition, labelled transition system, tau transition, internal transition, logic, monad, Büchi automata, Buechi automata, saturation, weak bisimulation, infinite trace.

1. Introduction

Automata theory is one of the core branches of theoretical computer science and formal language theory. One of the most fundamental state-based structures considered in the literature is a non-deterministic automaton and its relation with languages. Non-deterministic automata with a finite state-space are known to accept *regular languages*. These languages are characterized as subsets of words over a fixed finite alphabet that can be obtained from simple languages via a finite number of applications of three types of operations: union, concatenation and the Kleene star operation [27, 33]. This result is known under the name of *Kleene theorem for regular languages* and readily generalizes to other types of finite input (see e.g. [41]).

$$R ::= \varnothing \mid a, a \in \Sigma_\varepsilon \mid R + R \mid R \cdot R \mid R^*$$

Figure 1. Regular expression grammar

On the other hand, non-deterministic automata have a natural infinite semantics which is given in terms of infinite input satisfying the so-called Büchi acceptance condition (or *BAC* in short). The condition takes into account the terminal states of the automaton and requires them to be visited infinitely often. It is a common practise to use the term *Büchi automata* in order to refer to automata whenever their infinite semantics is taken into consideration.

Although the standard type of infinite input of a Büchi automaton is the set of infinite words over a given alphabet, other types (e.g. trees) are also commonly studied [41]. The class of languages of infinite words accepted by Büchi automata can also be characterized akin to the characterization of regular languages. This result is known under the name of *Kleene theorem for ω-regular languages* and its variants hold for many input types (see e.g. [15,23,31,41]).

input type	*Kleene theorem*	*where*
ω-words	$\bigcup_{i=1}^n L_i \cdot R_i^\omega$	$R_i, L_i =$ regular lang.
ω-trees	$T_0 \cdot [T_1 \ldots T_n]^\omega$	$T_i =$ regular tree lang.

Figure 2. Kleene theorems for ω-regular input

Roughly speaking, any language recognized by a Büchi automaton can be represented in terms of regular languages and the infinite iteration operator $(-)^\omega$. This begs the question whether these systems can be placed in a unifying framework and reasoned about on a more abstract level so that the analogues of Kleene theorems for (ω-)regular input are derived. The recent developments in the theory of coalgebra [16,42,45,50] show that the coalgebraic framework may turn out to be suitable to achieve this goal.

A coalgebra $X \to FX$ is an abstract (categorical) representation of a computation of a process [24,42]. The coalgebraic setting has already proved itself useful in modelling finite behaviour via least fixpoints (e.g. [11,26,45]) and infinite behaviour via greatest fixpoints of suitable mappings [17,29,49]. The infinite behaviour with BAC can be modelled by a combination of the two [40,50].

We plan to revisit the coalgebraic framework of (in)finite behaviour from the perspective of systems whose type functor is a monad. In the coalgebraic literature [10–14] these systems are often referred to by the name of *systems with internal moves*. This name is motivated by the research on a unifying theory of finite behaviour for systems with internal steps [9–11,13,14,45]. They arise in a natural manner in many branches of theoretical computer science, among which are process calculi [37] (labelled transition systems and their weak bisimulation) or automata theory (automata with ε-moves), to name only two. Intuitively, these systems have a special computation branch that is silent. This special branch, usually labelled by the letter τ or ε, is allowed to take several steps and is, in some, a neutral part of the process. As thoroughly discussed in [11], the nature of this type of transition suggests it is in fact (part of) the unit of a monad. Hence, from our point of view the following terms become synonymous:

$$\text{coalgebras with internal moves} \;=\; \text{coalgebras whose type is a monad.}$$

This observation allows for an elegant modelling of several coalgebraic behavioural equivalences which take silent steps into account [11, 13, 14]. If the type T of a coalgebra $\alpha : X \to TX$ is a monad then the map α becomes an endomorphism $\alpha : X \multimap X$ in the Kleisli category for T: a natural and simple setting to study composition and fixpoints. For instance, if T is taken to be the monad modelling labelled transition systems [11] then Milner's weak bisimulation [37] of an LTS given by α is a strong bisimulation on its *saturation* α^*, i.e. the smallest LTS over the same state space s.t. $\alpha \leq \alpha^*$, $\mathsf{id} \leq \alpha^*$ and $\alpha^* \cdot \alpha^* \leq \alpha^*$ (where the composition and the order are given in the Kleisli category for the LTS monad) [11]. Hence, intuitively, α^* is the reflexive and transitive closure of α and is formally defined as the least fixpoint $\alpha^* = \mu x.(\mathsf{id} \vee x \cdot \alpha)$. The fact that labelled transition systems' weak bisimulation can be modelled via saturation of endomorphisms of a given Kleisli category allows for an instant generalization of the setting to other systems (e.g. probabilistic [11, 13]). The only requirement is that type functor is a monad whose Kleisli category satisfies suitable conditions for the definition of $(-)^*$ to be meaningful.

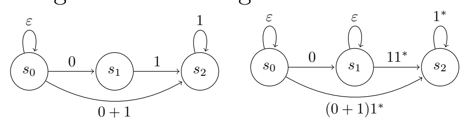


FIGURE 3. LTS with ε-moves and its saturation

The reflexive and transitive closure $\alpha \mapsto \alpha^*$ is understood as an accumulation of a *finite* number of compositions of the structure with itself. Hence, the concept of coalgebraic saturation is intrinsically related to *finite* behaviour of systems with a monadic type. A similar treatment of infinite behaviour (and their combination used to model Büchi acceptance condition) in the context of coalgebras whose type is a monad has not been considered so far. The closest to this goal would be [49,50], where (in)finite trace semantics is given in the setting of TF-coalgebras for a monad T and an endofunctor F. We take this treatment one step further and embed TF into a *monad* TF^∞ which is tailored to modelling (in)finite behaviours and their combinations. The new setting allows us to present clear definitions of coalgebraic (in)finite semantics and reason about them. In particular, it allows us to state

Kleene theorems for regular and ω-regular behaviours which would be challenging without the monadic types.

1.1. **Motivations.** Our purpose is to build a single coalgebraic setting that allows us to easily present definitions of (in)finite behaviours and reason about them aiming at their algebraic characterization. By finding a suitable monad T describing the type of systems taken into consideration we are able to state generic Kleene theorems connecting syntax and semantics of languages: the former imposed by the canonical algebraic nature of T and the latter given by T-automata and their behaviours.

By presenting a recipe to extend a functor to a suitable monad, we automatically encompass systems with invisible steps. However, this should not be viewed as our primary goal. Instead, from our point of view, it should be perceived as a by-product.

1.2. **The aim of the paper.** We plan to:

(A) revisit non-deterministic (tree) automata and their behaviour in the coalgebraic context of systems whose type is a monad,
(B) provide a type monad suitable for modelling (in)finite behaviour of general systems,
(C) present a setting for defining (in)finite behaviour for abstract automata whose type is a monad,
(D) state and prove coalgebraic Kleene theorems for (ω-)regular behaviour,
(E) put probabilistic automata into the framework.

The first point is achieved in Section 3 by describing non-deterministic (tree) automata and their finite and infinite behaviour in terms of different coalgebraic (categorical) fixpoint constructions calculated in the Kleisli category for a suitable monad. Section 3 serves as a motivation for the framework presented later in Section 4 and Section 5.

Originally [25, 45], coalgebras with internal moves were considered as systems $X \to TF_\varepsilon X$ for a monad T and an endofunctor F, where $F_\varepsilon \triangleq F + \mathcal{I}d$. Under some conditions the functor TF_ε can be embedded into the monad TF^*, where F^* is the free monad over F [11]. The monad TF^* is sufficient to model systems with internal moves and their finite behaviour [9, 11, 13]. However, it will prove itself useless in the context of infinite behaviour. Hence, by revisiting and tweaking the construction of TF^* from [11], Section 4 gives a general description of the monad TF^∞, the type functor TF (or TF_ε) embeds into, which is used in the remaining part of the paper to model the combination of finite and infinite behaviour. The reason why we find the expressive power of TF^∞ suitable is the following: the functor F^∞ is defined for any X as the carrier of the coproduct of the free algebra F^*X over X and the algebra F^ω obtained by inversing the final coalgebra map. Hence, by slighty abusing the notation, we can write $F^\infty = F^* \oplus F^\omega$.

Item (C) in the above list is achieved by using two fixpoint operators: the saturation operator $(-)^*$ and a new operator $(-)^\omega$ defined in the Kleisli category for a given monad. The combination of $(-)^*$ and $(-)^\omega$ allows us to define infinite behaviour with BAC.

Kleene-type theorems of (D) are a direct consequence of the definitions of finite and infinite behaviour with BAC using $(-)^*$ and $(-)^\omega$.

Finally, in Section 6 we put probabilistic automata into the framework of (in)finite behaviour for systems whose type is a monad.

This paper is an extended version of [12] with all missing proofs and additional Section 6 where probabilistic automata are considered.

2. Basic notions

We assume the reader is familiar with basic category theory concepts like a category, a functor, an adjunction. For a thorough introduction to category theory the reader is referred to [36]. See also e.g. [10,11,13] for an extensive list of notions needed here.

2.1. **Non-deterministic automata.**

The purpose of this subsection and the next one is to recall basic definitions and properties of non-deterministic automata and their tree counterparts: an automaton, its (ω-)language and Kleene theorems for regular and ω-regular languages. Note that the aim of this paper is to take these notions and statements and generalize them to the categorical setting.

Classically, a *non-deterministic automaton*, or simply *automaton*, is a tuple $\mathcal{Q} = (Q, \Sigma, \delta, q_0, \mathfrak{F})$, where Q is a finite set of *states*, Σ is a finite set called *alphabet*, $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ a *transition function* and $\mathfrak{F} \subseteq Q$ set of *accepting states*. We write $q_1 \stackrel{a}{\to} q_2$ if $q_2 \in \delta(q_1, a)$. There are two standard types of semantics of automata: finite and infinite. The finite semantics, also known as the *language of finite words* of $\mathcal{Q}$, is defined as the set of all finite words $a_1 \ldots a_n \in \Sigma^*$ for which there is a sequence of transitions $q_0 \stackrel{a_1}{\to} q_1 \stackrel{a_2}{\to} q_2 \ldots q_{n-1} \stackrel{a_n}{\to} q_n$ which ends in an accepting state $q_n \in \mathfrak{F}$ [27]. The infinite semantics, also known as the ω*-language* of $\mathcal{Q}$, is the set of infinite words $a_1 a_2 \ldots \in \Sigma^\omega$ for which there is a run $r = q_0 \stackrel{a_1}{\to} q_1 \stackrel{a_2}{\to} q_2 \stackrel{a_3}{\to} q_3 \ldots$ for which the set of indices $\{i \mid q_i \in \mathfrak{F}\}$ is infinite, or in other words, the run r visits the set of final states $\mathfrak{F}$ infinitely often. Often in the literature, in order to emphasize that the infinite semantics is taken into consideration the automata are referred to as *Büchi automata* [41]. In our work we consider (Büchi) automata without the initial state specified and define the (ω-)language in an automaton for any given state (see Section 3 for details).

2.1.1. *Kleene theorems.*

Finite and infinite semantics of non-deterministic automata can be characterized in terms of two Kleene theorems (see e.g. [27,41]). The first statement is the following. A language $L \subseteq \Sigma^*$ is a language of finite words of an automaton $\mathcal{Q}$ (a.k.a. *regular language*) if and only if it is a *rational language*, i.e. it can be obtained from languages of the form $\varnothing$ and $\{a\}$ for any $a \in \Sigma$ by a sequence of applications of finite union, concatenation and Kleene star operation with the latter two given respectively by:

$$R_1 \cdot R_2 \triangleq \{w_1 w_2 \mid w_1 \in R_1, w_2 \in R_2\},$$
$$R^* \triangleq \{w_1 \ldots w_n \mid w_i \in R \text{ and } n = 0, 1, \ldots\},$$

for $R_1, R_2, R \subseteq \Sigma^*$.

The second Kleene theorem focuses on ω-languages. A language $L_\omega \subseteq \Sigma^\omega$ is an ω-language (a.k.a. ω*-regular language*) of an automaton $\mathcal{Q}$ if and only if it is ω*-rational*, i.e. it can be written as a finite union

$$L_\omega = L_1 \cdot R_1^\omega \cup \ldots \cup L_n \cdot R_n^\omega, \tag{2.1}$$

where L_i, R_i are regular languages, the language $R_i^\omega \subseteq \Sigma^\omega$ is given by[1]:

$$R_i^\omega \triangleq \begin{cases} \{w_1 w_2 w_3 \ldots \mid w_i \in R_i\} & \varepsilon \notin R_i, \\ \Sigma^\omega & \text{otherwise} \end{cases}$$

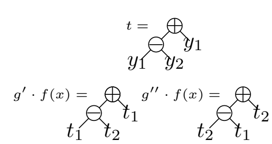
and $L \cdot R_\omega \triangleq \{wv \mid w \in L, v \in R_\omega\}$ for $L \subseteq \Sigma^*$ and $R_\omega \subseteq \Sigma^\omega$.

2.2. **Tree automata.** There are several other variants of input for non-deterministic Büchi automata known in the literature [23,41]. Here, we focus on *non-deterministic (Büchi) tree automaton*, i.e. a tuple $(Q, \Sigma, \delta, \mathfrak{F})$, where $\delta : Q \times \Sigma \to \mathcal{P}(Q \times Q)$ and the rest is as in the case of standard non-deterministic automata. The infinite semantics of this machine is given by a set of infinite binary trees with labels in Σ for which there is a run whose every branch visits $\mathfrak{F}$ infinitely often [23, 41]. We recall these notions here below (with minor modifications to suit our language) and refer the reader to e.g. [41] for more details.

2.2.1. *Trees.* Formally, a *binary tree* or simply *tree* with nodes in A is a function $t : P \to A$, where P is a non-empty prefix closed subset of $\{l, r\}^*$. The set $P \subseteq \{l, r\}^*$ is called the *domain* of t and is denoted by $\mathsf{dom}(t) \triangleq P$. Elements of P are called *nodes*. For a node $w \in P$ any node of the form wx for $x \in \{l, r\}$ is called a *child* of w. A tree is called *complete* if all nodes have either two children or no children. A *height* of a tree t is $\max\{|w| \mid w \in \mathsf{dom}(t)\}$. A tree t is *finite* if it is of a finite height, it is *infinite* if $\mathsf{dom}(t) = \{l, r\}^*$. The *frontier* of a tree t is $\mathsf{fr}(t) \triangleq \{w \in \mathsf{dom}(t) \mid \{wl, wr\} \cap P = \varnothing\}$. Elements of $\mathsf{fr}(t)$ are called *leaves*. Nodes from $\mathsf{dom}(t) \setminus \mathsf{fr}(t)$ are called *inner nodes*. The *outer frontier* of t is defined by $\mathsf{fr}^+(t) \triangleq \{wl, wr \mid w \in \mathsf{dom}(t)\} \setminus \mathsf{dom}(t)$. I.e. it consists of all the words $wi \notin \mathsf{dom}(t)$ such that $w \in \mathsf{dom}(t)$ and $i \in \{l, r\}$. Finally, set $\mathsf{dom}^+(t) \triangleq \mathsf{dom}(t) \cup \mathsf{fr}^+(t)$.

Let $T_\Sigma X$ denote the set of all complete trees $t : P \to \Sigma + X$ with inner nodes taking values in Σ and which have a finite number of leaves, all from the set X. Note that trees from $T_\Sigma X$ of height 0 can be thought of as elements of X. Hence, we may write $X \subseteq T_\Sigma X$. Moreover, trees of height 1 can be viewed as elements from $\Sigma \times X \times X$. Thus, $\Sigma \times X \times X \subseteq T_\Sigma X$. Additionally, any $f : X \to Y$ induces a map $T_\Sigma f : T_\Sigma X \to T_\Sigma Y$ which assigns to $t \in T_\Sigma X$ the tree obtained from t by replacing any occurrence of a leaf $x \in X$ with $f(x) \in Y$. This turns $T_\Sigma(-)$ into a Set-endofunctor. For two functions $f : X \to T_\Sigma Y$ and $g : Y \to T_\Sigma Z$ we may naturally define $g \cdot f : X \to T_\Sigma Z$ for which $(g \cdot f)(x)$ is a tree obtained from $f(x)$ with every occurence of a variable $y \in Y$ replaced with the tree $g(y) \in T_\Sigma Z$. It is a simple exercise to prove that $\cdot$ is associative. Moreover, if we denote the function $X \to T_\Sigma X; x \mapsto x$ by id then $\mathsf{id} \cdot f = f \cdot \mathsf{id}$. This follows from the fact that T_Σ is a monad and $g \cdot f$ is, in fact, the Kleisli composition for T_Σ (see Example 4.8 for details).

$$\Sigma = \{+, -\}$$
$$f : \{x\} \to T_\Sigma\{y_1, y_2\}; x \mapsto t$$
$$g' : \{y_1, y_2\} \to T_\Sigma Z; y_1 \mapsto t_1, y_2 \mapsto t_2$$
$$g'' : \{y_1, y_2\} \to T_\Sigma Z; y_1 \mapsto t_2, y_2 \mapsto t_1$$



[1]Our definition of R_i^ω and the one presented in e.g. [41] differ slightly on R_i with $\varepsilon \in R_i$. Indeed, in *loc. cit.*, $R_i^\omega = \{w_1 w_2 \ldots \mid w_i \in R_i \text{ and } w_i \neq \varepsilon\}$. This small difference does not change the formulation of the Kleene theorem. We choose our definition of $(-)^\omega$ because it can be viewed as the greatest fixpoint of a certain assignment. See the following sections for details.

Finally, $T^*_\Sigma X \subseteq T_\Sigma X$ and $T^\omega_\Sigma X \subseteq T_\Sigma X$ are sets of finite and infinite trees from $T_\Sigma X$ respectively. Note that trees in $T^\omega_\Sigma X$ have no leaves, hence $T^\omega_\Sigma X = T^\omega_\Sigma \varnothing$ for any set X.

2.2.2. *Büchi tree automata and their languages.* Let $\mathcal{Q} = (Q, \Sigma, \delta, \mathfrak{F})$ be a tree automaton. A *run* of the automaton $\mathcal{Q}$ on a finite tree $t \in T_\Sigma 1$ starting at the state $s \in Q$ is a map $\mathfrak{r} : \mathsf{dom}^+(t) \to Q$ such that $\mathfrak{r}(\varepsilon) = s$ and for any $x \in \mathsf{dom}(t) \setminus \mathsf{fr}(t)$ we have

$$(\mathfrak{r}(xl), \mathfrak{r}(xr)) \in \delta(\mathfrak{r}(x), t(x)).$$

We say that the run $\mathfrak{r}$ is *successful* if $\mathfrak{r}(w) \in \mathfrak{F}$ for any $w \in \mathsf{fr}^+(t)$ for the tree t. The set of finite trees recognized by a state s in $\mathcal{Q}$ is defined as the set of finite trees $t \in T^*_\Sigma 1$ for which there is a run in $\mathcal{Q}$ starting at s which accepts the tree t.

Finally, let $t \in T^\omega_\Sigma \varnothing$ be an infinite tree with nodes in Σ. An *infinite run* for t starting at $s \in Q$ is a map $\mathfrak{r} : \{l, r\}^* \to Q$ such that $\mathfrak{r}(\varepsilon) = s$ and:

$$(\mathfrak{r}(xl), \mathfrak{r}(xr)) \in \delta(\mathfrak{r}(x), t(x)) \text{ for all } x \in \{l, r\}^*.$$

The tree t is said to be *recognized by* the state s in $\mathcal{Q}$ if there is a run $\mathfrak{r}$ for t which start at s and for each path in t some final state occurs infinitely often [41].

2.2.3. *Rational tree languages.* Rational tree languages are analogues of rational languages for non-deterministic automata. Akin to the standard case, they are defined as sets of trees obtained from trees of height ≤ 1 by a sequence of applications of: finite union, composition and Kleene star closure. However, the non-sequential nature of trees requires us to consider composition of rational trees with more than one variable.

Formally, for any subset $T \subseteq X \to T_\Sigma X$ we define T^* by $T^* \triangleq \bigcup_n T^n$, where $T^0 = \{\mathsf{id}\}$ and $T^n = T^{n-1} \cup \{t' \cdot t \mid t' \in T^{n-1} \text{ and } t \in T\}$. For any natural number $n \in \{0, 1, \ldots\}$ we slightly abuse the notation and put $n \triangleq \{1, \ldots, n\}$ and define $\mathfrak{Rat}(1, n) \subseteq T_\Sigma n$ to be the smallest family of subsets which satisfies:

- $\varnothing \in \mathfrak{Rat}(1, n)$,
- $\{t\} \in \mathfrak{Rat}(1, n)$, where t is of height less than or equal to 1,
- if $T \in \mathfrak{Rat}(1, n)$ and $T_1, \ldots, T_m \in \mathfrak{Rat}(1, m)$ then:
$$\{[t_1, \ldots, t_n] \cdot t \mid t \in T, t_i \in T_i\} \in \mathfrak{Rat}(1, m),$$
- if $T \in \mathfrak{Rat}(1, n)$ then for any $i \in n$:
$$\{f(i) \mid f \in \{[t_1, \ldots, t_n] : n \to T_\Sigma n \mid t_i \in T\}^*\} \in \mathfrak{Rat}(1, n).$$

It is easy to check that if we extend the definition of $\mathfrak{Rat}$ and put

$$\mathfrak{Rat}(m, n) \triangleq m \to \mathfrak{Rat}(1, n)$$

then the last item in the above list implies that for any $T \in \mathfrak{Rat}(n, n)$ we have $T^* \in \mathfrak{Rat}(n, n)$.

Now for $T \subseteq n \to T^*_\Sigma n$ we define T^ω as the subset of $n \to T^\omega_\Sigma \varnothing$ consisting of common extensions of functions in $T^k \triangleq T^{k-1} \cdot T$ for any k with

$$T \cdot T' \triangleq \{t \cdot t' \mid t \in T, t' \in T'\}.$$

Finally, the ω-rational subset of trees is defined by [41]:

$$\omega\mathfrak{Rat} = \{T^\omega \cdot T' \mid T \in \mathfrak{Rat}(n, n) \text{ and } T' \in \mathfrak{Rat}(1, n)\}.$$

2.2.4. *Kleene theorems.* Let $\mathfrak{Reg}$ be the set of subsets of trees from $T_\Sigma^* 1$ for which there is an automaton accepting the given set of trees. Similarly, we define the set $\omega\mathfrak{Reg}$ of infinite trees accepted by the tree automata. In this case the Kleene theorems for regular and ω-regular input are respectively given by [41]:

$$\mathfrak{Reg} = \mathfrak{Rat}(1,1) \text{ and } \omega\mathfrak{Reg} = \omega\mathfrak{Rat}.$$

In Section 3 we will show that Kleene theorems for non-deterministic automata and tree automata are instances of a generic pair of theorems formulated on a categorical level.

2.3. **Algebras and coalgebras.** Let $F : \mathsf{C} \to \mathsf{C}$ be a functor. An *F-coalgebra* (*F-algebra*) is a morphism $\alpha : A \to FA$ (resp. $a : FA \to A$). The object A is called a *carrier* of the underlying F-(co)algebra. Given two coalgebras $\alpha : A \to FA$ and $\beta : B \to FB$ a morphism $h : A \to B$ is *homomorphism* from α to β provided that $\beta \circ h = F(h) \circ \alpha$. For two algebras $a : FA \to A$ and $b : FB \to B$ a morphism $h : A \to B$ is called *homomorphism* from a to b if $b \circ F(h) = h \circ a$. The category of all F-coalgebras (F-algebras) and homomorphisms between them is denoted by $\mathsf{CoAlg}(F)$ (resp. $\mathsf{Alg}(F)$). We say that a coalgebra $\zeta : Z \to FZ$ is *final* or *terminal* if for any F-coalgebra $\alpha : A \to FA$ there is a unique homomorphism $[[\alpha]] : A \to Z$ from α to ζ.

Example 2.1. Let Σ be a set of *labels*. *Labelled transition systems* (see e.g. [43]) can be viewed as coalgebras of the type $\mathcal{P}(\Sigma \times \mathcal{I}d) : \mathsf{Set} \to \mathsf{Set}$ [42]. Here, $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$ is the powerset functor which maps any X to the set $\mathcal{P}X = \{A \mid A \subseteq X\}$ and any $f : X \to Y$ to $\mathcal{P}f : \mathcal{P}X \to \mathcal{P}Y; A \mapsto f(A)$.

Non-deterministic automata as defined in Subsection 2.1 are modelled as coalgebras of the type $\mathcal{P}(\Sigma \times \mathcal{I}d + 1)$, where $1 = \{\checkmark\}$ (e.g. [26]). Indeed, any non-deterministic automaton $(Q, \Sigma, \delta, \mathfrak{F})$ is modelled by $\alpha : Q \to \mathcal{P}(\Sigma \times Q + 1)$ where:

$$\alpha(q) = \{(a, q') \mid q' \in \delta(a, q)\} \cup \chi(q),$$

where $\chi(q) = \begin{cases} \{\checkmark\} & q \in \mathfrak{F}, \\ \varnothing & \text{otherwise.} \end{cases}$ In a similar manner, we can model *tree automata* coalgebraically, i.e. as coalgebras of the type $Q \to \mathcal{P}(\Sigma \times Q \times Q + 1)$.

Example 2.2. *Fully probabilistic processes* [5] sometimes referred to as *fully probabilistic systems* [47] are modelled as $\mathcal{D}(\Sigma \times \mathcal{I}d)$-coalgebras [47]. Here, $\mathcal{D}$ denotes the subdistribution functor assigning to any set X the set $\{\mu : X \to [0,1] \mid \sum_x \mu(x) \leq 1\}$ of subdistributions with countable support and to any map $f : X \to Y$ the map $\mathcal{D}f : \mathcal{D}X \to \mathcal{D}Y; \mu \mapsto \mathcal{D}f(\mu)$ with

$$\mathcal{D}f(\mu)(y) = \sum \{\mu(x) \mid x \in X \text{ such that } f(x) = y\}.$$

2.4. **Monads.** For a general introduction to the theory of monads the reader is referred to e.g. [6, 36]. A *monad* on C is a triple (T, μ, η), where $T : \mathsf{C} \to \mathsf{C}$ is an endofunctor and $\mu : T^2 \implies T$, $\eta : \mathcal{I}d \implies T$ are two natural transformations for which the following diagrams commute:

$$\begin{tikzcd} T^3 \arrow[r, "\mu"] \arrow[d, "T\mu"'] & T^2 \arrow[d, "\mu"] \\ T^2 \arrow[r, "\mu"'] & T \end{tikzcd} \qquad \begin{tikzcd} T \arrow[r, "T\eta"] \arrow[d, "\eta_T"'] \arrow[dr, "\mathsf{id}"'] & T^2 \arrow[d, "\mu"] \\ T^2 \arrow[r, "\mu"'] & T \end{tikzcd}$$

The transformation μ is called *multiplication* and η *unit*.

Any monad $(T : \mathsf{C} \to \mathsf{C}, \mu, \eta)$ gives rise to the Kleisli category for T. To be more precise, the *Klesli category* $\mathcal{K}l(T)$ for T has the class of objects equal to the class of objects of C and for two objects X, Y in $\mathcal{K}l(T)$ we have $\mathcal{K}l(T)(X,Y) = \mathsf{C}(X, TY)$ with the composition $\cdot$ in $\mathcal{K}l(T)$ defined between two morphisms $f : X \to TY$ and $g : Y \to TZ$ by $g \cdot f := \mu_Z \circ T(g) \circ f$. Since most of the time we work with two categories at once, namely C and $\mathcal{K}l(T)$, morphisms in C will be denoted using standard arrows $\to$, whereas for morphisms in $\mathcal{K}l(T)$ we will use the symbol $\multimap\!\!\to$. Hence, $f : X \multimap\!\!\to Y = X \to TY$ and the composition is given by

$$X \overset{f}{\multimap\!\!\to} Y \overset{g}{\multimap\!\!\to} Z = X \xrightarrow{f} TY \xrightarrow{Tg} TTZ \xrightarrow{\mu_Z} TZ.$$

We define a functor $^\sharp : \mathsf{C} \to \mathcal{K}l(T)$ which sends each object $X \in \mathsf{C}$ to itself and each morphism $f : X \to Y$ in C to the morphism $f^\sharp : X \to TY; f^\sharp \triangleq \eta_Y \circ f$. Maps in $\mathcal{K}l(T)$ of the form $f^\sharp$ for some $f : X \to Y \in \mathsf{C}$ are referred to as *base morphisms*.

Every monad (T, μ, η) on a category C arises from the composition of a left and a right adjoint: $\mathsf{C} \leftrightarrows \mathcal{K}l(T)$, where the left adjoint is $^\sharp : \mathsf{C} \to \mathcal{K}l(T)$ and the right adjoint $U_T : \mathcal{K}l(T) \to \mathsf{C}$ is defined as follows: for any object $X \in \mathcal{K}l(T)$ (i.e. $X \in \mathsf{C}$) the object $U_T X$ is given by $U_T X := TX$ and for any morphism $f : X \to TY$ in $\mathcal{K}l(T)$ the morphism $U_T f : TX \to TY$ is given by $U_T f = \mu_Y \circ Tf$.

We say that a functor $F : \mathsf{C} \to \mathsf{C}$ *lifts to* a functor $\overline{F} : \mathcal{K}l(T) \to \mathcal{K}l(T)$ provided that the following diagram commutes:

$$\begin{tikzcd}
\mathcal{K}l(T) \arrow[r, "\overline{F}"] & \mathcal{K}l(T) \\
\mathsf{C} \arrow[u, "\sharp"] \arrow[r, "F"'] & \mathsf{C} \arrow[u, "\sharp"']
\end{tikzcd}$$

There is a one-to-one correspondence between liftings $\overline{F}$ and *distributive laws* $\lambda : FT \implies TF$ between the functor F and the monad T, i.e. natural transformations which satisfies extra conditions (see e.g. [30, 39] for details). Indeed, any lifting $\overline{F} : \mathcal{K}l(T) \to \mathcal{K}l(T)$ induces the transformation λ whose X-component $\lambda_X : FTX \to TFX$ is $\lambda_X = \overline{F}(\mathsf{id}_{TX} : TX \to TX)$ and any distributive law $\lambda : FT \implies TF$ gives rise to a lifting $\overline{F} : \mathcal{K}l(T) \to \mathcal{K}l(T)$ given by:

$$\overline{F}X = FX \text{ and } \overline{F}(X \xrightarrow{f} TY) = FX \xrightarrow{Ff} FTY \xrightarrow{\lambda_Y} TFY.$$

A monad (T, μ, η) on a cartesian closed category C is called *strong* if there is a natural transformation $t_{X,Y} : X \times TY \to T(X \times Y)$ called *tensorial strength* satisfying the strength laws listed in e.g. [34]. Existence of strength guarantees that for any object Σ the functor $\Sigma \times \mathcal{I}d : \mathsf{C} \to \mathsf{C}$ admits a lifting $\overline{\Sigma} : \mathcal{K}l(T) \to \mathcal{K}l(T)$ defined as follows. For any $X \in \mathcal{K}l(T)$ we put $\overline{\Sigma}X := \Sigma \times X$, and for any $f : X \multimap\!\!\to Y = X \to TY$ we define $\overline{\Sigma}f \triangleq t_{\Sigma,Y} \circ (id_\Sigma \times f)$. Existence of the transformation t is not a strong requirement. For instance all monads on Set are strong.

Example 2.3. The powerset endofunctor $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$, used in the definition of labelled transition systems, non-deterministic automata and tree automata, carries a monadic structure $(\mathcal{P}, \bigcup, \{-\})$ for which the multiplication and the unit are given by:

$$\bigcup : \mathcal{P}\mathcal{P}X \to \mathcal{P}X; S \mapsto \bigcup S, \qquad \{-\} : X \to \mathcal{P}X; x \mapsto \{x\}.$$

The Kleisli category $\mathcal{K}l(\mathcal{P})$ consists of sets as objects and morphisms given by the maps $f : X \multimap\!\!\!\rightarrow Y = X \to \mathcal{P}Y$ and $g : Y \multimap\!\!\!\rightarrow Z = Y \to \mathcal{P}Z$ with the composition $g \cdot f : X \multimap\!\!\!\rightarrow Z = X \to \mathcal{P}Z$ defined by

$$g \cdot f(x) = \{z \in Z \mid z \in \bigcup g(f(x))\}.$$

The identity morphisms $\mathsf{id} : X \multimap\!\!\!\rightarrow X = X \to \mathcal{P}X$ are given for any $x \in X$ by $\mathsf{id}(x) = \{x\}$. The Kleisli category for $\mathcal{P}$ is isomorphic to Rel - the category of sets as objects, and relations as morphisms. The X-component of the distributive law $\lambda : \Sigma \times \mathcal{P}X \to \mathcal{P}(\Sigma \times X)$ induced by strength of $\mathcal{P}$ is:

$$\lambda(a, X') = \{(a, x) \mid x \in X'\}.$$

Example 2.4. The subdistribution functor $\mathcal{D} : \mathsf{Set} \to \mathsf{Set}$ from Example 2.2 carries a monadic structure $(\mathcal{D}, \mu, \eta)$, where $\mu_X : \mathcal{D}\mathcal{D}X \to \mathcal{D}X$ is

$$\mu(\psi)(x) = \sum_{\phi \in \mathcal{D}X} \psi(\phi) \cdot \phi(x)$$

and $\eta_X : X \to \mathcal{D}X$ assigns to any x the Dirac delta distribution $\delta_x : X \to [0, 1]$.

Example 2.5. For any monoid $(M, \cdot, 1)$ the Set-functor $M \times \mathcal{I}d$ carries a monadic structure $(M \times \mathcal{I}d, m, e)$, where $m_X : M \times M \times X \to M \times X; (m, n, x) \mapsto (m \cdot n, x)$ and $e_X : X \to M \times X; x \mapsto (1, x)$.

From the perspective of this paper, the most imporant instance of the family of monads from Example 2.5 is the monad $(\Sigma^* \times \mathcal{I}d, m, e)$, where $(\Sigma^*, \cdot, \varepsilon)$ is the free monoid over Σ. The reason is that $\Sigma^* \times \mathcal{I}d$ is the free monad over the functor $\Sigma \times \mathcal{I}d$ and hence, since $\Sigma \times \mathcal{I}d$ lifts to the Kleisli category for any Set-based monad T (since all Set-based monads are strong), then so does $\Sigma^* \times \mathcal{I}d$ whose lifting is the free monad over the lifting of $\Sigma \times \mathcal{I}d$ [11]. In practice, this yields a monadic structure on $T(\Sigma^* \times \mathcal{I}d)$ for any monad T on the category of sets [11].

Example 2.6. If $T = \mathcal{P}$ then the Kleisli category for $\mathcal{P}(\Sigma^* \times \mathcal{I}d)$ has the composition given as follows [11]. For two morphisms $f : X \multimap\!\!\!\rightarrow Y = X \to \mathcal{P}(\Sigma^* \times Y)$ and $g : Y \multimap\!\!\!\rightarrow Z = Y \to \mathcal{P}(\Sigma^* \times Z)$ we have

$$g \cdot f(x) = \{(\sigma_1\sigma_2, z) \mid x \xrightarrow{\sigma_1}_f y \xrightarrow{\sigma_2}_g z \text{ for some } y \in Y\}.$$

The identity morphisms in this category are $\mathsf{id} : X \multimap\!\!\!\rightarrow X = X \to \mathcal{P}(\Sigma^* \times X)$ given by $\mathsf{id}(x) = \{(\varepsilon, x)\}$.

In a similar manner, using the remark above, we show that $\mathcal{D}(\Sigma^* \times \mathcal{I}d)$ carries a monadic structure.

2.5. **Coalgebras with internal moves and their type monads.** Coalgebras with internal moves were first introduced in the context of coalgebraic trace semantics as coalgebras of the type TF_ε for a monad T and an endofunctor F on C with F_ε defined by $F_\varepsilon \triangleq F + \mathcal{I}d$ [25, 45]. If we take $F = \Sigma \times \mathcal{I}d$ then we have $TF_\varepsilon = T(\Sigma \times \mathcal{I}d + \mathcal{I}d) \cong T(\Sigma_\varepsilon \times \mathcal{I}d)$, where $\Sigma_\varepsilon \triangleq \Sigma + \{\varepsilon\}$. In [11] we showed that given certain assumptions on T and F we may embed the functor TF_ε into the monad TF^*, where F^* is the free monad over F. In particular, if we apply this construction to $T = \mathcal{P}$ and $F = \Sigma \times \mathcal{I}d$ we obtain the monad $\mathcal{P}(\Sigma^* \times \mathcal{I}d)$ from Example 2.6. The construction of TF^* is revisited in this paper in Section 4. The trick of modelling the invisible steps via a monadic structure allows us *not* to specify the internal moves explicitly. Instead of considering TF_ε-coalgebras we consider T'-coalgebras for a monad T' on an arbitrary category.

The strategy of finding a suitable monad (for modelling the behaviour taken into consideration) will also be applied in this paper. Unfortunately, from the point of view of the infinite behaviour of coalgebras, considering systems of the type TF^* is not sufficient (see Section 3 for a discussion). Hence, in Section 4 we show how to obtain a monad suitable for modelling infinite behaviour. Intuitively, the new monad extends TF^* by adding an ingredient associated with the terminal F-coalgebra $\zeta : F^\omega \to FF^\omega$. The construction presented in Section 4 yields the monad $TF^\infty = T(F^* \oplus F^\omega)$ suitable to capture both: finite and infinite behaviour of systems. Below we give two examples of such monad.

Example 2.7. Although the monad $\mathcal{P}(\Sigma^* \times \mathcal{I}d)$ from Example 2.6 proves to be sufficient to model finite behaviours of non-deterministic automata (see [11, 13]), it will not be suitable to model their infinite behaviour (see Section 3 for details). Hence, we extend $\mathcal{P}(\Sigma^* \times \mathcal{I}d)$ and consider the following. Let Σ^ω be the set of all infinite sequences of elements from Σ. As it will be shown in sections to come, the functor $\mathcal{P}(\Sigma^* \times \mathcal{I}d + \Sigma^\omega)$ carries a monadic structure whose Kleisli composition is as follows. For $f : X \to \mathcal{P}(\Sigma^* \times Y + \Sigma^\omega)$ and $g : Y \to \mathcal{P}(\Sigma^* \times Z + \Sigma^\omega)$ the map $g \cdot f : X \multimap\!\!\to Z = X \to \mathcal{P}(\Sigma^* \times Z + \Sigma^\omega)$ satisfies:

$$x \xrightarrow{\sigma}_{g\cdot f} z \iff \exists y \text{ such that } x \xrightarrow{\sigma_1}_f y \text{ and } y \xrightarrow{\sigma_2}_g z, \text{ where } \sigma = \sigma_1\sigma_2 \in \Sigma^*,$$
$$x \downarrow_{g\cdot f} v \iff x \downarrow_f v \text{ or } x \xrightarrow{\sigma}_f y \text{ with } y \downarrow_g v' \text{ and } v = \sigma v' \in \Sigma^\omega.$$

In the above we write $x \xrightarrow{\sigma}_f y$ whenever $(\sigma, y) \in f(x)$ and $x \downarrow_f v$ if $v \in f(x)$ for $\sigma \in \Sigma^*$, $v \in \Sigma^\omega$. The identity morphisms in this category are the same as in the Kleisli category for the monad $\mathcal{P}(\Sigma^* \times \mathcal{I}d)$. The monadic structure of $\mathcal{P}(\Sigma^* \times \mathcal{I}d + \Sigma^\omega)$ arises as a consequence of a general construction of monads modelling (in)finite behaviour described in detail in Section 4.

Example 2.8. If we move from non-deterministic automata towards tree automata we have to find a suitable monadic setting to talk about their (in)finite behaviour. It turns out that a good candidate for this monad can be built from the ingredients already presented in this paper. Indeed, if we take the powerset monad and the monad T_Σ from Subsection 2.2.1, then their composition $\mathcal{P}T_\Sigma$ carries a monadic structure[2]. The formula for the composition in the Kleisli category for the monad $\mathcal{P}T_\Sigma$ is given for $f : X \to \mathcal{P}T_\Sigma Y$ and $g : Y \to \mathcal{P}T_\Sigma Z$ by $g\cdot f : X \multimap\!\!\to Z = X \to \mathcal{P}T_\Sigma Z$ with $g \cdot f(x)$ being a set of trees obtained from trees in $f(x) \subseteq T_\Sigma Y$ by replacing any occurence of the leaf $y \in Y$ with a tree from $g(y) \subseteq T_\Sigma Z$. As will be witnessed in Section 4, this monad and the monad $\mathcal{P}(\Sigma^* \times \mathcal{I}d + \Sigma^\omega)$ arise from the same categorical construction.

The list of examples of monads used in the paper will be extended in the upcoming sections.

2.6. **Categorical order enrichment.** Our main ingredients for defining (in)finite behaviours of automata will turn out to be two fixpoint operators: $(-)^*$ and $(-)^\omega$. In order to establish them on a categorical level we require the category under consideration to be suitably order enriched. A category is said to be *order enriched*,

[2]The proof of this claim can be found in Section 4. See Example 4.8 for details.

or simply *ordered*, if each hom-set is a poset with the order preserved by the composition. It is $\vee$-*ordered* if all hom-posets admit arbitrary finite (possibly empty) suprema. Note that, given such suprema exist, the composition in C does not have to distribute over them in general. We call a category *left distributive* (or *LD* in short) if $g \cdot (\bigvee_{i \in I} f_i) = \bigvee_{i \in I} g \cdot f_i$ for any finite set I. We define *right distributivity* analogously. In this paper we come across many left distributive categories that do not necessarily satisfy right distributivity. Still, however, all examples of Kleisli categories taken into consideration satisfy its weaker form. To be more precise, we say that the Kleisli category $\mathcal{K}l(T)$ for a monad T on C is *right distributive w.r.t. base morphisms* provided that $(\bigvee_{i \in I} f_i) \cdot j^\sharp = \bigvee_{i \in I} f_i \cdot j^\sharp$ for any $f_i : Y \multimap Z = Y \to TZ \in \mathcal{K}l(T)(Y, Z)$, any $j : X \to Y \in \mathsf{C}(X, Y)$ and any finite set I. We say that an order enriched category is $\omega\mathsf{Cpo}$-enriched if any countable ascending chain of morphisms $f_1 \leq f_2 \leq \ldots$ with common domain and codomain admits a supremum which is preserved by the morphism composition. Finally, in an ordered category with finite coproducts we say that *cotupling preserves order* if $[f_1, f_2] \leq [g_1, g_2] \iff f_1 \leq g_1$ and $f_2 \leq g_2$ for any f_i, g_i with suitable domains and codomains.

$$-\boxed{f}-\circledvee-\boxed{g}- \;=\; -\boxed{f \vee g}-$$

Remark 2.9. Right distributivity w.r.t. the base morphisms and cotupling order preservation are properties we often get as a consequence of other general assumptions. Indeed, any Set-based monad T whose order enrichment of the Kleisli category is a consequence of an order on TY for any Y satisfies these condition. To be more precise let TY be a poset for any Y and consider the ordering on $\mathcal{K}l(T)(X, Y) = \mathsf{Set}(X, TY)$ given by $f \leq g \iff f(x) \leq g(x)$ for any $x \in X$. If this hom-set ordering is compatible with the Kleisli composition (i.e. if it yields an order enriched category) then we say that the order enrichement of $\mathcal{K}l(T)$ is *pointwise induced*. In this case the Kleisli composition over any suprema or infima that exist is right distributive w.r.t. morphisms of the form $j^\sharp = \eta_Y \circ j : X \multimap Y = X \to TY$ for any set map $j : X \to Y$. A similar argument applies to cotupling order preservation.

Example 2.10. The next section of this paper focuses on three categories, namely: $\mathcal{K}l(\mathcal{P}(\Sigma^* \times \mathcal{I}d))$, $\mathcal{K}l(\mathcal{P}(\Sigma^* \times \mathcal{I}d + \Sigma^\omega))$ and $\mathcal{K}l(\mathcal{P}T_\Sigma)$. These categories are order-enriched with the hom-set ordering given by $f \leq g \iff f(x) \subseteq g(x)$ for any x. The base morphisms of the first two examples are of the form

$$X \to \mathcal{P}(\{\varepsilon\} \times Y); x \mapsto \{(\varepsilon, j(x))\}$$

for a set map $j : X \to Y$. The base morphisms of the third example are given by $X \to \mathcal{P}T_\Sigma Y; x \mapsto \{j(x)\}$. We leave it as an exercise to the reader to verify that all these examples satisfy the following conditions: the order enrichment is pointwise induced; they are $\omega\mathsf{Cpo}$-enriched; their hom-sets are complete lattices; they are left distributive[3]. These conditions play a central role in defining (in)finite behaviours on a coalgebraic level. We will elaborate more on them in Section 5.

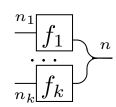
2.7. **Lawvere theories.** The primary interest of the theory of automata and formal languages focuses on automata over a *finite* state space. Hence, since we are interested in systems with internal moves (i.e. coalgebras $X \to TX$ for a monad T), without loss of generality we may focus our attention on coalgebras of the form

[3]We refer the reader to [11] for a proof that $\mathcal{K}l(\mathcal{P}(\Sigma^* \times \mathcal{I}d))$ satisfies these conditions.

$n \to Tn$, where $n = \{1, \ldots, n\}$ with $n = 0, 1, \ldots$. These morphisms are endomorphisms in a full subcategory of the Kleisli category for T, we will later refer to as (Lawvere) theory. Restricting the scope to this category instead of considering the whole Kleisli category for a given monad plays an important role in Kleene theorems characterizing regular and ω-regular behaviour (see e.g. [27, 41]).

Formally, a *Lawvere theory*, or simply *theory*, is a category whose objects are natural numbers $n \geq 0$ such that each n is an n-fold coproduct of 1. The definition used here is dual to the classical notion [35] and can be found in e.g. [18–20]. The reason why we use our version of the definition is the following: we want the connection between Lawvere theories and Kleisli categories for Set-based monads to be as direct as possible. Indeed, in our case, any monad T on Set induces a theory $\mathbb{T}$ associated with it by restricting the Kleisli category $\mathcal{K}l(T)$ to objects n for any $n \geq 0$. Conversely, for any theory $\mathbb{T}$ there is a Set based monad the theory is associated with (see e.g. [28] for details). This remark also motivates us to use the notation introduced before and denote morphisms from a theory by $-\!\!\circ\!\!\!\rightarrow$.

For any element $i \in n$ let $i_n : 1 -\!\!\circ\!\!\!\rightarrow n$ denote the i-th coproduct injection and $[f_1, \ldots, f_k] : n_1 + \ldots + n_k -\!\!\circ\!\!\!\rightarrow n$ the cotuple of $\{f_l : n_l -\!\!\circ\!\!\!\rightarrow n\}_l$ depicted in the diagram on the right. The coprojection morphism $n_i -\!\!\circ\!\!\!\rightarrow n_1 + \ldots + n_k$ into the i-th component of the coproduct will be denoted by $\mathsf{in}^{n_i}_{n_1+\ldots+n_k}$. Any morphism $k -\!\!\circ\!\!\!\rightarrow n$ of the form $[(i^1)_n, \ldots, (i^k)_n] : k -\!\!\circ\!\!\!\rightarrow n$ for $i^j \in n$ is called *base morphism* or *base map*. If $\mathbb{T}$ is associated with a monad T then the base morphisms in $\mathbb{T}$ are exactly given by $f^\sharp : m -\!\!\circ\!\!\!\rightarrow n = m \to Tn$ for Set-maps $f : m \to n$. Finally, let $! : n -\!\!\circ\!\!\!\rightarrow 1$ be defined by $! \triangleq [1_1, 1_1, \ldots, 1_1]$. We say that a theory $\mathbb{T}'$ is a *subtheory* of $\mathbb{T}$ if there is a faithful functor $\mathbb{T}' \to \mathbb{T}$ which maps any object n onto itself.



Example 2.11. By LTS, LTS^ω and TTS^ω we denote the theories associated with the monads $\mathcal{P}(\Sigma^* \times \mathcal{I}d)$, $\mathcal{P}(\Sigma^* \times \mathcal{I}d + \Sigma^\omega)$ and $\mathcal{P}T_\Sigma$ respectively.

3. Non-deterministic (tree) automata, coalgebraically

The purpose of this section is to give motivations for the abstract theory presented in the remainder of the paper. In the first part of this section we focus on finite non-deterministic (Büchi) automata and their (in)finite behaviour from the perspective of the categories $\mathcal{K}l(\mathcal{P}(\Sigma^* \times \mathcal{I}d))$ and $\mathcal{K}l(\mathcal{P}(\Sigma^* \times \mathcal{I}d + \Sigma^\omega))$. Afterwards, we deal with tree automata and their behaviour. Finally, we give a categorical perspective on Kleene theorems for automata taken into consideration.

3.1. Non-deterministic automata.

Without any loss of generality we may only consider automata over the state space $n = \{1, \ldots, n\}$ for some natural number n. As mentioned in Example 2.1 any non-deterministic automaton $(n, \Sigma, \delta, \mathfrak{F})$ may be modelled as a $\mathcal{P}(\Sigma \times \mathcal{I}d + 1)-$coalgebra $n \to \mathcal{P}(\Sigma \times n + 1)$ [42]. However, as it has been already noted in [50], from the point of view of infinite behaviour with BAC it is more useful to extract the information about the final states of the automaton and not to encode it into the transition map as above. Instead, given an automaton $(n, \Sigma, \delta, \mathfrak{F})$ we encode it as a pair $(\alpha, \mathfrak{F})$ where $\alpha : n \to \mathcal{P}(\Sigma \times n)$ is defined by $\alpha(i) = \{(a, j) \mid j \in \delta(a, i)\}$ and consider the map:

$$\mathfrak{f}_{\mathfrak{F}} : n \to \mathcal{P}(\{\varepsilon\} \times n); i \mapsto \begin{cases} \{(\varepsilon, i)\} & \text{if } i \in \mathfrak{F}, \\ \varnothing & \text{otherwise.} \end{cases}$$

Note that by extending the codomain of α and $\mathfrak{f}_{\mathfrak{F}}$ both maps can be viewed as endomorphisms in LTS and LTS^ω. The purpose of $\mathfrak{f}_{\mathfrak{F}}$ is to encode the set of accepting states with an endomorphism in the same Kleisli category in which the transition α is an endomorphism. Now, we have all the necessary ingredients to revisit finite and infinite behaviour (with BAC) of non-deterministic automata from the perspective of the theories LTS and LTS^ω.

3.1.1. *Finite behaviour.* Consider $\alpha^* : n \multimap n$ to be an endomorphism in LTS (or LTS^ω) given by $\alpha^* = \mu x.(\mathsf{id} \vee x \cdot \alpha) = \bigvee_{n\in\omega} \alpha^n$, where the order is as in Example 2.11. We have [11]:

$$\alpha^*(i) = \{(\sigma, j) \mid i \stackrel{\sigma}{\Longrightarrow} j\},$$

where $\stackrel{\sigma}{\Longrightarrow} \triangleq (\stackrel{\varepsilon}{\to})^* \circ \stackrel{a_1}{\to} \circ (\stackrel{\varepsilon}{\to})^* \circ \ldots (\stackrel{\varepsilon}{\to})^* \circ \stackrel{a_n}{\to} (\stackrel{\varepsilon}{\to})^*$ for $\sigma = a_1 \ldots a_n$, $a_i \in \Sigma$ and $\stackrel{\varepsilon}{\Longrightarrow} \triangleq (\stackrel{\varepsilon}{\to})^*$. Let us observe that the theory morphism $! : n \multimap 1$ is explicitly given in the case of theories LTS and LTS^ω by $!(i) = \{(\varepsilon, 1)\}$ for any $i \in n$. Finally, consider the morphism $! \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \alpha^* : n \multimap 1$ in LTS (or LTS^ω)) which is:

$$! \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \alpha^*(i) = \{(\sigma, 1) \mid \sigma \in \Sigma^* \text{ such that } i \stackrel{\sigma}{\Longrightarrow} j \text{ and } j \in \mathfrak{F}\}.$$

Since $\mathcal{P}(\Sigma^* \times 1) \cong \mathcal{P}(\Sigma^*)$, the set $! \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \alpha^*(i)$ represents the set of all finite words accepted by the state i in the automaton $(n, \Sigma, \delta, \mathfrak{F})$.

3.1.2. *Infinite behaviour.* Note that the hom-posets of theories LTS and LTS^ω are complete lattice orders and, hence (by the Tarski-Knaster theorem), come equipped with an operator which assigns to any endomorphism $\beta : n \multimap n$ the morphism $\beta^\omega : n \multimap 0$ defined as the greatest fixpoint of the assignment $x \mapsto x \cdot \beta$. For α the map $\alpha^\omega : n \multimap \mathcal{P}(\Sigma^* \times \varnothing) = \{\varnothing\}$ in LTS is given by $\alpha^\omega(i) = \varnothing$. However, if we compute $\alpha^\omega : n \multimap 0 = n \to \mathcal{P}(\Sigma^* \times \varnothing + \Sigma^\omega) = n \to \mathcal{P}(\Sigma^\omega)$ in LTS^ω the result will be different. Indeed, we have the following.

Proposition 3.1. *Let $\beta : n \to \mathcal{P}(\Sigma^* \times n) \hookrightarrow \mathcal{P}(\Sigma^* \times X + \Sigma^\omega)$. Then $\beta^\omega : n \multimap 0 = n \to \mathcal{P}(\Sigma^\omega)$ in LTS^ω is given by:*

$$(\diamond) \qquad \beta^\omega(i) = \bigcup \{|\sigma_1, \sigma_2, \ldots| \subseteq \Sigma^\omega \mid i \stackrel{\sigma_1}{\to}_\beta i_1 \stackrel{\sigma_2}{\to}_\beta i_2 \ldots \text{ for some } i_k \in n \text{ and } \sigma_k \in \Sigma^*\},$$

where $|-| : (\Sigma^*)^\omega \to \mathcal{P}(\Sigma^\omega)$ assigns to any sequence $\sigma_1, \sigma_2, \ldots$ of words over Σ the set

$$|\sigma_1, \sigma_2, \ldots| \triangleq \begin{cases} \{\sigma_1\sigma_2 \ldots\} \times \Sigma^\omega & \text{if } \sigma_1\sigma_2 \ldots \text{ is a finite word,} \\ \{\sigma_1\sigma_2 \ldots\} & \text{otherwise.} \end{cases}$$

Proof. At first let us note that we may assume $\beta = \beta^* \cdot \beta$. This is a consequence of the fact that $\beta^\omega = (\beta^* \cdot \beta)^\omega$ proven in Lemma 5.3. Restated, this condition means that for any $i, j, k \in n$ we have:

$$(\square) \qquad i \stackrel{\sigma_1}{\to}_\beta j \stackrel{\sigma_2}{\to}_\beta k \implies i \stackrel{\sigma_1\sigma_2}{\to}_\beta k.$$

Let $\beta^o : n \multimap 0 = n \to \mathcal{P}(\Sigma^\omega)$ be a map whose value $\beta^o(i)$ is given in terms of the right hand side of the equality $(\diamond)$. Observe that this map satisfies $\beta^o = \beta^o \cdot \beta$ in LTS^ω. This follows directly from the definition of β^o and the formula for the composition in $\mathcal{K}l(\mathcal{P}(\Sigma^* \times \mathcal{I}d + \Sigma^\omega))$. Thus, $\beta^o \leq \beta^\omega$. Now, by contradiction, if $\beta^o < \beta^\omega$ then there is i and $v \in \Sigma^\omega$ such that $v \in \beta^\omega(i)$ and $v \notin \beta^o(i)$. Hence, in particular, this means that there is an infinite sequence of transitions $i \stackrel{\sigma_1}{\to} i_1 \stackrel{\sigma_2}{\to} i_2 \ldots$ in β which starts at i. If there was no such sequence, this would mean that $\beta^o(i) = \varnothing = \beta^\omega(i)$ which cannot hold. Since $\beta^\omega = \beta^\omega \cdot \beta$, there is a state i_1 and

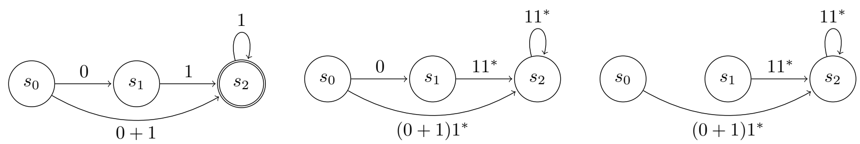


FIGURE 4. An automaton $(\alpha, \mathfrak{F})$ and the maps α^+ and $\mathfrak{f}_{\mathfrak{F}} \cdot \alpha^+$.

$\sigma_1 \in \Sigma^*$, $v_1 \in \Sigma^\omega$ such that $i \xrightarrow{\sigma_1}_\beta i_1$ and $v_1 \in \beta^\omega(i_1)$ with $v = \sigma_1 v_1$. Note that we may assume $\sigma_1 \neq \varepsilon$ as there has to be a prefix $\sigma_1 \neq \varepsilon$ of $v = \sigma_1 v_1$ with $i \xrightarrow{\sigma_1} i_1$ for some state i_1 with $v_1 \in \beta^\omega(i_1)$. If it was otherwise, then by ($\square$) we would have an infinite sequence $i \xrightarrow{\varepsilon} i_1 \xrightarrow{\varepsilon} i_2 \xrightarrow{\varepsilon} \ldots$ yielding $\beta^\omega(i) = \Sigma^\omega = \beta^o(i)$ which contradicts our assumptions. Hence, if $i \xrightarrow{\sigma_1} i_1$ for $\sigma_1 \neq \varepsilon$ and $v_1 \in \beta^\omega(i_1)$ then we also have $v_1 \notin \beta^o(i_1)$. By inductively repeating this argument we get a sequence $i \xrightarrow{\sigma_1} i_1 \xrightarrow{\sigma_2} i_2 \ldots$ in β such that $\sigma_k \neq \varepsilon$ and $v = \sigma_1\sigma_2 \ldots$. Thus, by the definition of β^o we also get $v \in \beta^o(i)$ which is a contradiction. $\square$

3.1.3. *Büchi acceptance condition.* Before we spell out the recipe on how to extract ω-language of any state in the automaton $(\alpha, \mathfrak{F})$ in terms of $(-)^*$, $(-)^\omega$ and the composition in LTS^ω, we need one last ingredient. Let us define $\alpha^+ \triangleq \alpha^* \cdot \alpha$ and note

$$\alpha^+(i) = \{(\sigma, j) \mid i \xrightarrow{a_1} i_1 \ldots \xrightarrow{a_k} i_k \text{ in } \alpha \text{ and } \sigma = a_1 \ldots a_k \text{ for } k \geq 1\}.$$

Hence, $\mathfrak{f}_{\mathfrak{F}} \cdot \alpha^+$ viewed as an endomorphism in LTS^ω is given by $\mathfrak{f}_{\mathfrak{F}} \cdot \alpha^+ : n \multimap n = n \to \mathcal{P}(\Sigma^* \times n + \Sigma^\omega)$ where:

$$\mathfrak{f}_{\mathfrak{F}} \cdot \alpha^+(i) = \{(\sigma, j) \mid i \xrightarrow{\sigma} j \text{ in } \alpha^+ \text{ and } j \in \mathfrak{F}\}.$$

Finally, consider the following map in LTS^ω:

$$(\mathfrak{f}_{\mathfrak{F}} \cdot \alpha^+)^\omega : n \multimap 0 = n \to \mathcal{P}(\Sigma^\omega).$$

By Proposition 3.1, the map $(\mathfrak{f}_{\mathfrak{F}} \cdot \alpha^+)^\omega$ satisfies:

$$(\mathfrak{f}_{\mathfrak{F}} \cdot \alpha^+)^\omega(i) = \text{ the } \omega\text{-language of } i \text{ in the Büchi automaton represented by } (\alpha, \mathfrak{F}).$$

The above statement suggests a general approach towards modelling (ω-)behaviours of abstract (coalgebraic) automata which we will develop in the sections to come.

3.2. **Tree automata.** Let us now focus our attention on tree automata and their behaviour. Just like in the previous subsection we may consider automata over the state space n. Moreover, as before, we also encode any tree automaton $(n, \Sigma, \delta, \mathfrak{F})$ as a pair $(\alpha : n \to \mathcal{P}(\Sigma \times n \times n), \mathfrak{F})$. Since $\mathcal{P}(\Sigma \times n \times n) \subseteq \mathcal{P}T_\Sigma n$, the transition map α can be viewed as $\alpha : n \to \mathcal{P}T_\Sigma n$ (i.e. as an endomorphism in the Kleisli category for the monad $\mathcal{P}T_\Sigma$ or, equivalently, as an endomorphism in the theory TTS^ω). The hom-sets of the Kleisli category for $\mathcal{P}T_\Sigma$ and its full subcategory TTS^ω admit ordering in which we can define β^*, β^+ and β^ω for any $\beta : n \multimap n$ as in the previous subsection. Not surpisingly, if we now compute $! \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \alpha^*$ and $(\mathfrak{f}_{\mathfrak{F}} \cdot \alpha^+)^\omega$ in TTS^ω we exactly get the following[4].

[4]The proof of Proposition 3.2 is intensionally omitted as it goes along the lines of the series of statements made in Subsection 2.1 for non-deterministic automata.

Proposition 3.2. *For any $i \in n$ we have:*

$$\begin{aligned} ! \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \alpha^*(i) = &\text{ the set of finite trees recognized by } i \text{ in } (n, \Sigma, \delta, \mathfrak{F}),\\ (\mathfrak{f}_{\mathfrak{F}} \cdot \alpha^+)^\omega(i) = &\text{ the set of inifinite trees recognized by } i \text{ in } (n, \Sigma, \delta, \mathfrak{F}). \end{aligned}$$

3.3. **Kleene theorems, categorically.** The purpose of this subsection is to restate classical Kleene theorems from Subsection 2.1 on the categorical level for LTS^ω and TTS^ω. Before we do this let us elaborate more on why we choose our setting to be systems whose type is a monad.

Remark 3.3 (**Why systems with internal moves?**)**.** As the examples of non-deterministic (tree) automata studied in the previous subsection do not admit silent moves, the reader may get an impression that the need for categorical modelling of infinite behaviour for systems with silent steps is *not* sufficiently justified. To add to this, although ε-moves are a standard feature of automata whenever it comes to their finitary languages, invisible moves rarely occur in practice in the classical literature on the *infinite* behaviour (with BAC) (see e.g. [41]). However, as already mentioned in the introduction, incorporation of silent moves should be viewed as a by-product of our paper's framework, not its main purpose. The main aim is to build a simple bridge between syntax and semantics of regular and ω-regular behaviours in the form of generic Kleene theorems. Once we embed our systems into systems whose type is a monad T, the syntax arises from the algebraicity of T and the semantics is provided by automata whose transition maps are certain T-coalgebras. This also allows us to abstract away from several "unnecessary" details and focus on core properties.

Hopefully, what we present below, and what is further generalized in Section 5, demonstrates that access to the language of a Kleisli category of a monad justifies the monadic extension of the setting, as it makes it possible to formulate generic results connecting syntax and semantics of (in)finite behaviours and provide their simple proofs which, in our opinion, would be tedious without such an extension.

As witnessed in Subsection 2.1, Kleene theorems for tree automata were slightly more involved than their classical counterparts for non-deterministic automata. The reason for this is simple: non-deterministic automata accept sequential data types. Whenever we deal with non-sequential data, e.g. trees, the set of (ω-)regular languages is expected to be closed under a more complex type of composition, i.e. the composition of regular languages with multiple variables [23, 41]. Hence, if we aim at categorical statements generalizing theory from Subsection 2.1 then we should expect the slighlty more involved formulation to be our point of reference. Hence, we start with presenting a categorical perspective of Kleene theorems for tree automata first.

3.3.1. *Tree automata.* The Kleene statements from Subsection 2.2.4 are equivalent to the following proposition.

Proposition 3.4. *Let $\mathfrak{Rat}$ be the smallest subtheory of TTS^ω such that:*

(a) it contains all maps of the form $n \to \mathcal{P}(\Sigma_\varepsilon \times n \times n) \hookrightarrow \mathcal{P}T_\Sigma n$,
(b) is closed under finite suprema,
(c) its endomorphisms are closed under $(-)^*$.

Then $\mathfrak{Rat}(1,1) = \mathfrak{Reg}$. Moreover, the set $\omega\mathfrak{Reg}$ of ω-regular languages for tree automata satisfies $\{r^\omega \cdot s \mid r \in \mathfrak{Rat}(n,n), s \in \mathfrak{Rat}(1,n)\} = \omega\mathfrak{Reg}$.

3.3.2. *Non-deterministic automata.* The formulation of Proposition 3.4 allows us to instantiate it for non-deterministic automata. A simple verification proves that the following holds.

Proposition 3.5. *Let $\mathfrak{Rat}$ be the smallest subtheory of LTS^ω such that:*

(a) it contains all maps of the form $n \to \mathcal{P}(\Sigma_\varepsilon \times n) \hookrightarrow \mathcal{P}(\Sigma^* \times n + \Sigma^\omega)$,
(b) it is closed under finite suprema,
(c) its endomorphisms are closed under $(-)^*$.

Then the hom-set $\mathfrak{Rat}(1,1)$ is given by:

$$\{r : 1 \to \mathcal{P}(\Sigma^* \times 1) \subseteq \mathcal{P}(\Sigma^* \times 1 + \Sigma^\omega) \mid r(1) = R \times \{1\} \text{ where } R \subseteq \Sigma^* \text{ is regular}\}.$$

Additionally, the set $\omega\mathfrak{Reg} \triangleq \{r : 1 \multimap 0 = 1 \to \mathcal{P}(\Sigma^\omega) \mid r(1) \text{ is } \omega\text{-regular}\}$ satisfies:

$$\{r^\omega \cdot s : 1 \multimap 0 = 1 \to \mathcal{P}(\Sigma^\omega) \mid r \in \mathfrak{Rat}(n,n), s \in \mathfrak{Rat}(1,n)\} = \omega\mathfrak{Reg}.$$

3.4. **Beyond non-deterministic (tree) automata.** There are variants of non-deterministic (Büchi) automata that accept other types of input (e.g. arbitrary finitely-branching trees, see e.g. [50]). In general, given a functor $F : \mathsf{Set} \to \mathsf{Set}$ we define a non-deterministic (Büchi) F-automaton as a pair $(\alpha, \mathfrak{F})$, where $\alpha : n \to \mathcal{P}Fn$ and $\mathfrak{F} \subseteq n$. A natural question that arises is the following: are we able to build a general categorical setting in which we can reason about the (in)finite behaviour of systems for arbitrary non-deterministic Büchi F-automata (or even more generally, for systems of the type TF for a monad T)? If so, then is it possible to generalize the Kleene theorem for ($\omega-$)regular languages to a coalgebraic level? We will answer these questions positively in the next sections.

4. Monads for (in)finite behaviour

Given a monad T and an endofunctor F on a common category, the purpose of this section is to provide a construction of a monad TF^∞ into which the functor TF embeds, that will prove itself sufficient to model the combination of finite and infinite behaviour (akin to the monad $\mathcal{P}(\Sigma^* \times \mathcal{I}d + \Sigma^\omega)$ for the functor $\mathcal{P}(\Sigma \times \mathcal{I}d)$, or $\mathcal{P}T_\Sigma$ for $\mathcal{P}(\Sigma \times \mathcal{I}d^2)$).

At first we list all assumptions required in the remainder of this section. Later, in Subsection 4.3, we revisit the construction of the monad TF^* from [11]. Finally, we give a description of TF^∞.

4.1. **Assumptions.** Let C be a category which admits binary coproducts. We denote the coproduct operator by $+$ and the coprojection into the first and the second component of a coproduct by inl and inr respectively. Moreover, let $F : \mathsf{C} \to \mathsf{C}$ be a functor. In what follows, in this section we additionally assume:

(A) (T, μ, η) is a monad on C and $F : \mathsf{C} \to \mathsf{C}$ lifts to $\mathcal{K}l(T)$ via a distributive law $\lambda : FT \implies TF$,
(B) there is an initial $F(-) + X$-algebra for any object X and a terminal F-coalgebra $\zeta : F^\omega \to FF^\omega$,
(C) the category $\mathsf{Alg}(F)$ of F-algebras admits binary coproducts (with the coproduct operator denoted by $\oplus$).

4.2. **Preliminaries.** The initial $F(-)+X$-algebra $i_X : FF^*X + X \to F^*X$ yields the free F-algebra over X given by $i_X \circ \mathsf{inl} : FF^*X \to F^*X$. Hence, by our assumptions we have an adjoint situation $\mathsf{C} \rightleftarrows \mathsf{Alg}(F)$, where the left adjoint is the free algebra functor which assigns to any object X the free algebra $i_X \circ \mathsf{inl}$ over it. The right adjoint is the forgetful functor which assigns to any F-algebra its carrier and is the identity on morphisms. The adjunction yields the monad $F^* : \mathsf{C} \to \mathsf{C}$ which assigns to any object X the carrier of the free F-algebra over X.

Example 4.1. For any set Σ and X the initial $\Sigma \times \mathcal{I}d + X$-algebra is given by the morphism $i_X : \Sigma \times \Sigma^* \times X + X \to \Sigma^* \times X$, where

$$i_X(a, (\sigma, x)) = (a\sigma, x) \text{ and } i_X(x) = (\varepsilon, x).$$

4.2.1. *Bloom algebras and the monad F^∞.* The purpose of this subsection is to recall basic definitions and properties of Bloom F-algebras [1] whose free algebras yield a monad F^∞ on C which extends the functor F. This will allow us to embed systems of the type $X \to TFX$ to systems of the type $X \to TF^\infty X$ and discuss their (in)finite behaviour in the latter context.

A pair $(a : FA \to A, (-)^\dagger)$ is called *Bloom F-algebra* provided that any F-coalgebra $e : X \to FX$ yields the map $e^\dagger : X \to A$ which satisfies:

$$\begin{array}{ccc} X & \xrightarrow{e^\dagger} & A \\ e\downarrow & & \uparrow a \\ FX & \xrightarrow[Fe^\dagger]{} & FA \end{array} \quad \text{and} \quad \begin{array}{ccc} X & \xrightarrow{h} & Y \\ e\downarrow & & \downarrow f \\ FX & \xrightarrow[Fh]{} & FY \end{array} \text{ implies } \begin{array}{ccc} X & \xrightarrow{h} & Y \\ & e^\dagger\searrow \quad \swarrow f^\dagger & \\ & A & \end{array}$$

A *homomorphism* from a Bloom algebra $(a : FA \to A, (-)^\dagger)$ to a Bloom algebra $(b : FB \to B, (-)^\ddagger)$ is a map $h : A \to B$ which is an F-algebra homomorphism from a to b, which additionally preserves the dagger, i.e. $e^\dagger \circ h = e^\ddagger$. The category of Bloom algebras and homomorphisms between them is denoted by $\mathsf{Alg}_B(F)$. We have the following theorem.

Theorem 4.2. *[1] The pair $(\zeta^{-1} : FF^\omega \to F^\omega, [[-]])$, where $[[-]]$ assigns to $e : X \to FX$ the unique coalgebra homomorphism $[[e]] : X \to F^\omega$ between e and ζ, is an initial object in $\mathsf{Alg}_B(F)$. Moreover, the F-algebra coproduct*

$$(i_X \circ \mathsf{inl} : FF^*X \to F^*X) \oplus (\zeta^{-1} : FF^\omega \to F^\omega)$$

is the free Bloom algebra over X.

Remark 4.3. Let $F^\infty : \mathsf{C} \to \mathsf{C}$ be defined as the composition of the left and right adjoints $\mathsf{C} \rightleftarrows \mathsf{Alg}_B(F)$ respectively, where the left adjoint is the free Bloom algebra functor and the right adjoint is the forgetful functor. The functor F^∞ carries a monadic structure which extends F^*. Indeed, by Th. 4.2, the monad F^* is a submonad of F^∞ (via the transformation induced by the coprojection into the first component of $i_X \circ \mathsf{inl} \oplus \zeta^{-1}$ in $\mathsf{Alg}(F)$). The formula for the free Bloom algebra from the above theorem indicates that F^∞ is a natural extension of F^* encompassing infinite behaviours of the final F-coalgebra. By abusing the notation slightly, we can write

$$F^\infty = F^* \oplus F^\omega.$$

The functor F_ε is a subfunctor of F^* [11, Lemma 4.12] and hence, by the above, also of F^∞. In the following sections this will let us turn any coalgebra $X \to TFX$ or $X \to TF_\varepsilon X$ into a system $X \to TF^\infty X$ and, by doing so, allow us to model their (in)finite behaviour.

Example 4.4. The terminal $\Sigma \times \mathcal{I}d$-coalgebra is

$$\zeta : \Sigma^\omega \to \Sigma \times \Sigma^\omega; a_1 a_2 \ldots \mapsto (a_1, a_2 a_3 \ldots).$$

The coproduct of $a : \Sigma \times A \to A$ and $b : \Sigma \times B \to B$ in $\mathsf{Alg}(F)$ is

$$a \oplus b : \Sigma \times (A + B) \to A + B; (\sigma, x) \mapsto \begin{cases} a(\sigma, x) & \text{if } x \in A, \\ b(\sigma, x) & \text{otherwise.} \end{cases}$$

Hence, the free Bloom algebra over X is a map $\Sigma \times (\Sigma^* \times X + \Sigma^\omega) \to \Sigma^* \times X + \Sigma^\omega$ explicitly given by:

$$(a, (\sigma, x)) \mapsto (a\sigma, x) \text{ and } (a, a_1 a_2 \ldots) \mapsto a a_1 a_2 \ldots.$$

Let $(a : FA \to A, (-)^\dagger)$ be a Bloom algebra, $b : FB \to B$ an F-algebra and $h : A \to B$ a homomorphism between F-algebras a and b. Then there is a unique assignment $(-)^\ddagger$ which turns $(b : FB \to B, (-)^\ddagger)$ into a Bloom algebra and h into a Bloom algebra homomorphism and it is defined as follows [1]: for $e : X \to FX$ the map $e^\ddagger : X \to B$ is $e^\ddagger \triangleq h \circ e^\dagger$.

$$\begin{tikzcd} X \arrow[r, "e^\dagger"] \arrow[rr, bend left, "e^\ddagger"] \arrow[d, "e"'] & A \arrow[r, "h"] & B \\ FX \arrow[r, "Fe^\dagger"] \arrow[rr, bend right, "Fe^\ddagger"'] & FA \arrow[u, "a"'] \arrow[r, "Fh"] & FB \arrow[u, "b"'] \end{tikzcd}$$

4.3. **Lifting monads to algebras.** Take an F-algebra $a : FA \to A$ and define $\bar{T}(a) \triangleq FTA \overset{\lambda_A}{\to} TFA \overset{Ta}{\to} TA$. If $h : A \to B$ is a homomorphism of algebras a and $b : FB \to B$ we put $\bar{T}(h) = T(h)$. In this case $\bar{T} : \mathsf{Alg}(F) \to \mathsf{Alg}(F)$ is a functor for which the morphism $\eta_A : A \to TA$ is an F-algebra homomorphism from $a : FA \to A$ to $\bar{T}(a) : FTA \to TA$. Moreover, $\mu_A : T^2 A \to TA$ is a homomorphism from $\bar{T}^2(a)$ to $\bar{T}(a)$ (see [7] for details). A direct consequence of this construction is the following.

Theorem 4.5. *[7] The triple $(\bar{T}, \bar{\mu}, \bar{\eta})$, where for $a : FA \to A$ we put*

$$\bar{\mu}_a : \bar{T}^2(a) \to \ \bar{T}(a); \bar{\mu}_a = \mu_A \text{ and } \bar{\eta}_a : a \to \bar{T}(a); \bar{\eta}_a = \eta_A$$

is a monad on $\mathsf{Alg}(F)$.

The above theorem together with the assumption of existence of an arbitrary free F-algebra in $\mathsf{Alg}(F)$ leads to a pair of adjoint situations captured by the diagram on the right. Since the composition of adjunctions is an adjunction this yields a monadic structure on the functor $TF^* : \mathsf{C} \to \mathsf{C}$.

$$\begin{tikzcd} \mathsf{C} \arrow[r, bend left] \arrow[r, phantom, "\perp"] & \mathsf{Alg}(F) \arrow[l, bend left] \arrow[r, bend left] \arrow[r, phantom, "\perp"] & \mathcal{K}l(\bar{T}) \arrow[l, bend left] \end{tikzcd}$$

Example 4.6. An example of this phenomenon is given by the monad $\mathcal{P}(\Sigma^* \times \mathcal{I}d)$ from Example 2.6 where in the above we set $T = \mathcal{P}$ and $F = \Sigma \times \mathcal{I}d$. This monad has already been described e.g. in [11], but it arose as a consequence of the composition of a different pair of adjunctions.

4.3.1. *Monads on Bloom algebras.* Above we gave a recipe for a general construction of a monadic structure on the functor TF^*. As witnessed in [9, 11], this monad is suitable to model coalgebras and their weak bisimulations and weak finite trace semantics (i.e. their finite behaviour). Our primary interest is in modelling infinite behaviour and this monad proves itself insufficient. The purpose of this subsection is to show how to tweak the middle category from the pair of adjunctions in pictured in the diagram above so that the monad obtained from the composition of two adjunctions is suitable for our needs.

Let $(a : FA \to A, (-)^\dagger)$ be a Bloom algebra and define

$$\bar{T}_B((a : FA \to A, (-)^\dagger)) \triangleq (\bar{T}(a) : FTA \to TA, (-)^\ddagger),$$

where for any $e : X \to FX$ the map $e^\ddagger$ is given by $\eta_A \circ e^\dagger$. Since $\eta_A : A \to TA$ is a homomorphism between $a : FA \to A$ and $\bar{T}(a) : FTA \to TA$ the pair $(\bar{T}(a), (-)^\ddagger)$ is a Bloom algebra. For a pair of Bloom algebras $(a : FA \to A, (-)^\dagger)$ and $(b : FB \to B, (-)^\ddagger)$ and a Bloom algebra homomorphism $h : A \to B$ between them put $\bar{T}_B(h) = T(h)$. This defines a functor $\bar{T}_B : \mathsf{Alg}_B(F) \to \mathsf{Alg}_B(F)$. Analogously to the previous subsection we have the following direct consequence of the construction.

Theorem 4.7. *The triple $(\bar{T}_B, \bar{\mu}^B, \bar{\eta}^B)$ is a monad on $\mathsf{Alg}_B(F)$, where for any Bloom algebra $(a : FA \to A, (-)^\dagger)$ the $(a, (-)^\dagger)$-components of the transformations $\bar{\mu}^B$ and $\bar{\eta}^B$ are*

$$\bar{\mu}^B_{(a,(-)^\dagger)} : \bar{T}^2_B(a, (-)^\dagger) \to \bar{T}_B(a, (-)^\dagger); \quad \bar{\mu}^B_{(a,(-)^\dagger)} = \mu_A \text{ and}$$
$$\bar{\eta}^B_{(a,(-)^\dagger)} : (a, (-)^\dagger) \to \bar{T}_B(a, (-)^\dagger) \text{ with } \bar{\eta}^B_{(a,(-)^\dagger)} = \eta_A.$$

Hence, we obtain two adjoint situations captured in the diagram below. These adjunctions impose a monadic structure on the functor $TF^\infty : \mathsf{C} \to \mathsf{C}$. The monad $\mathcal{P}(\Sigma^* \times \mathcal{I}d + \Sigma^\omega)$ from Example 2.7 arises from the composition of the above adjoint situations (see also Example 4.4).

$$\begin{tikzcd} \mathsf{C} \arrow[r, bend left] \arrow[r, phantom, "\perp"] & \mathsf{Alg}_B(F) \arrow[l, bend left] \arrow[r, bend left] \arrow[r, phantom, "\perp"] & \mathcal{K}l(\bar{T}_B) \arrow[l, bend left] \end{tikzcd}$$

Example 4.8. Let $F = \Sigma \times \mathcal{I}d^2$. This functor lifts to $\mathcal{K}l(\mathcal{P})$ [26] and, up to isomorphism, $F^\infty = T_\Sigma$ is a functor which assigns to any set X the set of complete binary trees (i.e. every node has either two children or no children) with inner nodes taking values in Σ and finitely many leaves, all taken from X [1] (see Subsection 2.1). This yields a monadic structure on $\mathcal{P}F^\infty = \mathcal{P}T_\Sigma$ defined in Example 2.8.

The above example can be easily generalized. Indeed, if T is a commutative Set-based monad then any polynomial functor[5] $F : \mathsf{Set} \to \mathsf{Set}$ lifts to $\mathcal{K}l(T)$ [26]. If it admits all free F-algebras and the final F-coalgebra then the assumptions from Subsection 4.1 hold for T and F yielding the monad TF^∞.

5. Abstract automata and their behaviour

The purpose of this section is to generalize the concepts from Section 3 to an arbitrary Kleisli category with a suitable ordering. In other words, given a Set-monad T, we define a T-automaton, its finite behaviour, its infinite behaviour with BAC and provide generic Kleene theorems for T.

Let (T, μ, η) be a Set-based monad. Since we will often consider the Lawvere theory $\mathbb{T}$ associated with it, recall that its objects are sets given by $n = \{1, \ldots, n\}$ for $n = 0, 1, \ldots$. We start with the definition of a T-automaton.

Definition 5.1. A *T-automaton* or simply *automaton* is a pair $(\alpha, \mathfrak{F})$, where $\alpha : X \multimap X = X \to TX$ is a T-coalgebra called *transition morphism* and $\mathfrak{F} \subseteq X$.

[5] A *polynomial functor* is a functor defined by the grammar $F \triangleq \Sigma \in \mathsf{Set} \mid \mathcal{I}d \mid F \times F \mid \sum F$ [26].

5.1. **Assumptions.** In order to define finite and infinite behaviour of $(\alpha, \mathfrak{F})$ and reason about it we require the Kleisli category for T to satisfy more assumptions. In this section we assume that:

(A) $\mathcal{K}l(T)$ is order enriched with a pointwise induced order,
(B) it is $\omega\mathsf{Cpo}$-enriched,
(C) it is left distributive,
(D) its hom-sets are complete lattices.

At first let us note that, in the light of Remark 2.9, by (A) we get:

- right distributivity w.r.t. the base morphisms and
- cotupling order preservation.

Additionally, since the Kleisli category for T is left distributive we have:

- the bottoms $\bot_{X,Y} \in \mathcal{K}l(T)(X,Y)$ satisfy $g \cdot \bot_{X,Y} = \bot_{X,Z}$ for any $g : Y \multimap Z$.

The axioms (A)-(C) guarantee that the saturation $\alpha \mapsto \alpha^*$ is computable in ω-steps and is expressive enough. For a given $\alpha : X \multimap X = X \to TX$ we can define maps $\alpha^*, \alpha^+ : X \multimap X = X \to TX$:

$$\alpha^* \triangleq \mu x.(\mathsf{id} \vee x \cdot \alpha) = \bigvee_{n<\omega} (\mathsf{id} \vee \alpha)^n, \text{ and } \alpha^+ \triangleq \alpha^* \cdot \alpha.$$

The operator $\alpha \mapsto \alpha^*$ was thoroughly studied in [10, 11, 13, 14] in the context of coalgebraic weak bisimulation. Its definition does *not* require a complete lattice order. See *loc. cit.* for a discussion.

Assumption (D) allows us to define the greatest fixpoint of the map $x \mapsto x \cdot \alpha$. Indeed for any $\alpha : X \multimap X = X \to TX$ put $\alpha^\omega : X \multimap 0 = X \to T0$ to be:

$$\alpha^\omega \triangleq \bigwedge_{\kappa \in \mathsf{Ord}} (x \mapsto x \cdot \alpha)^\kappa(\top),$$

where $\top : X \multimap 0 = X \to T0$ is the greatest element of $\mathcal{K}l(T)(X, 0)$ and

$$(x \mapsto x \cdot \alpha)^\kappa : (X \multimap 0) \to (X \multimap 0)$$

is defined in terms of the transfinite induction by

$$(x \mapsto x \cdot \alpha)^{\kappa+1} = (x \mapsto x \cdot \alpha) \circ (x \mapsto x \cdot \alpha)^\kappa$$

for a successor ordinal $\kappa + 1$ and $(x \mapsto x \cdot \alpha)^\kappa = \bigwedge_{\lambda<\kappa} (x \mapsto x \cdot \alpha)^\lambda$ for a limit ordinal κ. By the Tarski-Knaster theorem, the map $\alpha^\omega : X \multimap 0 = X \to T0$ is the greatest fixpoint of the assignment $x \mapsto x \cdot \alpha$.

Example 5.2. The Kleisli categories for the monads from Example 2.10 satisfy (A)-(D). Section 6 presents one more example of a Kleisli category that fits the setting in the context of probabilistic automata.

Before we present the definition of finite and infinite behaviour of automata we need one more technical result.

Lemma 5.3. *For any $\alpha, \beta : X \multimap X = X \to TX$ we have:*

(1) $\mathsf{id}^* = \mathsf{id}$, $\mathsf{id} \leq \alpha^*$ *&* $\alpha^* \cdot \alpha^* = \alpha^*$,
(2) $(\alpha \cdot \beta)^\omega = (\beta \cdot \alpha)^\omega \cdot \beta$,
(3) $(\alpha^n)^\omega = \alpha^\omega$ *for any* $n > 0$,
(4) $\alpha^\omega = (\alpha^+)^\omega$.

Proof. The proof of (1) can be found in [11, 13]. To see (2) holds, i.e. $(\alpha\cdot\beta)^\omega = (\beta\cdot\alpha)^\omega\cdot\beta$ note that $(\beta\cdot\alpha)^\omega\cdot\beta$ is a fixpoint of $\lambda x.x\cdot\alpha\cdot\beta$ and hence $(\beta\cdot\alpha)^\omega\cdot\beta \leq (\alpha\cdot\beta)^\omega$. By a similar argument we show $(\alpha\cdot\beta)^\omega\cdot\alpha \leq (\beta\cdot\alpha)^\omega$. Thus,

$$(\beta\cdot\alpha)^\omega = (\beta\cdot\alpha)^\omega\cdot\beta\cdot\alpha \leq (\alpha\cdot\beta)^\omega\cdot\alpha \leq (\beta\cdot\alpha)^\omega.$$

To prove $(\alpha^n)^\omega = \alpha^\omega$ note that by (2) we have $(\alpha^n)^\omega = (\alpha^{n-1}\cdot\alpha)^\omega = (\alpha\cdot\alpha^{n-1})^\omega\cdot\alpha$. Hence, $(\alpha^n)^\omega \leq \alpha^\omega$. Moreover, since $\alpha^\omega\cdot\alpha^n = \alpha^\omega\cdot\alpha^{n-1} = \ldots = \alpha^\omega$ we get the converse inequality, i.e. $\alpha^\omega \leq (\alpha^n)^\omega$. This proves the assertion.

Finally, note that by monotonicity of $(-)^\omega$ since $\alpha \leq \alpha^*\cdot\alpha = \alpha^+$ we have $\alpha^\omega \leq (\alpha^+)^\omega$. Moreover,

$$\begin{aligned}&(\alpha^+)^\omega = (\alpha^*\cdot\alpha)^\omega \leq (\alpha^*\cdot\alpha)^\omega\cdot(\mathsf{id}\vee\alpha) = (\alpha^*\cdot\alpha)^\omega \vee (\alpha^*\cdot\alpha)^\omega\cdot\alpha \leq \\ &(\alpha^*\cdot\alpha)^\omega \vee (\alpha^*\cdot\alpha)^\omega\cdot\alpha^*\cdot\alpha = (\alpha^*\cdot\alpha)^\omega \vee (\alpha^*\cdot\alpha)^\omega = (\alpha^+)^\omega\end{aligned}$$

Hence, by induction we prove that $(\alpha^+)^\omega\cdot(\mathsf{id}\vee\alpha)^i = (\alpha^+)^\omega$. By the fact that our theory is $\omega\mathsf{Cpo}$-enriched we get:

$$(\alpha^+)^\omega\cdot\alpha^* = (\alpha^+)^\omega\cdot\bigvee_i(\mathsf{id}\vee\alpha)^i = \bigvee_i(\alpha^+)^\omega\cdot(\mathsf{id}\vee\alpha)^i = (\alpha^+)^\omega.$$

This proves that $(\alpha^+)^\omega$ satisfies $(\alpha^+)^\omega\cdot\alpha^*\cdot\alpha = (\alpha^+)^\omega\cdot\alpha$. Thus, $(\alpha^+)^\omega \leq \alpha^\omega$ which completes the proof. □

5.2. **Finite and infinite behaviour.** The purpose of this subsection is to present the definitions of the finite and infinite behaviour with BAC for T-automata. Let $(\alpha : X \to TX, \mathfrak{F} \subseteq X)$ be a T-automaton. Before we start, let us first encode the set $\mathfrak{F}$ of accepting states in terms of an endomorphism $\mathfrak{f}_\mathfrak{F} : X \multimap X = X \to TX$ by:

$$\mathfrak{f}_\mathfrak{F}(x) = \begin{cases} \eta_X(x) & \text{if } x\in\mathfrak{F}, \\ \perp & \text{otherwise}\end{cases} \quad \text{for any } x \in X,$$

where $\perp$ denotes the bottom element of the poset TX.

Definition 5.4. *Finite* and *ω-behaviour* of the automaton $(\alpha,\mathfrak{F})$ are given respectively in terms of morphisms in $\mathcal{K}l(T)$ by: $||(\alpha,\mathfrak{F})|| : X \multimap 1$ and $||(\alpha,\mathfrak{F})||_\omega : X \multimap 0$, where

$$||(\alpha,\mathfrak{F})|| \triangleq ! \cdot \mathfrak{f}_\mathfrak{F}\cdot\alpha^* \text{ and } ||(\alpha,\mathfrak{F})||_\omega \triangleq (\mathfrak{f}_\mathfrak{F}\cdot\alpha^+)^\omega.$$

Finite behaviour of a state $x \in X$ of $(\alpha,\mathfrak{F})$ is the map $||(\alpha,\mathfrak{F})||\cdot x_X : 1 \multimap 1$, and its *$\omega$-behaviour* is given by $||(\alpha,\mathfrak{F})||_\omega\cdot x_X : 1 \multimap 0$. Here,

$$x_X : 1 \multimap X = 1 \to TX; 1 \mapsto \eta_X(x).$$

Example 5.5. As we have already seen in Section 3, the finite and ω-behaviour of $\mathcal{P}(\Sigma^*\times\mathcal{I}d + \Sigma^\omega)$-automata coincides with the classical notions whenever the tuple is given by $(\alpha : n \to \mathcal{P}(\Sigma\times n), \mathfrak{F} \subseteq n)$. The same applies to tree automata (see Proposition 3.2).

5.3. **Additional remarks.** Our approach to defining semantics for coalgebras seems to diverge slightly from the established coalgebraic takes known from e.g. [9, 26, 45, 50]. The purpose of this subsection is to compare our setting with the frameworks presented in the literature and try to justify the (slight) differences.

Our approach builds on top of two fixpoint operators, namely $(-)^*$ and $(-)^\omega$. The choice of these two operators, and not other (e.g. $(-)^\dagger$ from [9, 19]) follows from the premise that we wanted to make the connection with the classical results in regular and ω-regular languages as clear and as direct as possible. As we witness here, the classical Kleene star operation and $(-)^\omega$ [41] prove to have their general categorical counterparts.

It may not be clear to the reader why the finite and ω-behaviour maps have different codomains, i.e. the former is a map $X \multimap\!\!\to 1$ and the latter $X \multimap\!\!\to 0$. Let us focus on the finite behaviour first. So far in the coalgebraic literature, finite behaviour of systems was introduced in terms of finite trace [9, 30, 45]. In the setting of systems $X \to TFX$ it is obtained in terms of the initial algebra-final coalgebra coincidence [9, 26]. When translated to the setting of systems with internal moves, the finite trace is given by $\alpha^\dagger = \mu x.x \cdot \alpha : X \multimap\!\!\to 0$ and is calculated in the Kleisli category for the monad TF^* [10, 11]. However, this holds for coalgebras whose type monad encodes accepting states. From the point of our setting, the accepting states are not part of the transition and are encoded in terms of $\mathfrak{F} \subseteq X$ instead. The direct use of initial algebra-final coalgebra coincidence makes no sense here, as the initial algebra would simply be degenerate. Luckily, there is a simple formal argument showing that our approach from this paper and the aforementioned approach established in the coalgebraic literature coincide. For the monad T and a T-automaton $(\alpha, \mathfrak{F})$ consider the monad $T(\mathcal{I}d+1)$[6] and the map $\underline{\alpha} : X \to T(X+1)$ defined for any $x \in X$ by:

$$\underline{\alpha}(x) = \alpha(x) \vee \chi_{\mathfrak{F}}(x),$$

where $\chi_{\mathfrak{F}} : X \to T(X+1); x \mapsto \begin{cases} \eta_{X+1}(1) & \text{if } x \in \mathfrak{F}, \\ \bot & \text{otherwise.} \end{cases}$ It is a simple exercise to prove that the least fixpoint $\mu x.x \cdot \underline{\alpha} : X \multimap\!\!\to 0 = X \to T(0+1) = X \to T1$ calculated in $\mathcal{K}l(T(\mathcal{I}d+1))$ is the same as the finite behaviour map $||(\alpha, \mathfrak{F})|| : X \multimap\!\!\to 1 = X \to T1$ calculated in $\mathcal{K}l(T)$. Therefore, our definition of finite behaviour via $(-)^*$ coincides with the coalgebraic finite trace semantics via $\alpha^\dagger = \mu x.x \cdot \alpha$.

The finite behaviour of a state of an automaton from Definition 5.4 is of type $1 \multimap\!\!\to 1$. We argue that this map should be viewed as a generalization of a finitary language. Classically, these languages have been considered in some algebraic context, e.g. with the familiar algebraic operations of concatenation, Kleene start closure and finite union. These operations considered on our abstract categorical level directly translate into morphism composition, saturation and finite joins of endomorphisms $1 \multimap\!\!\to 1$ respectively.

As far as the infinite behaviour is concerned, it should be noted here that our prototypical example of a monad is $TF^\infty = T(F^* \oplus F^\omega)$ from Section 4. By Theorem 4.2 the object $(F^* \oplus F^\omega)(0) = F^\omega$ is the carrier of the terminal F-coalgebra $\zeta : F^\omega \to FF^\omega$ making $TF^\infty 0 = TF^\omega$. This is exactly what we expect to have as a codomain of an infinite trace map (see also for comparison [17, 50]).

Additionally, the type of the infinite behaviour of a state of an automaton is $1 \multimap\!\!\to 0$ and it reflects the *partial* algebraic nature of (in)finitary languages. In particular, it makes sense to compose (concatenate) a finitary langugage $1 \multimap\!\!\to 1$ with an infinitary language $1 \multimap\!\!\to 0$ and get an infinitary one ($1 \multimap\!\!\to 1 \multimap\!\!\to 0 = 1 \multimap\!\!\to 0$), but *not* vice versa. Moreover, it does *not* necessarily make sense to compose two infinitary languages ($1 \multimap\!\!\to 0$ and $1 \multimap\!\!\to 0$) with each other.

[6] It can be easily verified that for any monad T the functor $T(\mathcal{I}d+1)$ carries a monadic structure. It follows from the fact that the exception monad $\mathcal{I}d + E$ induces an exception monad transfomer $T \mapsto T(\mathcal{I}d + E)$.

5.4. **Kleene theorems.** The purpose of this part of the paper is to state and prove Kleene theorems akin to Proposition 3.4 and 3.5. These theorems require us to work with *finite* automata and their behaviour, so we will restrict the setting of this subsection to the Lawvere theory $\mathbb{T}$ associated with the monad T.

In this subsection we consider a set $\mathcal{A}$ of endomorphisms from $\mathbb{T}$ such that:

- $\mathcal{A}$ contains all base map endomorphisms,
- $\perp_n^n \colon n \multimap n \in \mathcal{A}$ for any $n < \omega$,
- for any $\alpha : n \multimap n \in \mathcal{A}$ we have $\perp \cdot \alpha = \perp$,
- if $\{\alpha_k : n_i \multimap n_i\}_{k=1,\ldots,k} \subseteq \mathcal{A}$ then $\alpha_1 + \ldots + \alpha_k \in \mathcal{A}$,
- $\mathcal{A}$ is closed under taking finite suprema.

The set $\mathcal{A}$ plays a role of a set of admissible transition functions for automata taken into consideration. A T-automaton $(\alpha, \mathfrak{F})$ whose transition $\alpha : n \multimap n$ is an arrow in $\mathbb{T}$ is called $\mathcal{A}$-*automaton* if $\alpha \in \mathcal{A}$.

Example 5.6. In the case of our leading examples of theories, namely LTS^ω and TTS^ω, the prototypical choice for $\mathcal{A}$ was given in condition (a) in Proposition 3.5 and 3.4 respectively.

Because the order enrichment of $\mathcal{K}l(T)$ is pointwise induced, any base morphism f satisfies $\perp \cdot f = \perp$. As a simple consequence of this observation, we instantly get that $\perp \cdot \alpha^* = \alpha^* \cdot \perp = \perp$ for any $\alpha \in \mathcal{A}$.

Definition 5.7. The set of *regular morphisms* $m \multimap p \in \mathbb{T}$ is defined by:

$$\begin{aligned}\mathfrak{Reg}(m,p) \triangleq \{ & j' \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \alpha^* \cdot j \mid (\alpha : n \multimap n, \mathfrak{F}) \text{ is a } \mathcal{A}\text{-automaton and} \\ & j : m \multimap n, j' : n \multimap p \text{ are base maps}\}.\end{aligned}$$

The set of regular morphisms $\mathfrak{Reg}(1,p)$ will be often referred to as the set of *regular trees with variables in p*. Note that $\mathfrak{Reg}(1,1)$ is exactly the set of finite behaviours of states in $\mathcal{A}$-automata.

We list the statements without proofs which we later provide in Subsection 5.4.1.

Lemma 5.8. *The identity maps in $\mathbb{T}$ are regular morphisms. Moreover, regular morphisms are closed under composition from $\mathbb{T}$.*

The above lemma precisely says that the collection of objects $n = 0, 1, \ldots$ with morphisms $\mathfrak{Reg}(m,n)$ forms a category with the composition taken from $\mathbb{T}$. We denote this category by $\mathfrak{Reg}(\mathcal{A})$.

Theorem 5.9 (Kleene theorem for regular behaviour)**.** *The category $\mathfrak{Reg}(\mathcal{A})$ is a subtheory of $\mathbb{T}$ such that:*

(a) it contains all maps from $\mathcal{A}$,
(b) it admits finite suprema,
(c) its endomorphisms are closed under $(-)^*$.

Moreover, if $\mathfrak{Rat}(\mathcal{A})$ denotes the smallest subtheory of $\mathbb{T}$ which satsfies (a)-(c) then

$$\mathfrak{Rat}(\mathcal{A}) = \mathfrak{Reg}(\mathcal{A}).$$

Finally, we define

$$\omega\mathfrak{Rat}(\mathcal{A}) \triangleq \{[r_1, \ldots, r_m]^\omega \cdot r \mid r, r_i \in \mathfrak{Rat}(1,m) \text{ for } m < \omega\}$$

$$\omega\mathfrak{Reg}(\mathcal{A}) \triangleq \{||(\alpha, \mathfrak{F})||_\omega \cdot i_n : 1 \multimap 0 \mid (\alpha, \mathfrak{F}) \text{ is an } \mathcal{A}\text{-automaton}\}.$$

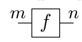
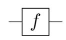
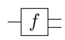
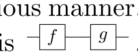
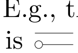
Theorem 5.10 (Kleene theorem for ω-regular behaviour)**.** *We have*

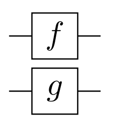
$$\omega\mathfrak{Rat}(\mathcal{A}) = \omega\mathfrak{Reg}(\mathcal{A}).$$

5.4.1. *Omitted proofs.* The purpose of this subsection is to present the proofs of statements above. In order to do this, we need to introduce some new notions and define a notation used below. We start off by defining $[\mathcal{A}]$ to be the set of morphisms from $\mathbb{T}$ obtained by (pre- and post-)composing maps from $\mathcal{A}$ with base morphisms with suitable domains and codomains:

$$[\mathcal{A}] \triangleq \{i \cdot \alpha \cdot j \mid \alpha : n \mathrel{-\!\circ\!\!\to} n \in \mathcal{A} \text{ and } i : n \mathrel{-\!\circ\!\!\to} p, j : m \mathrel{-\!\circ\!\!\to} n \text{ are base maps}\}.$$

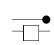
Let us now develop a string diagram notation which will simplify our proofs considerably. We adopt the standard string diagram calculus for monoidal categories [21, 44] which will be tailored for our purposes.

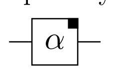
A morphism $f : m \mathrel{-\!\circ\!\!\to} n$ is depicted by  . We will often drop the (co)domain types from the notation and depict f simply by  . If $f : m \mathrel{-\!\circ\!\!\to} n+p$ and the coproduct codomain needs to be emphasized by the diagram notation then we depict f by  . This generalizes to $m_1 + \ldots + m_k \mathrel{-\!\circ\!\!\to} n_1 + \ldots + n_l$ in an obvious manner. Whenever $f : m \mathrel{-\!\circ\!\!\to} n$ and $g : n \mathrel{-\!\circ\!\!\to} p$ then the composition $g \cdot f : m \mathrel{-\!\circ\!\!\to} p$ is  . We will often use a separate notation to denote special morphisms. E.g., the identity map $\mathsf{id} : n \mathrel{-\!\circ\!\!\to} n$ is depicted by , the map $\mathsf{in}^m_{m+n} : m \mathrel{-\!\circ\!\!\to} m+n$ is . Given two maps $f : m_1 \mathrel{-\!\circ\!\!\to} n_1$ and $g : m_2 \mathrel{-\!\circ\!\!\to} n_2$ the coproduct $f + g : m_1 + m_2 \mathrel{-\!\circ\!\!\to} n_1 + n_2$ is depicted by



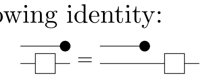
The cotuple string diagram notation has already been presented in Subsection 2.7. However, in the case of the cotuple

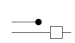
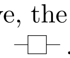
$$[\bot^m_n : m \mathrel{-\!\circ\!\!\to} n, f : n \mathrel{-\!\circ\!\!\to} n] : m + n \mathrel{-\!\circ\!\!\to} n$$

the following diagram is used to depict it: . Finally, given any endomorphism $\alpha : n \mathrel{-\!\circ\!\!\to} n$ we depict the saturated map α^* by



Now we are ready to list some basic observations and remarks about the diagram calculus introduced above. First of all since $\bot$ satisfies $f \cdot \bot = \bot$ (Assumption C) for any f, we have: $f \cdot [\bot, \mathsf{id}] = [f \cdot \bot, f \cdot \mathsf{id}] = [\bot, f]$, which diagrammatically is represented in terms of the following identity:



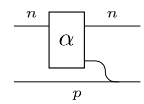
In the above, the right hand side of the equality, namely , is the composition of  and . Secondly, the map $\bot^m_n + \mathsf{id}_p : m + p \mathrel{-\!\circ\!\!\to} n + p$ satisfies:

$$\bot^m_n + \mathsf{id}_p = \mathsf{in}^p_{m+p} \cdot [\bot^m_p, \mathsf{id}_p].$$

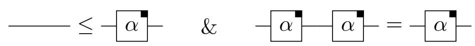
Hence, we can capture $\bot^m_n + \mathsf{id}_p$ by . Moreover, $[\alpha, \mathsf{in}^p_{n+p}] : n + p \mathrel{-\!\circ\!\!\to} n + p$ satisfies:

$$[\alpha, \mathsf{in}^p_{n+p}] = (\mathsf{id}_n + [\mathsf{id}_p, \mathsf{id}_p]) \cdot (\alpha + \mathsf{id}_p)$$

Hence, we can depict it in the diagram:

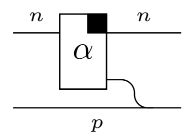


Additionally, by the properties of saturation the following diagram (in)equalities hold:

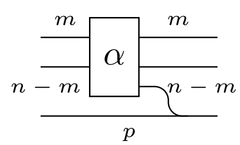


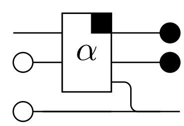
Definition 5.11. A regular morphism $r \in \mathfrak{Reg}(m,p)$ is said to be in *normal form* (NF) if $r = [\perp_p^n, \mathsf{id}_p] \cdot [\alpha, \mathsf{in}_{n+p}^p]^* \cdot \mathsf{in}_{n+p}^m$ for some $\alpha : n \multimap n+p \in [\mathcal{A}]$ and $m \leq n$.

Given $\alpha : n \multimap n+p$ the saturated map $[\alpha, \mathsf{in}_{n+p}^p]^* : n+p \multimap n+p$ is of the form $[\gamma, \mathsf{in}_{n+p}^p]$ for a morphism $\gamma : n \multimap n+p$, which follows by the fact that cotupling preserves the order. Hence, in order to simplify the notation we will extend it and depict $[\alpha, \mathsf{in}_{n+p}^p]^*$ by the following picture:

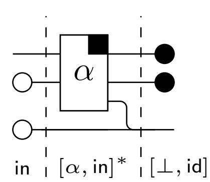


Hence, the map $[\alpha, \mathsf{in}^p]$ and r in NF from Definition 5.11 are respectively depicted by the string diagrams below:





The right-hand-side diagram is a correct representation of r as r is the result of the composition of three maps:



The following lemma states that all regular morphisms can be given in their normal forms and that they can be obtained from regular trees via cotupling.

Lemma 5.12. *The following equality is true:*

$$\mathfrak{Reg}(m,p) = \{[r_1, \ldots, r_m] \mid r_i \in \mathfrak{Reg}(1,p)\} =$$
$$\{[\perp_p^n, \mathsf{id}_p] \cdot [\alpha, \mathsf{in}_{n+p}^p]^* \cdot \mathsf{in}_{n+p}^m \mid \alpha : n \multimap n+p \in [\mathcal{A}] \text{ and } m \leq n\}.$$

Proof. The proof is technical and is divided into two parts.
Part 1. In the first step of the proof we show that

$$\mathfrak{Reg}(m,p) = \{[r_1, \ldots, r_m] \mid r_i \in \mathfrak{Reg}(1,p)\}.$$

. It is clear that $\mathfrak{Reg}(m,p) \subseteq \{[r_1, \ldots, r_m] \mid r_i \in \mathfrak{Reg}(1,p)\}$. Indeed, take

$$j' \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \alpha^* \cdot j \in \mathfrak{Reg}(m,p)$$

for an $\mathcal{A}$-automaton $(\alpha : n \rightarrow\!\!\circ\!\!\rightarrow n, \mathfrak{F})$ and base maps $j : m \rightarrow\!\!\circ\!\!\rightarrow n, j' : n \rightarrow\!\!\circ\!\!\rightarrow p$, and note that

$$j' \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \alpha^* \cdot j = [j' \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \alpha^* \cdot (j^1)_n, \ldots, j' \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \alpha^* \cdot (j^m)_n],$$

where $j = [(j^1)_n, \ldots, (j^m)_n]$ (as it is a base morphism). To prove the converse take $[r_1, \ldots, r_m]$ for $r_k = j'_k \cdot \mathfrak{f}_{\mathfrak{F}_k} \cdot \alpha_k^* \cdot j_k$, where $\alpha_k : n_k \rightarrow\!\!\circ\!\!\rightarrow n_k \in \mathcal{A}$, $\mathfrak{F}_k \subseteq n_k$ and $j_k : 1 \rightarrow\!\!\circ\!\!\rightarrow n_k$, $j'_k : n_k \rightarrow\!\!\circ\!\!\rightarrow p$ are base maps. Put $n = n_1 + \ldots + n_m$ and define $\alpha : n \rightarrow\!\!\circ\!\!\rightarrow n$ by $\alpha \triangleq \alpha_1 + \ldots + \alpha_m$ and $\mathfrak{F} = \overline{\mathfrak{F}_1} \cup \ldots \cup \overline{\mathfrak{F}_m}$ is a subset of n, where $\overline{\mathfrak{F}_k} \subseteq n$ is given by $\overline{\mathfrak{F}_1} = \mathfrak{F}_1$ and

$$\overline{\mathfrak{F}_k} \triangleq \{i + n_1 + \ldots n_{k-1} \mid i \in \mathfrak{F}_k\} \text{ for } k > 1.$$

Note that $\mathfrak{F}$ satisfies

$$\mathfrak{f}_{\mathfrak{F}} = \mathfrak{f}_{\mathfrak{F}_1} + \ldots + \mathfrak{f}_{\mathfrak{F}_m} : n_1 + \ldots + n_m \rightarrow\!\!\circ\!\!\rightarrow n_1 + \ldots + n_m.$$

Finally, let us define $j : 1 + \ldots + 1 = m \rightarrow\!\!\circ\!\!\rightarrow n_1 + \ldots + n_m = n$ and $j' : n_1 + \ldots + n_m = m \rightarrow\!\!\circ\!\!\rightarrow p$ by:

$$j' = j'_{n_1} + \ldots + j'_{n_m} \text{ and } j = [j_1, \ldots, j_m].$$

Note that j, j' are base maps and by the properties of $\mathcal{A}$ it follows that $(\alpha, \mathfrak{F})$ is an $\mathcal{A}$-automaton. Moreover,

$$j' \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \alpha^* \cdot j = j' \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \left(\bigvee_{l<\omega} (\alpha \vee \mathsf{id})^l \right) \cdot j \overset{\diamond}{=} \bigvee_l j' \cdot \mathfrak{f}_{\mathfrak{F}} \cdot (\alpha \vee \mathsf{id})^l \cdot j =$$

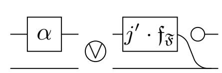
$$\bigvee_l [j'_1 \cdot \mathfrak{f}_{\mathfrak{F}_1} \cdot (\alpha_1 \vee \mathsf{id})^l \cdot j_1, \ldots, j'_m \cdot \mathfrak{f}_{\mathfrak{F}_m} \cdot (\alpha_m \vee \mathsf{id})^l \cdot j_m] \overset{\square}{=} [r_1, \ldots, r_m].$$

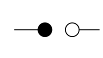
The equation marked with $(\diamond)$ follows by left distributivity, right distributivity w.r.t. the base morphisms and $\omega\mathsf{Cpo}$-enrichment. The equation $(\square)$ is a consequence of the fact that cotupling preserves the order.

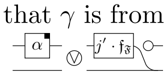
Part 2. We will now show that any $r \in \mathfrak{Reg}(m, p)$ can be given in its normal form. By the definition of $\mathfrak{Reg}$ we know that there is a suitable $\mathcal{A}$-automaton $(\alpha : n \rightarrow\!\!\circ\!\!\rightarrow n, \mathfrak{F})$ and base maps j', j such that $r = j' \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \alpha^* \cdot j$. Put $\gamma : n + p \rightarrow\!\!\circ\!\!\rightarrow n + p$ to be

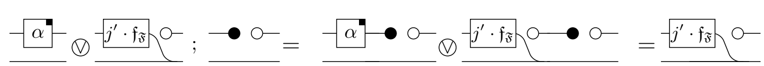
$$\gamma \triangleq [\mathsf{in}^n_{n+p} \cdot \alpha \vee \mathsf{in}^p_{n+p} \cdot j' \cdot \mathfrak{f}_{\mathfrak{F}}, \mathsf{in}^p_{n+p}]$$

and define $\mathfrak{F}' \subseteq n + p$ by $\mathfrak{F}' \triangleq \{n + 1, \ldots n + p\}$. Then $\mathfrak{f}_{\mathfrak{F}'} = (\perp^n_n + \mathsf{id}_p)$ and the transition map γ together with $\mathfrak{f}_{\mathfrak{F}'}$ are given in terms of the following string diagrams respectively:

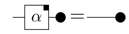


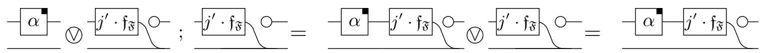


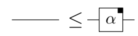
Note that γ is from $[\mathcal{A}]$ due to the fact that $j' \cdot \mathfrak{f}_{\mathfrak{F}'}$ is a member of $[\mathcal{A}]$. If we define γ' by  then $\gamma^* \geq \gamma' \geq \gamma \vee \mathsf{id}$ which proves $\gamma'^* = \gamma^*$. Moreover, $\mathfrak{f}_{\mathfrak{F}'} \cdot \gamma'$ equals[7]:

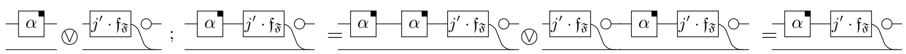


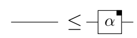
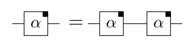
[7]We denote the composition *of* string diagrams by ; to emphasize the fact that in this case we use the inverse notation. Moreover, in order to avoid writing brackets, we assume that in the string diagram notation $\vee$ binds stronger than ;.

The last identity in the above follows from left distributivity and  (this identity is a consequence of the fact that for any $\alpha \in \mathcal{A}$ we have $\perp \cdot \alpha^* = \perp$). Now, $\mathfrak{f}_{\mathfrak{F}'} \cdot \gamma'^2$ is given by:



The last identity follows from the inequality . Finally, $\mathfrak{f}_{\mathfrak{F}'} \cdot \gamma'^3$ is:



where the last identity is a consequence of  and . Hence, $\mathfrak{f}_{\mathfrak{F}'} \cdot \gamma'^3 = \mathfrak{f}_{\mathfrak{F}'} \cdot \gamma'^2$ and, therefore, by left distributivity we get: $\mathfrak{f}_{\mathfrak{F}'} \cdot \gamma'^2 = \mathfrak{f}_{\mathfrak{F}'} \cdot \gamma'^* = \mathfrak{f}_{\mathfrak{F}'} \cdot \gamma^*$. Moroever,

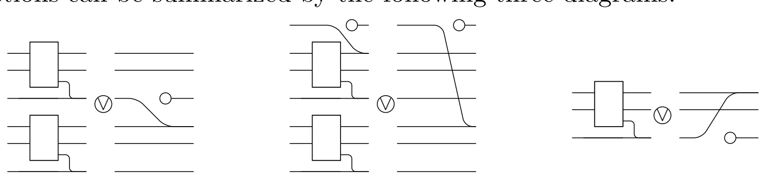
$$\begin{aligned}&[\perp_p^n, \mathsf{id}_p] \cdot \gamma^* \cdot \mathsf{in}_{n+p}^n \cdot j = [\perp_p^n, \mathsf{id}_p] \cdot \mathfrak{f}_{\mathfrak{F}'} \cdot \gamma^* \cdot \mathsf{in}_{n+p}^n \cdot j = \\ &[\perp_p^n, \mathsf{id}_p] \cdot \mathfrak{f}_{\mathfrak{F}'} \cdot \gamma'^2 \cdot \mathsf{in}_{n+p}^n \cdot j = j' \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \alpha^* \cdot j.\end{aligned}$$

Now, since the terminal states in the automaton $(\gamma, \{n+1, \ldots, n+p\})$ are disjoint from the image of the map $\mathsf{in}_{n+p}^n \cdot j$ we may rename the states and obtain an automaton $(\gamma'', \{n+1, \ldots, n+p\})$ such that

$$[\perp_p^n, \mathsf{id}_p] \cdot \gamma^* \cdot \mathsf{in}_{n+p}^n \cdot j = [\perp_p^n, \mathsf{id}_p] \cdot \gamma''^* \cdot \mathsf{in}_{n+p}^m.$$

This proves the assertion. □

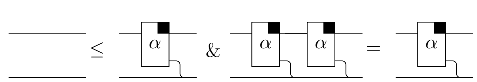
We will now proceed with proofs of Lemma 5.8 and Theorem 5.9. In particular, they show that regular morphisms with suitable domains and codomains are closed under composition, finite joins and the saturation operation. The constructions used in the proofs of the results below are simple generalization of classical constructions of non-deterministic automata with ε-moves used in proving that concatenation/finite union/Kleene star of regular languages is regular (see e.g. [27]). Hence, in our opinion, it can be considered a folklore in computer science which presents itself very aesthetically in terms of the string diagram calculus. These constructions can be summarized by the following three diagrams.



$r_1 \cdot r_2$ $\qquad$ $r_1 \vee r_2$ $\qquad$ r^*

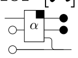
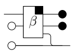
Note that for classical regular languages it was enough to consider the case where the normal form $[\perp, \mathsf{id}_p] \cdot [\alpha, \mathsf{in}^p]^* \cdot \mathsf{in}^m$ of the expressions satisfied $m = p = 1$ (i.e. one initial and one final state).

Proof. (Lemma 5.8) For a morphism $\alpha : n \multimap n + p$ the saturated map $[\alpha, \mathsf{in}_{n+p}^p]^*$ satisfies:



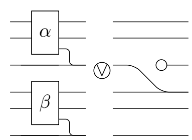
It is easy to see that $\mathsf{id}_m \in \mathfrak{Reg}(m,m)$. Let $r_1 \in \mathfrak{Reg}(m_1,m_2)$ and $r_2 \in \mathfrak{Reg}(m_2,m_3)$. By Lemma 5.12 both r_1 and r_2 are given in their normal forms:

$$r_1 = [\perp^{n_1}_{m_2}, \mathsf{id}_{m_2}] \cdot [\alpha, \mathsf{in}^{m_2}_{n_1+m_2}]^* \cdot \mathsf{in}^{m_1}_{n_1+m_2} \text{ and}$$
$$r_2 = [\perp^{n_2}_{m_3}, \mathsf{id}_{m_3}] \cdot [\beta, \mathsf{in}^{m_3}_{n_2+m_3}]^* \cdot \mathsf{in}^{m_2}_{n_2+m_3}.$$

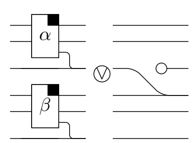
for $[\mathcal{A}]$-maps $\alpha : n_1 \to n_1 + m_2$, $\beta : n_2 \to n_2 + m_3$. We can depict r_1 and r_2 by  and  respectively. Define $\gamma : n_1+m_2+n_2+m_3 \multimap n_1+m_2+n_2+m_3$ by:

$$\gamma \triangleq ([\alpha, \mathsf{in}^{m_2}_{n_1+m_2}] + [\beta, \mathsf{in}^{m_3}_{n_2+m_3}]) \vee \mathsf{t}$$

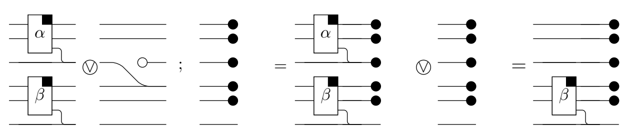
where t is given by $\mathsf{t} \triangleq [\mathsf{in}^{n_1}, \mathsf{in}^{n_2+m_3}_{n_1+m_2+n_2+m_3} \cdot \mathsf{in}^{m_2}_{n_2+m_3}, \mathsf{in}^{n_2+m_3}]$. Note that γ is a map from $[\mathcal{A}]$ and it is given by the following diagram.

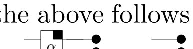


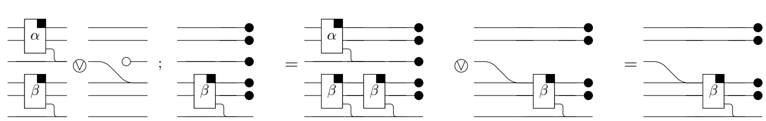
The map γ satisfies $\gamma^* \geq \gamma' \geq \gamma \vee \mathsf{id}$ (which means that $\gamma'^* = \gamma^*$) for γ' defined in terms of the diagram below:

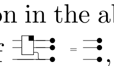
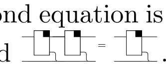


We have $[\perp^{n_1+m_2+n_2}_{m_3}, \mathsf{id}_{m_3}] \cdot \gamma' = [\perp^{n_1+m_2}_{m_3}, [\perp^{n_2}_{m_3}, \mathsf{id}_{m_3}] \cdot [\beta, \mathsf{in}^{m_3}]^*]$ since:

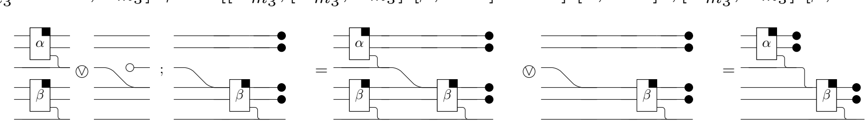


The first equation in the above follows from left distributivity and the second equation is a consequence of . Moreover, we have $[\perp^{n_1+m_2+n_2}_{m_3}, \mathsf{id}_{m_3}] \cdot \gamma'^2 = [[\perp^{n_1}_{m_3}, [\perp^{n_2}_{m_3}, \mathsf{id}_{m_3}] \cdot [\beta, \mathsf{in}^{m_3}]^* \cdot \mathsf{in}^{m_2}], [\perp^{n_2}_{m_3}, \mathsf{id}_{m_3}] \cdot [\beta, \mathsf{in}^{m_3}]^*]$:

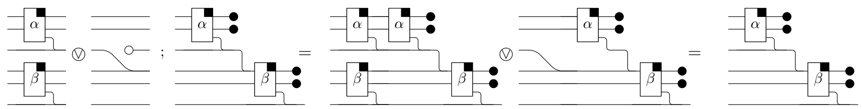


The first equation in the above follows by left distributivity. The second equation is a consequence of , the fact that this is the least morphism and . Finally,

$$[\perp^{n_1+m_2+n_2}_{m_3}, \mathsf{id}_{m_3}] \cdot \gamma'^3 = [[\perp^{n_1}_{m_3}, [\perp^{n_2}_{m_3}, \mathsf{id}_{m_3}] \cdot [\beta, \mathsf{in}^{m_3}]^* \cdot \mathsf{in}^{m_2}] \cdot [\alpha, \mathsf{in}^{m_2}]^*, [\perp^{n_2}_{m_3}, \mathsf{id}_{m_3}] \cdot [\beta, \mathsf{in}^{m_3}]^*] :$$



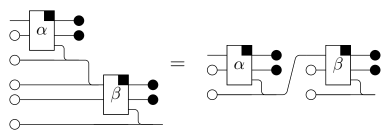
and $[\perp^{n_1+m_2+n_3}_{m_3}, \mathsf{id}_{m_3}] \cdot \gamma'^4 = [\perp^{n_1+m_2+n_3}_{m_3}, \mathsf{id}_{m_3}] \cdot \gamma'^3$:



Hence, $[\perp^{n_1+m_2+n_3}_{m_3}, \mathsf{id}_{m_3}] \cdot \gamma^* = [\perp^{n_1+m_2+n_3}_{m_3}, \mathsf{id}_{m_3}] \cdot \gamma'^* = [\perp^{n_1+m_2+n_3}_{m_3}, \mathsf{id}_{m_3}] \cdot \gamma'^3$ and, therefore, we have:

$$([\perp^{n_1+m_2+n_3}_{m_3}, \mathsf{id}_{m_3}]) \cdot \gamma^* \cdot \mathsf{in}^{n_1+m_2}_{n_1+m_2+n_2+m_3} \cdot \mathsf{in}^{m_1}_{n_1+m_2} =$$
$$[\perp^{n_1+m_2+n_3}_{m_3}, \mathsf{id}_{m_3}] \cdot \gamma'^3 \cdot \mathsf{in}^{n_1+m_2}_{n_1+m_2+n_2+m_3} \cdot \mathsf{in}^{m_1}_{n_1+m_2} =$$
$$[\perp^{n_2}_{m_3}, \mathsf{id}_{m_3}] \cdot [\beta, \mathsf{in}^{m_3}_{n_2+m_3}]^* \cdot \mathsf{in}^{m_2}_{n_2+m_3} \cdot [\perp^{n_1}_{m_2}, \mathsf{id}_{m_2}] \cdot [\alpha, \mathsf{in}^{m_2}_{n_1+m_2}]^* \cdot \mathsf{in}^{m_1}_{n_1+m_2} = r_2 \cdot r_1,$$

which can be depicted in terms of the diagrams as follows:



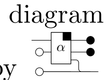
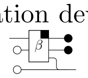
This proves the assertion. $\square$

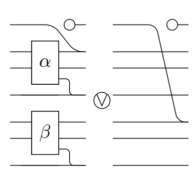
Proof. (Theorem 5.9) The proof of this theorem is divided into four parts.
Part 1. At first we show that any $[\mathcal{A}]$ map is in $\mathfrak{Reg}$. Let $a : m \multimap n \in [\mathcal{A}]$. Then:

$$a = [\perp, \mathsf{id}_n] \cdot [\mathsf{in}^n_{m+n} \cdot a, \mathsf{in}^n_{m+n}]^* \cdot \mathsf{in}^m_{m+n}.$$

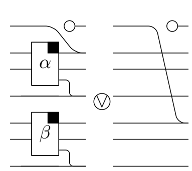
By Lemma 5.12 this means that $a \in \mathfrak{Reg}(m,n)$. In particular, this means that $\perp^m_n \in \mathfrak{Reg}(m,n)$.
Part 2. Let $r_1, r_2 \in \mathfrak{Reg}(m,p)$. In this part we show that $r_1 \vee r_2 \in \mathfrak{Reg}(m,p)$.
For r_1 and r_2 we have:

$$r_1 = [\perp, \mathsf{id}_m] \cdot [\alpha, \mathsf{in}^p_{n_1+p}]^* \cdot \mathsf{in}^m_{n_1+p} \text{ and } r_2 = [\perp, \mathsf{id}_m] \cdot [\beta, \mathsf{in}^p_{n_2+p}]^* \cdot \mathsf{in}^m_{n_2+p}$$

for $[\mathcal{A}]$-maps $\alpha : n_1 \multimap n_1 + p$, $\beta : n_2 \multimap n_2 + p$. We follow the string diagram notation developed above. Hence, just like before, we depict r_1 and r_2 by  and  respectively. Put $n = m + n_1 + p + n_2 + p$ and $\gamma : n \multimap n$ to be the morphism depicted by:

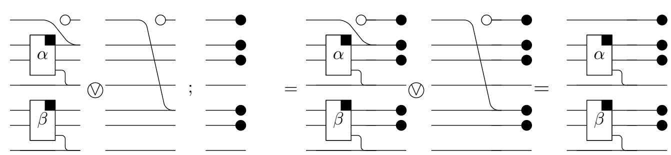


The morphism γ is an $\mathcal{A}$-map which satisfies $\gamma^* \geq \gamma' \geq \gamma \vee \mathsf{id}$, where γ' is given by:

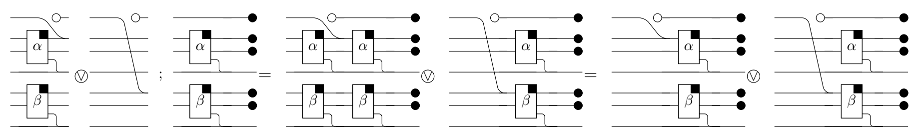


This means that $\gamma^* = \gamma'^*$. Moreover, we have:

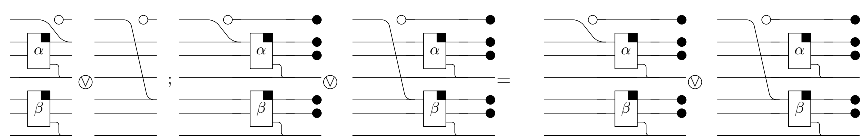
$$[\perp_n^m, [\perp_{n_1+p}^{n_1}, \mathsf{id}_p] + [\perp_{n_2+p}^{n_2}, \mathsf{id}_p]] \cdot \gamma' =$$
$$[\perp^m, [\perp_{n_1+p}^{n_1}, \mathsf{id}_p] \cdot [\alpha, \mathsf{in}_{n_1+p}^p]^* + [\perp_{n_2+p}^{n_2}, \mathsf{id}_p] \cdot [\beta, \mathsf{in}_{n_2+p}^p]^*] :$$



Additionally, $[\perp_n^m, [\perp_{n_1+p}^{n_1}, \mathsf{id}_p] + [\perp_{n_2+p}^{n_2}, \mathsf{id}_p]] \cdot \gamma'^2$ satisfies:



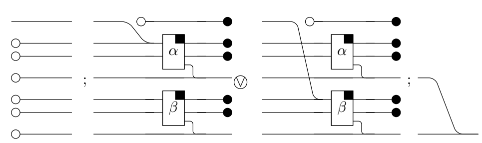
Finally, by left distributivity and right distributivity w.r.t. the base morphisms we get that $[\perp_n^m, [\perp_{n_1+p}^{n_1}, \mathsf{id}_p] + [\perp_{n_2+p}^{n_2}, \mathsf{id}_p]] \cdot \gamma'^3 = [\perp_n^m, [\perp_{n_1+p}^{n_1}, \mathsf{id}_p] + [\perp_{n_2+p}^{n_2}, \mathsf{id}_p]] \cdot \gamma'^2$:



Therefore, we have:

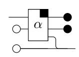
$$[\perp_n^m, [\perp_{n_1+p}^{n_1}, \mathsf{id}_p] + [\perp_{n_2+p}^{n_2}, \mathsf{id}_p]] \cdot \gamma^* = [\perp_n^m, [\perp_{n_1+p}^{n_1}, \mathsf{id}_p] + [\perp_{n_2+p}^{n_2}, \mathsf{id}_p]] \cdot \gamma'^* =$$
$$[\perp_n^m, [\perp_{n_1+p}^{n_1}, \mathsf{id}_p] + [\perp_{n_2+p}^{n_2}, \mathsf{id}_p]] \cdot \gamma'^2.$$

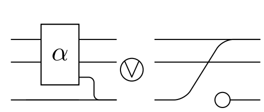
and by left distributivity and right distributivity w.r.t. the base morphisms the map given below equals $r_1 \vee r_2$:



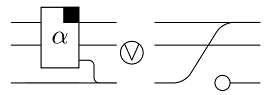
By the definition of $\mathfrak{Reg}(m, p)$ this means that $r_1 \vee r_2 \in \mathfrak{Reg}(m, p)$.

Part 3. Finally, let $r \in \mathfrak{Reg}(m, m)$. Here, we show that $r^* \in \mathfrak{Reg}(m, m)$.

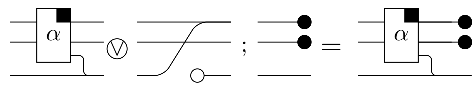
For sake of clarity of notation let $m_1 \triangleq m_2 \triangleq m$. We introduce m_1 and m_2 to denote the same number m in order to distinguish between m used in the domain and the codomain of r. Hence, $r : m_1 \multimap m_2$. By Lemma 5.12 we have $r = [\perp, \mathsf{id}_{m_2}] \cdot [\alpha, \mathsf{in}_{m'+m_2}^{m_2}]^* \cdot \mathsf{in}_{m'+m_2}^{m_1}$ for $\alpha : m' \multimap m' + m_2$ (i.e. $\alpha : m' \multimap m' + m$), where $m_1 = m \leq m'$. As before we depict r by . Consider $\gamma : m \multimap m$ defined by the following diagram:



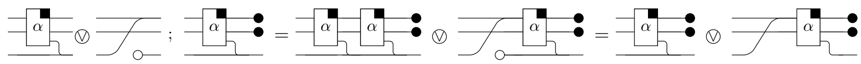
Note that γ is a $\mathcal{A}$-map which satisfies $\gamma^* \geq \gamma' \geq \gamma \vee \mathsf{id}$ for γ' given by:



We have $[\perp, \mathsf{id}_{m_2}] \cdot \gamma' = [\perp, \mathsf{id}_{m_2}] \cdot [\alpha, \mathsf{in}^{m_2}_{m'+m_2}]^*$ since:



Moreover, $[\perp, \mathsf{id}_{m_2}] \cdot \gamma'^2 =$



Hence, by right distributivity w.r.t. the base morphisms we have $[\perp, \mathsf{id}_{m_2}] \cdot \gamma'^2 \cdot \mathsf{in}^{m_2} = \mathsf{id}_{m_2} \vee r$ and $[\perp, \mathsf{id}_{m_2}] \cdot \gamma'^2 \cdot \mathsf{in}^{m'} = [\perp, \mathsf{id}_{m_2}] \cdot \gamma' \cdot \mathsf{in}^{m'}$. Since $[\perp, \mathsf{id}_{m_2}] \cdot \gamma' \cdot \mathsf{in}^{m_2} = \mathsf{id}_{m_2}$ we have:

$$[\perp, \mathsf{id}_{m_2}] \cdot \gamma'^2 \leq (\mathsf{id} \vee r) \cdot [\perp, \mathsf{id}_{m_2}] \cdot \gamma'.$$

Moreover,

$$[\perp, \mathsf{id}_{m_2}] \cdot \gamma'^2 = [r', \mathsf{id} \vee r] \geq [\perp^{m'}, \mathsf{id}_{m_2} \vee r] = (\mathsf{id} \vee r) \cdot [\perp^{m'}, \mathsf{id}_{m_2}],$$

for $r' \triangleq [\perp, \mathsf{id}_{m_2}] \cdot [\alpha, \mathsf{in}^{m_2}]^* \cdot \mathsf{in}^{m'}$. Note that $r' \cdot \mathsf{in}^{m_1}_{m'} = r$. Hence, to summarize:

$$(\mathsf{id} \vee r) \cdot [\perp, \mathsf{id}] = [\perp, \mathsf{id} \vee r] \leq [r', \mathsf{id} \vee r] \leq [\perp, \mathsf{id}_{m_2}] \cdot \gamma'^2 \leq (\mathsf{id} \vee r) \cdot [\perp^{m'}, \mathsf{id}_{m_2}] \cdot \gamma'.$$

Therefore, in general, by induction we get:

$$(\mathsf{id} \vee r)^n \cdot [\perp^{m'}, \mathsf{id}_{m_2}] \leq (\mathsf{id} \vee r)^{n-1} \cdot [r', \mathsf{id} \vee r] \overset{\triangle}{\leq} [\perp, \mathsf{id}_{m_2}] \cdot \gamma'^{2n} \leq (\mathsf{id} \vee r)^n \cdot [\perp^{m'}, \mathsf{id}_{m_2}] \cdot \gamma',$$

where the inequality marked with $(\triangle)$ follows from $(\mathsf{id} \vee r)^{n-1} \cdot [r', \mathsf{id} \vee r] = (\mathsf{id} \vee r)^{n-1} \cdot [\perp, \mathsf{id}] \cdot \gamma'^2 \leq (\mathsf{id} \vee r)^{n-2} \cdot [\perp, \mathsf{id}] \cdot \gamma'^2 \cdot \gamma'^2 \leq \ldots \leq [\perp, \mathsf{id}] \cdot \gamma'^{2n}$. This proves that

$$r^* \cdot [r', \mathsf{id} \vee r] \leq [\perp, \mathsf{id}_{m_2}] \cdot \gamma'^* \leq r^* \cdot [\perp^{m'}, \mathsf{id}_{m_2}] \cdot \gamma'$$

and therefore:

$$\begin{aligned}
&r^* = r^* \cdot [r, \mathsf{id} \vee r] \cdot \mathsf{in}^{m_2}_{m'+m_2} \leq [\perp, \mathsf{id}_{m_2}] \cdot \gamma'^* \cdot \mathsf{in}^{m_2}_{m'+m_2} \leq \\
&r^* \cdot [\perp^{m'}, \mathsf{id}_{m_2}] \cdot \gamma' \cdot \mathsf{in}^{m_2}_{m'+m_2} = r^* \text{ and} \\
&r^+ = r^* \cdot r = r^* [r', \mathsf{id} \vee r] \cdot \mathsf{in}^{m_1}_{m'+m_2} \leq \\
&[\perp, \mathsf{id}_{m_2}] \cdot \gamma'^* \cdot \mathsf{in}^{m_1}_{m'+m_2} \leq r^* \cdot [\perp^{m'}, \mathsf{id}_{m_2}] \cdot \gamma' \cdot \mathsf{in}^{m_1}_{m'+m_2} = \\
&r^* \cdot r = r^+.
\end{aligned}$$

This proves that $r^* \in \mathfrak{Reg}(m, m)$. This completes the proof of the fact that $\mathfrak{Reg}(\mathcal{A})$ is an ordered theory which:

(a) contains all maps from $\mathcal{A}$,
(b) admits finite suprema,
(c) endomorphisms are closed under $(-)^*$.

Part 4. Now, if $\mathfrak{Rat}(\mathcal{A})$ denotes the smallest subtheory of $\mathbb{T}$ which satisfies $(a)-(c)$ then it contains all morphisms from $\mathfrak{Reg}(\mathcal{A})$. Hence,

$$\mathfrak{Rat}(\mathcal{A}) = \mathfrak{Reg}(\mathcal{A}).$$

$\square$

Before we proceed with the proof of Theorem 5.10 we require one extra statement. Let us define:

$$\omega\mathfrak{Rat}(\mathcal{A})(n) \triangleq \{[r_1,\ldots,r_m]^\omega \cdot r \mid r \in \mathfrak{Reg}(n,m), r_i \in \mathfrak{Reg}(1,m) \text{ for } m < \omega\},$$

$$\omega\mathfrak{Reg}(\mathcal{A})(n) \triangleq \{||(\alpha,\mathfrak{F})||_\omega \cdot \mathsf{in}_m^n : n \multimap 0 \mid (\alpha,\mathfrak{F}) \text{ is } \mathcal{A}\text{-aut. with } \alpha : m \multimap m\}.$$

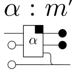
and note that $\omega\mathfrak{Rat}(\mathcal{A}) = \omega\mathfrak{Rat}(\mathcal{A})(1)$ and $\omega\mathfrak{Reg}(\mathcal{A}) = \omega\mathfrak{Reg}(\mathcal{A})(1)$. Additionally, the following holds.

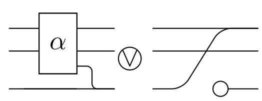
Lemma 5.13. *For any $r \in \mathfrak{Reg}(m,m)$ we have:*

$$r^\omega \in \omega\mathfrak{Reg}(\mathcal{A})(m).$$

Proof. For sake of clarity of notation let us define $m_1 \triangleq m$ and $m_2 \triangleq m$. We introduce m_1 and m_2 to denote the same number m for the same reason as in the proof of Theorem 5.9, i.e. to distinguish between the domain and codomain of r. Hence, $r : m_1 \multimap m_2$. By Lemma 5.12 we have

$$r = [\perp, \mathsf{id}_{m_2}] \cdot [\alpha, \mathsf{in}^{m_2}_{m'+m_2}]^* \cdot \mathsf{in}^{m_1}_{m'+m_2}$$

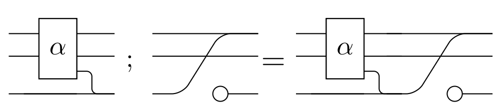
for $\alpha : m' \to m' + m_2$ (i.e. $\alpha : m' \to m' + m$), where $m_1 = m \le m'$. We depict r by . Consider $\gamma : m \multimap m$ defined by the following diagram:



and note that this is exactly the same transition map as in **Part 3.** of the proof of Theorem 5.9. Hence, by the properties listed in that part of the proof we have:

$$r^+ = [\perp, \mathsf{id}_{m_2}] \cdot \gamma^* \cdot \mathsf{in}^{m_1}_{m'+m_2}.$$

Moreover, let $\xi : m \multimap m$ be defined by



The map ξ is a $[\mathcal{A}]$-map which satisfies $\xi^* = \gamma^*$. Then

$$r^+ = [\perp, \mathsf{id}_{m_2}] \cdot \xi^* \cdot \mathsf{in}^{m_1}_{m'+m_2} = [\perp, \mathsf{id}_{m_2}] \cdot \xi^* \cdot \xi \cdot \mathsf{in}^{m_2}_{m'+m_2} = [\perp, \mathsf{id}_{m_2}] \cdot \xi^+ \cdot \mathsf{in}^{m_2}_{m'+m_2}.$$

Since $(\perp + \mathsf{id}) = \mathsf{in}^{m_2} \cdot [\perp, \mathsf{id}]$ we get that

$$\mathsf{in}^{m_2} \cdot r^+ = \mathsf{in}^{m_2} \cdot [\perp, \mathsf{id}_{m_2}] \cdot \xi^+ \cdot \mathsf{in}^{m_2}_{m'+m_2} = (\perp + \mathsf{id}_{m_2}) \cdot \xi^+ \cdot \mathsf{in}^{m_2}.$$

From the above we get:

$$r^\omega = (r^+)^\omega = ((\perp + \mathsf{id}_{m_2}) \cdot \xi^+)^\omega \cdot \mathsf{in}^{m_2}_{m'+m_2}.$$

The above identity follows from a more general property: given two coalgebras $\alpha : X \multimap X = X \to TX$ and $\beta : Y \multimap Y = Y \to TY$ and a Set-map $j : X \to Y$ which is a coalgebra homomomorphism (or, equivalently, $j^\sharp \cdot \alpha = \beta \cdot j^\sharp$ in $\mathcal{K}l(T)$) we have: $\alpha^\omega = \beta^\omega \cdot j^\sharp = \beta^\omega \circ j$. This property known as *uniformity* of $(-)^\omega$

w.r.t. the base maps (see e.g. [46]). In our setting, the fixpoint operator $(-)^\omega$ is uniform w.r.t. the base maps since the order of $\mathcal{K}l(T)$ is pointwise induced and since $\alpha^\omega = \bigwedge_{\kappa \in \mathsf{Ord}} (\lambda x.x \cdot \alpha)^\kappa(\top)$ (see Remark 2.9).

This completes the proof of the lemma. □

We are now ready to proceed with the proof of Theorem 5.10.

Proof. (Theorem 5.10) We have

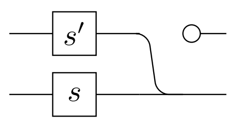
$$\omega\mathfrak{Rat}(\mathcal{A}) \supseteq \omega\mathfrak{Reg}(\mathcal{A})$$

as it is enough to note that

$$||(\alpha, \mathfrak{F})||_\omega \cdot i_n = (\mathfrak{f}_\mathfrak{F} \cdot \alpha^+)^\omega \cdot i_n = (\mathfrak{f}_\mathfrak{F} \cdot \alpha^+)^\omega \cdot \mathfrak{f}_\mathfrak{F} \cdot \alpha^+ \cdot i_n$$

and take $r = \mathfrak{f}_\mathfrak{F} \cdot \alpha^+ \cdot i_n$, $r_k = \mathfrak{f}_\mathfrak{F} \cdot \alpha^+ \cdot k_n$.

Conversely, let $r, r_i \in \mathfrak{Reg}(1, m)$. Put $s = [r_1, \ldots, r_m]$ and consider any regular morphism $s' \in \mathfrak{Reg}(m, m)$ such that $r = s' \cdot 1_m$. For sake of clarity of notation let $m_1 = m_2 = m$. Consider the map γ defined by:



Note that this map is a regular morphism, so the map $\gamma^\omega \cdot 1_{m_1+m_2}$ is ω-regular. Moreover, we get:

$$\gamma^\omega \cdot 1_{m_1+m_2} = \gamma^\omega \cdot \gamma \cdot 1_{m_1+m_2} = \gamma^\omega \cdot \mathsf{in}^{m_2} \cdot s' \cdot 1_{m_1} = [r_1, \ldots, r_m]^\omega \cdot r.$$

This completes the proof. □

6. Probabilistic automata

The main purpose of this section is to put probabilistic systems [3–5, 47, 48, 50] into the framework of Section 5. Here, we focus our attention on probabilistic automata which are akin to fully probabilistic systems from Example 2.2 [2,5,13,47].

6.1. **Preliminaries.** A *probabilistic automaton* is a tuple

$$(X, \Sigma, P : X \times \Sigma \times X \to [0,1], \mathfrak{F}),$$

where X is a set of *states*, Σ an *alphabet*, P is a *probability transition function*, i.e. a function such that for any $x \in X$ we have $\sum_{(a,y)\in\Sigma\times X} P(x,a,y) = 1$, and $\mathfrak{F} \subseteq X$ the set of *accepting states*.

For $C \subseteq X$ we define $P(x,a,C) \triangleq \sum_{y\in C} P(x,a,y)$. An *execution fragment* is a finite sequence $\mathfrak{s} = x_0 \stackrel{a_0}{\to} x_1 \stackrel{a_1}{\to} x_2 \ldots x_{n-1} \stackrel{a_{n-1}}{\to} x_n$ such that $P(x_i, a_i, x_{i+1}) > 0$. We define $first(\mathfrak{s}) = x_0$, $last(\mathfrak{s}) = x_n$, $length(\mathfrak{s}) = n$, $trace(\mathfrak{s}) = a_0 \ldots a_{n-1}$ and $P(\mathfrak{s}) = \prod_{i=0,\ldots,n-1} P(x_i, a_i, x_{i+1})$. An *execution* is an infinite sequence $\mathfrak{p} = x_0 \stackrel{a_0}{\to} x_1 \stackrel{a_1}{\to} x_2 \stackrel{a_2}{\to} \ldots$ with $P(x_i, a_i, x_{i+1}) > 0$.

Let $first(\mathfrak{p}) \triangleq x_0$, $trace(\mathfrak{p}) \triangleq a_0 a_1 \ldots$, $\mathfrak{p}^{(n)} \triangleq x_0 \stackrel{a_0}{\to} \ldots \stackrel{a_{n-1}}{\to} x_n$ and $\mathfrak{p}_n \triangleq x_n$. For an execution fragment $\mathfrak{s}$ of length n let $\mathfrak{s}\uparrow$ denote the set of all executions $\mathfrak{p}$ such that $\mathfrak{p}^{(n)} = \mathfrak{s}$.

Let $Exec(x)$ denote the set of all executions $\mathfrak{p}$ such that $first(\mathfrak{p}) = x$. Let $\Sigma(x)$ be the smallest sigma field on $Exec(x)$ which contains all sets $\mathfrak{s}\uparrow$ for any execution fragment $\mathfrak{s}$ with $first(\mathfrak{s}) = x$. Finally, let $\mathcal{Q}_x$ denote the unique probability measure on $\Sigma(x)$ such that $\mathcal{Q}_x(\mathfrak{s}\uparrow) = P(\mathfrak{s})$ for any execution fragment $\mathfrak{s}$ with $first(\mathfrak{s}) = x$. We will often drop the subscript and write $\mathcal{Q}$ instead of $\mathcal{Q}_x$ if the measure can be deduced from the context.

For $\Lambda \subseteq \Sigma^*$ and $C \subseteq X$ define $Exec(\Lambda, C)$ to be the set of all executions $\mathfrak{p} = x_0 \stackrel{a_0}{\to} x_1 \ldots$ for which there is n with $trace(\mathfrak{p}^{(n)}) \in \Lambda$ and $x_n \in C$ and consider $Exec(x, \Lambda, C) \triangleq Exec(\Lambda, C) \cap Exec(x)$. As stated in [5] the set $Exec(x, \Lambda, C)$ is $\Sigma(x)$-measurable. An execution $\mathfrak{p} = x = x_0 \stackrel{a_0}{\to} x_1 \stackrel{a_1}{\to} \ldots$ staring at x is called C-*accepting* provided that it visists C infinitely often, i.e. the set $\{i < \omega \mid x_i \in C\}$ is infinite. Let $AccExec(C)$ denote all C-accepting executions and let $AccExec(x, C) = AccExec(C) \cap Exec(x)$. The set $AccExec(x, C)$ is $\Sigma(x)$-measurable as

$$\begin{aligned} &AccExec(x, C) = \bigcap_{n \geq 0} \bigcup_{k \geq n} \{\mathfrak{p} \in Exec(x) \mid \mathfrak{p}_k \in C\} = \\ &\bigcap_{n \geq 0} \bigcup_{k \geq n} Exec(x, \{\sigma \in \Sigma^* \mid \text{ length of } \sigma = k\}, C) = \\ &\bigcap_{n \geq 0} Exec(x, \{\sigma \in \Sigma^* \mid \text{ length of } \sigma \geq n\}, C). \end{aligned}$$

A curious reader is referred to e.g. [2, 5] for more details on fully probabilistic systems and probability measures they induce.

The remaining part of this section will focus on finding a suitable setting in which we can model probabilistic automata and their (in)finite behaviour using the framework presented in the previous section.

Remark 6.1. Any probabilistic automaton can be modelled coalgebraically as a pair $(\alpha : X \to \mathcal{D}(\Sigma \times X), \mathfrak{F} \subseteq X)$, where $\mathcal{D}$ is the subdistribution monad from Example 2.2 and α is given by (see e.g. [9, 48]):

$$\alpha(x)(a, y) \triangleq P(x, a, y).$$

The Kleisli category associated with the monad $\mathcal{D}(\Sigma^* \times \mathcal{I}d + \Sigma^\omega)$[8] is order enriched with the hom-set ordering given for $f, g : X \to \mathcal{D}(\Sigma^* \times Y + \Sigma^\omega)$ by:

$$f \leq g \iff f(x)(y) \leq g(x)(y).$$

Although the ordering is $\omega\mathsf{Cpo}$-enriched, it does *not* admit arbitrary finite suprema [13, 26]. In other words, the setting is incompatible with the setting from the previous section. Hence, the remaining part of this section is focused on solving this issue based on ideas given in [13, 22].

6.2. **Choosing the right monad.** Here, we introduce a monad which is a good replacement of $\mathcal{D}$ and which satisfies the desired properties to make it suitable for modeling (in)finite behaviours of probabilistic automata. Inspired by [22] we consider the *continuous continuation monad* parametrized by the set $[0, 1]$ whose functorial part is defined for any set X by:

$$\mathfrak{D}X \triangleq (X \to [0, 1]) \to_\omega [0, 1],$$

where $P \to_\omega Q$ denotes the set of functions between two $\omega\mathsf{Cpo}$'s P and Q which preserve suprema of ω-chains. The identity maps and the composition in the Kliesli

[8]By applying the construction from Section 4 to $T = \mathcal{D}$ and $F = \Sigma \times \mathcal{I}d$ we obtain a monadic structure on the aforementioned functor.

category for $\mathfrak{D}$ are as follows:

$$\mathsf{id} : X \to (X \to [0,1]) \to_\omega [0,1]; x \mapsto \Delta_x, \text{ where } \Delta_x(d) \triangleq d(x).$$

$$g \cdot f : X \to (Z \to [0,1]) \to_\omega [0,1], \text{ where } (g \cdot f)(x)(d) \triangleq f(x)(y \mapsto g(y)(d))$$

with $f : X \to (Y \to [0,1]) \to_\omega [0,1]$ and $g : Y \to (Z \to [0,1]) \to_\omega [0,1]$. Note that any $d : X \to [0,1] \in \mathcal{D}X$ can be assigned a function $\nu_X(d) : (X \to [0,1]) \to_\omega [0,1]$ which maps any $d' : X \to [0,1]$ onto

$$\nu_X(d)(d') = \sum_{x \in X} d'(x) \cdot d(x). \tag{6.1}$$

It is not hard to see that this turns the family $\{\nu_X : \mathcal{D}X \to \mathfrak{D}X\}$ into a natural transformation $\nu : \mathcal{D} \implies \mathfrak{D}$ which is a monad morphism between the monads $\mathcal{D}$ and $\mathfrak{D}$. Additionally, it is easy to see that there is a natural ordering of arrows in $\mathcal{K}l(\mathfrak{D})$ which share the same domain and codomain. Indeed, for $f, g : X \to \mathfrak{D}Y$ we have:

$$f \leq g \iff f(x)(d) \leq g(x)(d) \text{ for any } x \in X \text{ and } d : Y \to [0,1].$$

This turns the Kleisli category for the monad $\mathfrak{D}$ into an order enriched category. Moreover, the following theorem is true.

Lemma 6.2. *The order enrichment of $\mathcal{K}l(\mathfrak{D})$ is pointwise induced, every hom-set of $\mathcal{K}l(\mathfrak{D})$ is a complete lattice and the category is $\omega\mathsf{Cpo}$-enriched and left distributive.*

Proof. The fact that the partial order is a complete lattice order is a direct corollary from the definition of $\mathfrak{D}$. We will now prove that Kleisli category for $\mathfrak{D}$ is $\omega\mathsf{Cpo}$-enriched. Take any ascending chain $\{f_i : X \to \mathfrak{D}Y\}_{i<\omega}$ of morphisms and note that

$$\begin{aligned}
&[g \cdot (\bigvee_i f_i)](x)(d) = (\bigvee_i f_i)(x)(y \mapsto g(y)(d)) = \\
&\bigvee_i f_i(x)(y \mapsto g(y)(d)) = \bigvee_i g \cdot f_i(x)(d) \text{ and} \\
&[(\bigvee_i f_i) \cdot h](z)(d) = h(z)(x \mapsto \bigvee_i f_i(x)(d)) \overset{\diamond}{=} \\
&\bigvee_i h(z)(x \mapsto f_i(x)(d)) = \bigvee_i (h \cdot f_i)(z)(d),
\end{aligned}$$

where the equality marked with $(\diamond)$ follows from the fact that $\mathfrak{D}X$ consists of functions $(X \to [0,1]) \to_\omega [0,1]$ that preserve ω-chains. In order to see left distributivity consider morphisms $\{f_i\}_{i \in I} : X \to \mathfrak{D}Y$ and $g : Y \to \mathfrak{D}Z$ and note that

$$\begin{aligned}
&[g \cdot (\bigvee_i f_i)](x)(d) = (\bigvee_i f_i)(x)(y \mapsto g(y)(d)) = \\
&\bigvee_i f_i(x)(y \mapsto g(y)(d)) = \bigvee_i g \cdot f_i(x)(d).
\end{aligned}$$

This proves the assertion. $\square$

From the above it follows that $\mathcal{K}l(\mathfrak{D})$ satisfies (A)-(D) from Section 5.

6.2.1. *The monad* $\mathfrak{D}(\Sigma^*\times\mathcal{I}d+\Sigma^\omega)$. By instantiating the construction from Section 4 for $T=\mathfrak{D}$ and $F=\Sigma\times\mathcal{I}d$ we obtain the monad $\mathfrak{D}(\Sigma^*\times\mathcal{I}d+\Sigma^\omega)$. We explicitly spell out the formula for the identity maps and the composition in $\mathcal{K}l(\mathfrak{D}(\Sigma^*\times\mathcal{I}d+\Sigma^\omega))$ as it will be used throughout the remaining part of this section. For any set X the identity map id is given by:

$$\begin{aligned}&\mathsf{id}: X\to(\Sigma^*\times X+\Sigma^\omega\to[0,1])\to_\omega[0,1],\\&\mathsf{id}(x)(d)=d(\varepsilon,x).\end{aligned}$$

Moreover, for $f:X\to\mathfrak{D}(\Sigma^*\times Y+\Sigma^\omega)$ and $g:Y\to\mathfrak{D}(\Sigma^*\times Z+\Sigma^\omega)$ the map $g\cdot f:X\to\mathfrak{D}(\Sigma^*\times Z+\Sigma^\omega)$ is:

$$\begin{aligned}&g\cdot f: X\to(\Sigma^*\times Z+\Sigma^\omega\to[0,1])\to_\omega[0,1],\\&(g\cdot f)(x)(d)=f(x)((\sigma,y)\mapsto g(y)(d_{|\sigma})\text{ and }v\mapsto d(v)),\end{aligned}$$

where for $d:\Sigma^*\times Z+\Sigma^\omega\to[0,1]$ and $\sigma\in\Sigma^*$ the map $d_{|\sigma}:\Sigma^*\times Z+\Sigma^\omega\to[0,1]$ is given by $d_{|\sigma}(\tau,z)=d(\sigma\tau,z)$ for $\tau\in\Sigma^*$, $z\in Z$ and $d_{|\sigma}(v)=d(\sigma v)$ for $v\in\Sigma^\omega$.

The following statement is a direct consequence of the definition of the monad $\mathfrak{D}(\Sigma^*\times\mathcal{I}d+\Sigma^\omega)$ and the properties of $\mathfrak{D}$ from the previous subsection.

Theorem 6.3. *$\mathcal{K}l(\mathfrak{D}(\Sigma^*\times\mathcal{I}d+\Sigma^\omega))$ satisfies (A)-(D) from Section 5.*

6.3. **(In)finite behaviour for probabilistic automata.** Let $(\alpha,\mathfrak{F})$ be a pair as in Remark 6.1 and consider

$$(\widehat{\alpha}:X\stackrel{\alpha}{\to}\mathcal{D}(\Sigma\times X)\stackrel{\nu_{\Sigma\times X}}{\to}\mathfrak{D}(\Sigma\times X)\hookrightarrow\mathfrak{D}(\Sigma^*\times X+\Sigma^\omega),\mathfrak{F}),$$

where ν is given by (6.1). We see that the map $\widehat{\alpha}$ can be viewed as an endomorphism in the Kleisli category for the monad $\mathfrak{D}(\Sigma^*\times\mathcal{I}d+\Sigma^\omega)$.

The rest of this section is devoted to presenting some properties of $||(\widehat{\alpha},\mathfrak{F})||$ and $||(\widehat{\alpha},\mathfrak{F})||_\omega$ for the pair $(\widehat{\alpha},\mathfrak{F})$. As it turns out below the values $||(\widehat{\alpha},\mathfrak{F})||(x)$ and $||(\widehat{\alpha},\mathfrak{F})||_\omega(x)$ encode probabilities of certain events from $\Sigma(x)$.

6.3.1. *Finite behaviour.* The following lemma is a direct consequence of the definition of $\widehat{\alpha}^*,\widehat{\alpha}^+$ and the composition in $\mathcal{K}l(\mathfrak{D}(\Sigma^*\times\mathcal{I}d+\Sigma^\omega))$.

Lemma 6.4. *We have:*

$$\begin{aligned}&\widehat{\alpha}^+,\widehat{\alpha}^*: X\to(\Sigma^*\times X+\Sigma^\omega\to[0,1])\to_\omega[0,1];\\&\widehat{\alpha}^+(x)(d)=\sum_{(a,y)\in\Sigma\times X}\widehat{\alpha}^*(y)(d_{|a})\cdot\alpha(x)(a,y)\\&\widehat{\alpha}^*(x)(d)=\max\{d(\varepsilon,x)\ ,\sum_{(a,y)\in\Sigma\times X}\widehat{\alpha}^*(y)(d_{|a})\cdot\alpha(x)(a,y)\}.\end{aligned}$$

For $\mathfrak{F}\subseteq X$ we have:

$$\begin{aligned}&\mathfrak{f}_{\mathfrak{F}}: X\to(\Sigma^*\times X+\Sigma^\omega\to[0,1])\to_\omega[0,1],\\&\mathfrak{f}_{\mathfrak{F}}(x)(d)=\begin{cases}d(\varepsilon,x)&\text{for }x\in\mathfrak{F},\\0&\text{otherwise.}\end{cases}\end{aligned}$$

Hence, the map $!\cdot\mathfrak{f}_{\mathfrak{F}}:X\to(\Sigma^*\times 1+\Sigma^\omega\to[0,1])\to_\omega[0,1]$ is given by:

$$(!\cdot\mathfrak{f}_{\mathfrak{F}})(x)(d)=\begin{cases}d(\varepsilon,1)&\text{for }x\in\mathfrak{F},\\0&\text{otherwise.}\end{cases}$$

Therefore, the finite behaviour of $(\widehat{\alpha},\mathfrak{F})$ is given by

$$\begin{aligned}
&||(\widehat{\alpha},\mathfrak{F})||: X \to (\Sigma^* \times 1 + \Sigma^\omega \to [0,1]) \to_\omega [0,1]\\
&||(\widehat{\alpha},\mathfrak{F})||(x)(d) = (! \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \widehat{\alpha}^*)(x)(d) =\\
&\widehat{\alpha}^*(x)\left((\sigma,y) \mapsto \left\{\begin{array}{cc} d(\sigma,1) & y \in \mathfrak{F},\\ 0 & \text{otherwise}\end{array}\right. \text{ and } v \mapsto 0\right).
\end{aligned}$$

For $\Lambda \subseteq \Sigma^*$ and $\sigma \in \Sigma$ put $\Lambda/\sigma \triangleq \{\tau \mid \tau\sigma \in \Lambda\}$ and consider a mapping $\chi_\Lambda : \Sigma^* \times 1 + \Sigma^\omega \to [0,1]; \chi_\Lambda(x) = \left\{\begin{array}{cc} 1 & x = (\sigma,1) \text{ and } \sigma \in \Lambda\\ 0 & \text{otherwise}\end{array}\right.$. If we define a function $\chi_{\mathfrak{F},\Lambda} : \Sigma^* \times X + \Sigma^\omega \to [0,1]$ by:

$$\chi_{\mathfrak{F},\Lambda}(\tau,y) = \left\{\begin{array}{cc} 1 & y \in \mathfrak{F} \text{ and } \tau \in \Lambda,\\ 0 & \text{otherwise}\end{array}\right. \quad \text{and } \delta_{\mathfrak{F},\Lambda}(v) = 0.$$

then

$$\begin{aligned}
&||(\widehat{\alpha},\mathfrak{F})||(x)(\chi_\Lambda) = \widehat{\alpha}^*(x)(\delta_{\mathfrak{F},\Lambda}) =\\
&\left\{\begin{array}{cc} 1 & \varepsilon \in \Lambda \text{ and } x \in \mathfrak{F}\\ \sum_{(a,y)\in\Sigma\times X} \widehat{\alpha}^*(y)(\delta_{\mathfrak{F},\Lambda/a}) \cdot \alpha(x)(a,y) & \text{otherwise.}\end{array}\right.
\end{aligned}$$

A careful analysis of the formulae from [5,13] describing the value $\mathcal{Q}(Exec(x,\Lambda,C))$ allows us to see that the following statement holds:

Theorem 6.5. *We have:*

$$||(\widehat{\alpha},\mathfrak{F})||(x)(\chi_\Lambda) = \mathcal{Q}(Exec(x,\Lambda,\mathfrak{F})),$$

i.e. $||(\widehat{\alpha},\mathfrak{F})||(x)(\chi_\Lambda)$ *is the probability of reaching a state in* $\mathfrak{F}$ *from* x *via an execution fragment whose trace is a member of* Λ.

6.3.2. *Infinite behaviour.* Let us focus on the infinite behaviour of $(\widehat{\alpha},\mathfrak{F})$ introduced in the previous section given by

$$||(\widehat{\alpha},\mathfrak{F})||_\omega = (\mathfrak{f}_{\mathfrak{F}} \cdot \widehat{\alpha}^+)^\omega : X \to \mathfrak{D}(\Sigma^\omega).$$

Take any execution fragment $\mathfrak{s}$ which starts at x and note that the image of $\mathfrak{s}\uparrow \subseteq Exec(x)$ under the function $trace : Exec(x) \to \Sigma^\omega$ satisfies the following.

Theorem 6.6. *We have:*

$$\begin{aligned}
&||(\widehat{\alpha},\mathfrak{F})||_\omega : X \to (\Sigma^\omega \to [0,1]) \to_\omega [0,1],\\
&||(\widehat{\alpha},\mathfrak{F})||_\omega(x)(\chi_{trace(\mathfrak{s}\uparrow)}) = \mathcal{Q}(\mathfrak{s}\uparrow \cap AccExec(x,\mathfrak{F})).
\end{aligned}$$

Proof. We will first show that $||(\widehat{\alpha},\mathfrak{F})||_\omega(x)(\chi_{\Sigma^\omega}) = \mathcal{Q}(AccExec(x,\mathfrak{F}))$. Indeed, since $AccExec(x,\mathfrak{F}) = \bigcap_{n\geq 0}\bigcup_{k\geq n}\{\mathfrak{p} \in Exec(x) \mid \mathfrak{p}_k \in \mathfrak{F}\}$ we have:

$$\mathcal{Q}(AccExec(x,\mathfrak{F})) = \lim_{n\to\infty} \mathcal{Q}(A^x_n),$$

where $A^x_n \triangleq \bigcup_{k\geq n}\{\mathfrak{p} \in Exec(x) \mid \mathfrak{p}_k \in \mathfrak{F}\} = Exec(x, \{\sigma \in \Sigma^* \mid \text{ length of } \sigma \geq n\}, \mathfrak{F})$ is a descending chain of $\Sigma(x)$-measurable sets. Let us now consider a family of maps $\{G_n : X \to (\Sigma^\omega \to [0,1]) \to_\omega [0,1]\}_n$ defined inductively as follows:

$$G_0(x)(d) = 1 \text{ and } G_{n+1} = G_n \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \widehat{\alpha}^+.$$

Note that the sequence $\{G_n(x)(d)\}_n$ is descending for any fixed $x \in X$ and d and that

$$||(\widehat{\alpha},\mathfrak{F})||_\omega(x)(d) = \lim_{n\to\infty} G_n(x)(d).$$

We will now show that $G_n(x)(\chi_{\Sigma^\omega}) = \mathcal{Q}(A_n^x)$ for $n \geq 1$. For G_1 we have:

$$G_1(x)(\chi_{\Sigma^\omega}) = (G_0 \cdot \mathfrak{f}_{\mathfrak{F}} \cdot \widehat{\alpha}^+)(x)(\chi_{\Sigma^\omega}) = \tag{6.2}$$

$$\widehat{\alpha}^+(x)((\sigma, y) \mapsto G_0 \cdot \mathfrak{f}_{\mathfrak{F}}(y)(\chi_{\Sigma^\omega|\sigma}) \text{ and } v \mapsto \chi_{\Sigma^\omega}(v)) = \tag{6.3}$$

$$\widehat{\alpha}^+(x)((\sigma, y) \mapsto G_0 \cdot \mathfrak{f}_{\mathfrak{F}}(y)(\chi_{\Sigma^\omega}) \text{ and } v \mapsto \chi_{\Sigma^\omega}(v)). \tag{6.4}$$

In the above

$$G_0 \cdot \mathfrak{f}_{\mathfrak{F}}(y)(\chi_{\Sigma^\omega}) = \mathfrak{f}_{\mathfrak{F}}(y)((\tau, z) \mapsto G_0(z)(\chi_{\Sigma^\omega|\tau}) \text{ and } v \mapsto \chi_{\Sigma^\omega}(v)) =$$
$$\mathfrak{f}_{\mathfrak{F}}(y)((\tau, z) \mapsto 1 \text{ and } v \mapsto 1) = \begin{cases} 1 & y \in \mathfrak{F} \\ 0 & \text{otherwise.} \end{cases}$$

Hence, if we continue with (6.4) we get:

$$\widehat{\alpha}^+(x)\left((\sigma, y) \mapsto \begin{cases} 1 & y \in \mathfrak{F} \\ 0 & \text{otherwise} \end{cases} \text{ and } v \mapsto 1\right) = \mathcal{Q}(A_1^x).$$

If we now assume by induction that the statement holds for some $n > 1$ then

$$G_n \cdot \mathfrak{f}_{\mathfrak{F}}(y)(\chi_{\Sigma^\omega}) = \mathfrak{f}_{\mathfrak{F}}(y)((\tau, z) \mapsto \mathcal{Q}(A_n^z) \text{ and } v \mapsto 1) = \begin{cases} \mathcal{Q}(A_n^y) & y \in \mathfrak{F} \\ 0 & \text{otherwise.} \end{cases}$$

Hence, by following a similar reasoning to the one applied to G_1 we get:

$$G_{n+1}(x)(\chi_{\Sigma^\omega}) = \widehat{\alpha}^+\left((\sigma, y) \mapsto \begin{cases} \mathcal{Q}(A_n^y) & y \in \mathfrak{F} \\ 0 & \text{otherwise} \end{cases} \text{ and } v \mapsto 1\right) = \mathcal{Q}(A_{n+1}^x).$$

Assume $\mathfrak{s} = x = x_0 \stackrel{a_0}{\to} \ldots \stackrel{a_{n-1}}{\to} x_n = x'$. Then $\chi_{trace(\mathfrak{s}\uparrow)} = \chi_{\{a_0 a_1 \ldots a_{n-1}\} \times \Sigma^\omega}$ and for $\sigma \in \Sigma^*$ of length less than or equal to n we have $\chi_{\{a_0 a_1 \ldots a_{n-1}\} \times \Sigma^\omega|\sigma}$ is equal $\chi_{\{a_k \ldots a_{n-1}\} \times \Sigma^\omega}$ if $\sigma = a_0 \ldots a_{k-1}$ and it is the constantly equal to zero function otherwise. This observation together with the fact that

$$\mathcal{Q}(\mathfrak{s}\uparrow \cap AccExec(x, \mathfrak{F})) = \mathcal{Q}(AccExec(x, \mathfrak{F}) \mid \mathfrak{s}\uparrow) \cdot Q(\mathfrak{s}\uparrow) =$$
$$\mathcal{Q}(AccExec(x', \mathfrak{F})) \cdot \mathcal{Q}(\mathfrak{s}\uparrow)$$

and induction allows us to prove $||(\widehat{\alpha}, \mathfrak{F})||_\omega(x)(\chi_{trace(\mathfrak{s}\uparrow)}) = \mathcal{Q}(\mathfrak{s}\uparrow \cap AccExec(x, \mathfrak{F}))$. □

Remark 6.7. By Theorem 6.3 probabilistic automata can be put into the framework of Section 5. Hence, Kleene theorems hold for any suitable choice of $\mathcal{A}$. In particular, we may take $\mathcal{A}$ to be the least set of maps containing all morphisms of the form

$$n \stackrel{\alpha}{\to} \mathcal{D}(\Sigma \times n) \stackrel{\nu_{\Sigma \times n}}{\to} \mathfrak{D}(\Sigma \times n) \hookrightarrow \mathfrak{D}(\Sigma^* \times n + \Sigma^\omega)$$

and satisfying the properties listed in the beginning of Subsection 5.4.

6.4. **Summary.** The purpose of this section was to put probabilistic automata into a monadic framework from Section 5 and reason about their (in)finite behaviours. We achieved this by introducing the continuous continuation monad $\mathfrak{D}$ and viewing probabilistic automata transition maps as coalgebras

$$X \to \mathfrak{D}(\Sigma^* \times X + \Sigma^\omega).$$

The monad $\mathfrak{D}(\Sigma^* \times \mathcal{I}d + \Sigma^\omega)$ gives rise to a Kleisli category which satisfies (A)-(D) from Section 5 making it possible to consider finite and infinite behaviours of automata taken into consideration. We proved that the behaviour maps encode probabilities of certain events from the execution space. These probabilities were attained without changing the underlying category: the type monad and the automata taken into consideration are Set-based. Additionally, Theorem 6.6 suggests that our infinite behaviour with BAC for probabilistic automata is similar to the one presented in [50]. However, in *loc. cit.* the base category for probabilistic systems was the category of measurable spaces and measurable functions. Hence, by Remark 6.7, Kleene theorems can be instantiated in our setting directly, but it is not possible to do so in the setting from [50].

7. Summary

The purpose of this paper was to develop a coalgebraic (categorical) framework to reason about abstract automata and their finite and infinite behaviours satisfying BAC. We achieved this goal by constructing a monad suitable to handle the types of behaviours we were interested in and defining them in the right setting. A natural and direct consequence of this treatment was Theorem 5.9 and Theorem 5.10, i.e. a (co)algebraic characterization of regular and ω-regular behaviour for systems whose type is a Set-based monad satisfying some additional properties. Our theory of finite and infinite behaviour for abstract automata has been successfully instantiated on: non-deterministic automata, tree automata and probabilistic automata.

Future work. Given our natural characterization of coalgebraic (ω-)regular languages we ask if it is possible to characterize it in terms of a preimage of a subset of a finite algebraic structure. Especially, considering the fact that by Theorem 5.3 the pair of hom-sets $(\mathbb{T}(n,n),\mathbb{T}(n,0))$ equipped with suitable operations resembles a Wilke algebra used in the algebraic characterization of these languages (see e.g. [41] for details).

Related work. The first coalgebraic take on ω-languages was presented in [16], where authors put deterministic Muller automata with Muller acceptance condition into a coalgebraic framework. Our work is related to a more recent paper [50], where Urabe *et al.* give a coalgebraic framework for modelling behaviour with Büchi acceptance condition for (T,F)-systems. The main ingredient of their work is a solution to a system of equations which uses least and greatest fixpoints. This is done akin to Park's [40] classical characterization of ω-languages via a system of equations. In our paper we also use least and greatest fixpoints, however, the operators we consider are the two natural types of operators $(-)^* = \mu x.\mathsf{id} \vee x \cdot (-)$ and $(-)^\omega = \nu x.x\cdot(-)$ which generalize the language operators $(-)^*$ and $(-)^\omega$ known from the classical theory of regular and ω-regular languages. The definitions of behaviours of an automaton are presented in terms of simple expressions involving Kleisli composition and the above operators. This allows us to state and prove generic Kleene theorems for (ω-)regular input which was not achieved in [50] and (in our opinion) would be difficult to obtain in that setting. To summarize, the major differences between our work and [50] are the following:

- we use the setting of systems with internal moves (i.e. coalgebras over a monad) to discuss infinite behaviour with BAC, which is given in terms of a simple expression using $(-)^*$ and $(-)^\omega$ in the Kleisli category,
- we provide the definition of (in)finite behaviours of a system and build a bridge between regular and ω-regular behaviours by characterizing them on a categorical level in terms of the Kleene theorems.

Abstract finite automata have already been considered in the computer science literature in the context of Lawvere iteration theories with analogues of Kleene theorems stated and proven (see e.g. [8, 18–20]). Some of these results seem to be presented using a slightly different language than ours (see Theorem 5.9 and e.g. [8, Theorem 1.4]). We decided to state Theorem 5.9 the way we did, in order to make a direct generalization of the classical Kleene theorem for regular input and to give a coalgebraic interpretation which is missing in [8, 18–20]. We should also mention that the infinite behaviour with BAC was defined in *loc. cit.* only for a very specific type of theories (i.e. the matricial theories over an algebra with an infinite iteration operator), which do not encompass e.g. non-deterministic Büchi tree automata and their infinite tree languages.

References

[1] J. Adámek, M. Haddadi, and S. Milius. Corecursive algebras, corecursive monads and Bloom monads. *Logical Methods in Computer Science*, 10(3), 2014.

[2] C. Baier, B. Engelen, and M. E. Majster-Cederbaum. Deciding bisimilarity and similarity for probabilistic processes. *J. Comput. Syst. Sci.*, 60(1):187–231, 2000.

[3] C. Baier and M. Grosser. Recognizing ω-regular languages with probabilistic automata. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*, LICS '05, pages 137–146, Washington, DC, USA, 2005. IEEE Computer Society.

[4] C. Baier, M. Grösser, and N. Bertrand. Probabilistic omega-automata. *J. ACM*, 59(1):1:1–1:52, Mar. 2012.

[5] C. Baier and H. Hermanns. Weak bisimulation for fully probabilistic processes. In *CAV*, pages 119–130, 1997.

[6] M. Barr and C. Wells. *Toposes, Triples and Theories*. 2002.

[7] J. Beck. Distributive laws. In B. Eckmann, editor, *Seminar on Triples and Categorical Homology Theory*, pages 119–140, Berlin, Heidelberg, 1969. Springer Berlin Heidelberg.

[8] S. Bloom and Z. Ésik. *Iteration Theories. The Equational Logic of Iterative Processes.* Monographs in Theoretical Computer Science. Springer, 1993.

[9] F. Bonchi, S. Milius, A. Silva, and F. Zanasi. Killing epsilons with a dagger: A coalgebraic study of systems with algebraic label structure. *Theoretical Computer Science*, 604:102–126, 2015.

[10] T. Brengos. On coalgebras with internal moves. In M. M. Bonsangue, editor, *Proc. CMCS*, Lecture Notes in Computer Science, pages 75–97. Springer, 2014.

[11] T. Brengos. Weak bisimulation for coalgebras over order enriched monads. *Logical Methods in Computer Science*, 11(2):1–44, 2015.

[12] T. Brengos. A coalgebraic take on regular and ω-regular behaviour for systems with internal moves. In S. Schewe and L. Zhang, editors, *29th International Conference on Concurrency Theory (CONCUR 2018)*, volume 118 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:18, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[13] T. Brengos, M. Miculan, and M. Peressotti. Behavioural equivalences for coalgebras with unobservable moves. *Journal of Logical and Algebraic Methods in Programming*, 84(6):826–852, 2015.

[14] T. Brengos and M. Peressotti. A uniform framework for timed automata. In J. Desharnais and R. Jagadeesan, editors, *27th International Conference on Concurrency Theory (CONCUR 2016)*, volume 59 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[15] J. R. Büchi. *On a decision method in restricted second order arithmetic*, pages 425–435. Springer New York, New York, NY, 1990.

[16] V. Ciancia and Y. Venema. Stream automata are coalgebras. In *Proc. CMCS*, volume 7399 of *Lecture Notes in Computer Science*, pages 90–108, 2012.

[17] C. Cîrstea. Generic infinite traces and path-based coalgebraic temporal logics. *Electr. Notes Theor. Comput. Sci.*, 264(2):83–103, 2010.

[18] Z. Ésik and T. Hajgató. Iteration grove theories with applications. In *Proc. Algebraic Informatics*, volume 5725 of *Lecture Notes in Computer Science*, pages 227–249. Springer, 2009.

[19] Z. Ésik and W. Kuich. *A unifying Kleene theorem for weighted finite automata*, pages 76–89. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[20] Z. Ésik and W. Kuich. *Modern Automata Theory*, page 222. 2013.

[21] B. Fong and D. I. Spivak. *Seven sketches in compositionality: An invitation to applied category theory.* arxiv:1803.05316, 2018.

[22] S. Goncharov and D. Pattinson. Coalgebraic weak bisimulation from recursive equations over monads. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, editors, *Proc. ICALP*, volume 8573 of *Lecture Notes in Computer Science*, pages 196–207. Springer, 2014.

[23] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata Logics, and Infinite Games: A Guide to Current Research*, page 392. Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[24] H. P. Gumm. *Elements of the general theory of coalgebras.* LUATCS 99, Rand Afrikaans University, 1999.

[25] I. Hasuo. Generic forward and backward simulations. In *Prof. CONCUR*, volume 4137 of *Lecture Notes in Computer Science*, pages 406–420, 2006.

[26] I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3(4), 2007.

[27] J. E. Hopcroft, R. Motwani, Rotwani, and J. D. Ullman. *Introduction to Automata Theory, Languages and Computability.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2000.

[28] M. Hyland and J. Power. The category theoretic understanding of universal algebra: Lawvere theories and monads. *Electronic Notes in Theoretical Computer Science*, 172:437–458, 2007.

[29] B. Jacobs. Trace semantics for coalgebras. *Electr. Notes Theor. Comput. Sci.*, 106:167–184, 2004.

[30] B. Jacobs, A. Silva, and A. Sokolova. Trace semantics via determinization. In *Proc. CMCS*, volume 7399 of *Lecture Notes in Computer Science*, pages 109–129, 2012.

[31] B. Khoussainov and A. Nerode. *Automata Theory and Its Applications.* Birkhauser Boston, Inc., USA, 2001.

[32] D. J. King and P. Wadler. Combining monads. In *Functional Programming*, Workshops in Computing, pages 134–143. Springer, 1992.

[33] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, Princeton, NJ, 1956.

[34] A. Kock. Strong functors and monoidal monads. *Archiv der Mathematik*, 23(1):113–120, 1972.

[35] F. W. Lawvere. Functorial semantics of algebraic theories. *Proc. Nat. Acad. Sci. U.S.A.*, 50:869–872, 1963.

[36] S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer-Verlang New York, 1978.

[37] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[38] E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.

[39] P. S. Mulry. Lifting theorems for Kleisli categories. In S. D. Brookes, M. G. Main, A. Melton, M. W. Mislove, and D. A. Schmidt, editors, *Mathematical Foundations of Programming Semantics, 9th International Conference, New Orleans, LA, USA, April 7-10, 1993, Proceedings*, volume 802 of *Lecture Notes in Computer Science*, pages 304–319. Springer, 1993.

[40] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Theoretical Computer Science*, pages 167–183, Berlin, Heidelberg, 1981. Springer Berlin Heidelberg.

[41] J.-E. Pin and D. Perrin. *Infinite Words: Automata, Semigroups, Logic and Games*, page 538. Elsevier, 2004.

[42] J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.

[43] D. Sangiorgi. *Introduction to Bisimulation and Coinduction.* Cambridge University Press, 2011.

[44] P. Selinger. A survey of graphical languages for monoidal categories. volume 813, pages 289–355, 2011.

[45] A. Silva and B. Westerbaan. A coalgebraic view of ϵ-transitions. In R. Heckel and S. Milius, editors, *Proc. CALCO*, volume 8089 of *Lecture Notes in Computer Science*, pages 267–281. Springer, 2013.

[46] A. Simpson and G. Plotkin. Complete axioms for categorical fixed-point operators. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science*, LICS '00, pages 30–, Washington, DC, USA, 2000. IEEE Computer Society.

[47] A. Sokolova. Probabilistic systems coalgebraically: A survey. *Theoretical Computer Science*, 412(38):5095–5110, 2011.

[48] A. Sokolova, E. P. de Vink, and H. Woracek. Coalgebraic weak bisimulation for action-type systems. *Scientific Annals of Computer Science*, 19:93–144, 2009.

[49] N. Urabe and I. Hasuo. Coalgebraic infinite traces and kleisli simulations. *CoRR*, abs/1505.06819, 2015.

[50] N. Urabe, S. Shimizu, and I. Hasuo. Coalgebraic trace semantics for buechi and parity automata. In J. Desharnais and R. Jagadeesan, editors, *27th International Conference on Concurrency Theory (CONCUR 2016)*, volume 59 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

E-mail address: `t.brengos@mini.pw.edu.pl`

Faculty of Mathematics and Information Science, Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warszawa, Poland