Stream Automata Are Coalgebras

Vincenzo Ciancia, Yde Venema

Stream automata are coalgebras *

Vincenzo Ciancia, Yde Venema

Institute of Logic, Language and Computation, University of Amsterdam

Abstract. Stream automata (also called ω-automata) and ω-regular languages are of paramount importance in Computer Science and Logic. A coalgebraic treatment of these structures has not been given yet. We study a simple two-sorted setting where deterministic Muller automata can be cast as coalgebras, so that coalgebraic bisimilarity coincides with language equivalence. From this characterisation, we derive concise and natural decision procedures for complementation, union, intersection, and equivalence check.

1 Introduction

Finite state automata and their variants are among the most fundamental structures in Computer Science. Their applications range over a wide domain, including for instance controlling digital circuits, representing languages in formal language theory, parsing expressions in compilers, model-checking of software systems. Deterministic finite state automata (DFAs) are possibly the simplest finite memory machines, although they can represent infinite sets; part of their usefulness is that set-theoretical operations, such as complementation, union and intersection of languages can be implemented directly on DFAs.

DFAs are key examples of coalgebras [8]. The crucial observation is that they admit a universal model, namely the *final coalgebra*. States of the final coalgebra represent the regular languages. The most direct consequence of the coalgebraic presentation is that language equivalence coincides with bisimilarity, yielding a coinductive proof method to decide language equivalence. Applications range over a wide domain, including the probabilistic setting [1], and weighted automata [3].

Stream automata (or ω-automata, see [7] for a comprehensive reference) accept languages of infinite words. Besides being a fundamental tool in the study of modal fixpoint logics, they are of vital importance in the field of automated program verification. There are various kinds of stream automata in the literature, among which we mention Büchi, parity and Muller automata, in their deterministic and non-deterministic variants. With the exception of deterministic Büchi automata, all these models accept the same class of languages, namely the ω-*regular* ones. The accepting condition is usually expressed in terms of the set $Inf(\rho)$ of states that are passed infinitely often in a run ρ of the automaton on an infinite word. A coalgebraic formulation of stream automata would

* Work supported by the second author's VICI grant 639.073.501 of the Netherlands Organization for Scientific Research (NWO).

endow the theory of ω-regular languages with coinduction, a universal model, and possible generalisations, similarly to DFAs. However, such a characterisation is non-obvious, since the accepting condition of stream automata has non-local flavour: it depends on the whole structure of an automaton, not on single states.

In this paper we show that nevertheless it is possible to represent stream automata as coalgebras. We take Muller automata ($DMAs$) as the subject of our investigation, as they can express all the ω-regular languages, even if they are deterministic. The starting point is a well-known characterisation of ω-regular languages by their *ultimately periodic fragment*, that is, the set of infinite words of the form sl^ω, consisting of the concatenation of a finite word s and a non-empty word l which is repeated infinitely many times. This result simplifies the theory of ω-regular languages, as one just needs to study languages of ultimately periodic words. Such words admit a finite syntactic representation as *lassos*, that is, pairs (s, l) representing the word sl^ω.

In [4], Calbrix, Nivat and Podelski proposed to describe the lasso languages corresponding to Büchi automata using automata over finite words, whose alphabet is augmented with a special symbol "$". The dollar syntactically represents the moment when an ultimately periodic word sl^ω starts to repeat l. Well-formed automata only accept strings with a single occurrence of $, which ought not to be the last symbol in a word. We continue this program by taking a *two-sorted* coalgebraic perspective, that is, we define coalgebras whose state spaces live in a category of two-sorted structures. This improves on the existing characterisation as there is no need to augment the alphabet with a special symbol. The "switch" to the periodic part of a ultimately periodic word is semantically represented by changing the sort; thus no special conditions on accepted words are needed.

Technically, we present two results. First, we represent Muller automata as coalgebras for an endo-functor, called Ω, in the functor category $\mathtt{Set}^2$. The representation is such that language equivalence coincides with the coalgebraic formulation of bisimilarity. This is done in Section 3. The inclusion of DMAs into Ω-coalgebras is proper. More precisely, we call *balanced* those coalgebras whose states correspond in a suitable sense to ω-regular languages. The ones whose image in the final coalgebra is finite are (up-to bisimilarity) the image of DMAs into Ω-coalgebras. Balanced coalgebras are in a special form, which makes them accept *bisimulation invariant* lasso languages. Our second result is described in Section 4. Therein, we identify two conditions, namely *circularity* and *coherence*, that are equivalent to bisimulation invariance, and we characterise the coalgebras corresponding to DMAs as those that are circular and coherent, and have finite image in the final coalgebra.

The most basic application of coalgebras, in the presence of a final object, is a generalised notion of partition refinement, yielding minimisation up-to coalgebraic bisimilarity. When the states are finite, one obtains a decision procedure; this is the case for the coalgebras obtained from DMAs. We spell out the procedure in Section 5; in the same section, we go further in considering balanced coalgebras as acceptors of lasso languages, by showing that boolean operations are defined in a particularly simple and natural way.

2 Preliminaries

Here we recall the notion of *deterministic Muller automaton* and ω-regular language. For more background on these topics, the reader is referred to [7].

Remark 1 (Notation). We often omit parentheses in function (and functor) application, writing e.g. fx in place of $f(x)$. We write $X \sim Y$ to signify that X is isomorphic to Y. We denote the set of natural numbers with the symbol ω. Throughout the paper, we fix a finite set C called the *alphabet*. We let C^* be the free monoid over C, called the set of *finite words*, or just words, whose neutral element is the *empty word* ϵ. The length of a word u is denoted with $|u|$. We let $C^+ = C^* \setminus \{\epsilon\}$ be the set of non-empty words, and C^ω be the set of *infinite words* or *streams* over C, that is, the function space $\omega \to C$. Concatenation, denoted by juxtaposition, is defined over finite words as the binary operation of the monoid, and extended to $u\alpha$, for u finite and α infinite, in the obvious way. For $u \in C^*$, we let u^ω be the stream that repeats u infinitely many times. For $f : X \to X^C$ a (deterministic transition) function, we write $x \xrightarrow{c}_f y$ for $y = f(x)(c)$, omitting the subscript f when clear from the context. The two projections of a binary product are written π_1, π_2. For $f : X \to Y \times Z$, we let $f^1 = \pi_1 \circ f$ and $f^2 = \pi_2 \circ f$ (f is the pairing of f^1 and f^2).

Definition 1 (DMA). *A* deterministic Muller automaton *(DMA in the following) is a tuple $(X, \delta, \mathcal{M})$ such that X is a finite set of* states*, $\delta : X \to X^C$ is the* transition function*, $\mathcal{M} \subseteq \mathcal{P}(X)$ is the* Muller acceptance condition*.*

We adopt a formulation of DMAs that does not include initial states, thus we will speak of the language *of a given state*, not of an automaton; in other words every state can be chosen as initial. The definition of accepted language is also simplified by the deterministic setting.

Definition 2 ($\widehat{\delta}$, δ°). *Given $\delta : X \to X^C$, we define by induction on the structure of words the functions $\widehat{\delta} : X \to X^{C^*}$ as $\widehat{\delta}x(\epsilon) = x$, $\widehat{\delta}x(cu) = (\widehat{\delta}(\delta xc)u)$, and $\delta^\circ : X \to \mathcal{P}(X)^{C^*}$ as $\delta^\circ x(\epsilon) = \emptyset$, $\delta^\circ x(cu) = \{\delta xc\} \cup \delta^\circ(\delta xc)u$.*

In words, $\widehat{\delta}x(u)$ is the state reached by the automaton after it processed the word u, starting at position x, and $\delta^\circ x(u)$ is the set of states the automaton passed along the way (the starting state x not being included).

Definition 3 (Inf). *For $x \in X$, and α a stream, we let $Inf(x, \alpha) \subseteq X$ represent the set of states that are traversed infinitely often when following α starting from x, that is, $y \in Inf(x, \alpha)$ if and only if for each u, β such that $\alpha = u\beta$ there are u', β' such that $\beta = u'\beta'$, and $\widehat{\delta}x(uu') = y$.*

Definition 4 ($\mathcal{L}(x)$). *For $x \in X$, we let $\mathcal{L}(x)$ denote the* language *of x, defined as the set of streams $\{\alpha \in C^\omega | Inf(x, \alpha) \in \mathcal{M}\}$. We call a set of streams* ω-regular *if it is $\mathcal{L}(x)$ for x a state in a DMA.*

3 Muller automata are coalgebras

DMAs can be turned into coalgebras, in such a way that behavioural equivalence coincides with language equivalence. In Section 3.1 we introduce a simple, abstract construction, which is meant to be propaedeutic to a more concrete definition. The latter is provided in Section 3.5. The prerequisites are: the fact that the *looping language* of a state in a DMA is regular, which we prove in Section 3.2; the framework of *dependent coalgebras*, presented in Section 3.3; a coalgebraic characterisation of *regular languages of non-empty words*, given in Section 3.4. We assume basic prerequisites on coalgebras, and refer the reader to [9] for further details.

3.1 An abstract characterisation

Definition 5 ($Loop(x)$). *We call* looping language of $x \in X$ *the set* $Loop(x) = \{u \in C^+ | u^\omega \in \mathcal{L}x\}$.

Consider the set $\mathcal{P}(C^+)$ (that is, the set of sets of non-empty words over the *finite* set C). We will treat its elements as *observations*, or *colours* associated to each state of a coalgebra. We are going to describe DMAs as coalgebras for the functor $\mathtt{T}(X) = X^C \times S$, where $S = \mathcal{P}(C^+)$. We have that $\mathtt{T}$ is ω-accessible (finitary) and weak-pullback preserving. Thus a final coalgebra exists, and behavioural equivalence (that is, identifiability by coalgebra morphisms) can be equivalently defined as bisimilarity (connectability via a bisimulation).

Definition 6 ($\simeq$). *Let X be a set and (X, f) be a $\mathtt{T}$-coalgebra. Behavioural equivalence $\simeq \subseteq X \times X$ is the kernel of the unique morphism f into the final coalgebra, that is $\simeq = \{(x, y) \mid f(x) = f(y)\}$.*

Remark 2. Equivalently, call $R \subseteq X \times X$ a *bisimulation* if and only if, whenever $x\,R\,y$, then $\pi_2(f(x)) = \pi_2(f(y))$, and, for all $c \in C$, $\pi_1(f(x))(c)\,R\,\pi_1(f(y))(c)$. It is easy to see that two states are behaviourally equivalent if and only if they are bisimilar, that is, linked by some bisimulation. Behavioural equivalence can also be defined as a relation over two different coalgebras, using cospans of morphisms; the two conditions can be obtained from each other using coproducts of coalgebras.

Definition 7 (Coalgebra of a DMA). *Given a DMA $(X, \delta, \mathcal{M})$, the coalgebra map $f : X \to X^C \times \mathcal{P}(C^+)$ is defined as $f(x) = (\delta(x), Loop(x))$.*

The transition structure of the automaton is preserved in the translation. For each state, its looping language is observed. Roughly, we may say that the ability to observe the looping language of a state in a coalgebra plays the same role of the acceptance condition of a DMA.

In Proposition 1 below, we prove that language equivalence coincides with bisimilarity. For this we need to introduce *lasso* languages and a well-known characterisation theorem for ω-regular languages.

Definition 8 ($UP(\mathcal{L})$). *An* ultimately periodic word *is a stream $\alpha \in C^\omega$ of the form $s(l^\omega)$. For $\mathcal{L}$ a set of streams, we denote with $UP(\mathcal{L})$ the subset of its ultimately periodic words, called the* ultimately periodic fragment *of $\mathcal{L}$.*

Definition 9 ($Lasso(\mathcal{L})$). *A* lasso *is a pair (s, l), with $s \in C^*$ and $l \in C^+$ (the* spoke *and the* loop *of the lasso). A lasso (s, l) is a syntactic representation for the ultimately periodic word sl^ω. For $\mathcal{L}$ a set of streams, call $Lasso(\mathcal{L})$ the set $\{(s, l)|sl^\omega \in UP(\mathcal{L})\}$. For x a state in a Muller automaton, let $Lasso(x) = Lasso(UP(\mathcal{L}(x)))$.*

Fact 1 below is well known (see Fact 1, [4]), and needed for Proposition 1.

Fact 1 (Lasso iff. ω-regular) *The sets $Lasso(x)$ and $\mathcal{L}(x)$ uniquely determine each other: we have $\mathcal{L}(x) = \mathcal{L}(y)$ if and only if $Lasso(x) = Lasso(y)$.*

Proposition 1 (Language equivalence is bisimilarity). *Let $(X, \delta, \mathcal{M})$ be a DMA, and (X, f) be the corresponding coalgebra. For $x, y \in X$, we have $\mathcal{L}(x) = \mathcal{L}(y)$ if and only if $x \simeq y$.*

Proof. (Proposition 1) By Fact 1, it suffices to prove that $Lasso(x) = Lasso(y)$ if and only if $x \simeq y$. First we prove that $R = \{(x, y) \in X \times X | Lasso(x) = Lasso(y)\}$ is a bisimulation with respect to f. Observe that $Lasso(x)$ uniquely determines $Loop(x) = \{u \in c^+ | (\epsilon, u) \in Lasso(x)\}$. Thus $(x, y) \in R \Rightarrow \pi_2 fx = \pi_2 fy$. On the other hand, $Lasso(x) = Lasso(y) \Rightarrow$ [Fact 1] $\mathcal{L}(x) = \mathcal{L}(y) \Rightarrow \forall c \in C.\mathcal{L}((\delta x)c) = \mathcal{L}((\delta y)c) \Rightarrow$ [Fact 1] $\forall c.Lasso((\delta x)c) = Lasso((\delta y)c) \Rightarrow \forall c.((\pi_1 fx)c, (\pi_1 fy)c) \in R$. Thus, $Lasso(x) = Lasso(y) \Rightarrow x \simeq y$. For the other direction, assume $Lasso(x) \neq Lasso(y)$. Then there is a lasso (s, l) such that (without loss of generality) $sl^\omega \in \mathcal{L}(x) \setminus \mathcal{L}(y)$. We have $l^\omega \in \mathcal{L}(\widehat{\delta} xs) \wedge l^\omega \notin \mathcal{L}(\widehat{\delta} ys) \Rightarrow l \in Loop(\widehat{\delta} xs) \wedge l \notin Loop(\widehat{\delta} ys) \Rightarrow \pi_2 f(\widehat{\delta} xs) \neq \pi_2 f(\widehat{\delta} ys) \Rightarrow \widehat{\delta} xs \not\simeq \widehat{\delta} ys$. For all $s \in C^*$, it is easy to prove by induction on the length of s that $x \simeq y \Rightarrow \widehat{\delta} xs \simeq \widehat{\delta} ys$, so we may conclude that $x \not\simeq y$.

3.2 *Loop*(x) is regular

Here we prove that, given a state x of a DMA, it is possible to construct a *deterministic finite automaton* (DFA) that accepts $Loop(x)$. The construction is similar to the one provided in [4] for Büchi automata.

Definition 10 (DFA for $Loop(x)$). *Fix a DMA $(X, \delta, \mathcal{M})$. For $x \in X$, let (T, γ, F_x, t^0) be a DFA having:*

- *states in $T = (X \times \mathcal{P}(X))^X$;*
- *transition function $\gamma : T \to T^C$, defined as $\gamma(t)(c) = \lambda y.(\delta(\pi_1(t\, y))c, \pi_2(t\, y) \cup \{\delta(\pi_1(t\, y))c\})$.*
- *initial state $t^0 = \lambda y.(y, \emptyset)$;*
- *accepting states in $F_x \subseteq T$ defined as follows. Consider $t \in (X \times \mathcal{P}(X))^X$. Define a sequence of states indexed by ω: $x_0 = x$, $x_{i+1} = \pi_1(t(x_i))$. Since X is finite, there is a least index k such that $x_k \in \{x_h | h < k\}$, so there is $i < k$ such that $x_i = x_k$. Let $t \in F_x$ if and only if $\bigcup_{i \leq j < k} \pi_2(t(x_j)) \in \mathcal{M}$.*

In Definition 10, notice that only F_x depends on x, and that it is never the case that $t^0 \in F_x$, thus the language of t^0 does not include the empty word. Intuitively, the DFA we define consumes a word u by following it in parallel starting from all the states of the automaton. Moreover, in each of these parallel runs, the states that are traversed are collected. Therefore states of the system are tuples in $(X \times \mathcal{P}(X))^X$ indexed by the elements of X. At each index x we can find a pair (z, s), consisting of a reached state z and a set of traversed states s. This idea is formalised by Lemma 1 (whose proof is a routine induction on u) and its Corollary 1.

Lemma 1. *Consider the DFA of Definition 10. For each $u \in C^*$, $t \in T$, and $z \in X$ with $tz = (x_1, s_1)$, we have*

$$(\widehat{\gamma}tu)z = (\widehat{\delta}x_1u, s_1 \cup \delta^\circ x_1 u).$$

Corollary 1. *For each $u \in C^*$ and $x_1 \in X$, we have*

$$(\widehat{\gamma}t^0u)x_1 = (\widehat{\delta}x_1u, \delta^\circ x_1 u).$$

Lemma 1 is concerned with the structure of the DFA for x, that is T, γ, t^0. The idea behind the definition of F_x is to compute $\mathit{Inf}(x, u^\omega)$ for u a finite word. This is clarified by Lemma 2.

Lemma 2. *For u a finite non-empty word, consider the state $t = \widehat{\gamma}t^0u$, and the objects $(x_h)_{h\in\omega}$, i, k from Definition 10. We have $\bigcup_{i\leq j<k} \pi_2(t(x_j)) = \mathit{Inf}(x, u^\omega)$.*

Proof. By Corollary 1 and the hypothesis, for each h, we have $\widehat{\delta}x_hu = x_{h+1}$, and $\pi_2(t(x_h)) = \delta^\circ x_h u$. Since the sequence (x_h) loops, by the automaton being deterministic we get the thesis.

Finally, as a direct consequence of Lemma 2, we get regularity of $\mathit{Loop}(x)$.

Proposition 2 ($\mathit{Loop}(x)$ is regular). *Given a DMA $(X, \delta, \mathcal{M})$ and $x \in X$, the language of finite words $\mathit{Loop}(x)$ is regular, and it is accepted by the automaton of Definition 10.*

Proof. Importing definitions from Definition 10, by Lemma 2, a non-empty word is accepted from t in the DFA if and only if $\mathit{Inf}(x, u^\omega) \in \mathcal{M}$, that is $u^\omega \in \mathit{Loop}(x)$.

Remark 3. In case we are dealing with *parity* automata rather than with Muller automata, the above construction can be somewhat simplified. Let $A = (X, \delta, \Omega)$ be a parity automaton, with $\Omega : X \to \omega$ the priority map. Now we can still base the DFA for $\mathit{Loop}(y)$ (where $y \in X$) on the idea of considering parallel runs from each state of A. We no longer have to collect the full set of states that were passed during each run, but only the *maximal priority* of these states. In other words, the carrier set of the automaton recognizing the looping language for y can be based on the set $T_\Omega = (X \times \mathit{Ran}(\Omega))^X$, where $\mathit{Ran}(\Omega) \subseteq \omega$ is the (finite) range of Ω. We leave further details to the reader.

3.3 Dependent coalgebras

We now develop a kind of *dependent* behavioural equivalence. Behavioural equivalence depends on *observations* on states. In our setting, observations are in turn states of another system. Dependent behavioural equivalence dictates that these observations are behavioural equivalent in turn, instead of just equal. This notion is captured by coalgebras in a category where objects are not merely sets, but rather two-sorted structures.

Let 2 be the two-elements set $\{1, 2\}$ considered a discrete category. The category $\mathtt{Set}^2$ is the category of functors from 2 to $\mathtt{Set}$ and their natural transformations. In other words, an object X can be represented as a pair (X_1, X_2). We shall call the sets X_1 and X_2 the first and second sorts. An arrow $f : X \to Y$ is a pair of functions $f_1 : X_1 \to Y_1$, $f_2 : X_2 \to Y_2$.

Consider two functors $\mathtt{T}_1, \mathtt{T}_2 : \mathtt{Set} \to \mathtt{Set}$. We define the *dependent* functor $\mathtt{T}_1 \triangleleft \mathtt{T}_2 : \mathtt{Set}^2 \to \mathtt{Set}^2$.

Definition 11 ($\mathtt{T}_1 \triangleleft \mathtt{T}_2$). *The functor $\mathtt{T}_1 \triangleleft \mathtt{T}_2$ is defined on objects as*

$$(\mathtt{T}_1 \triangleleft \mathtt{T}_2 X)_1 = (\mathtt{T}_1 X_1) \times X_2 \qquad (\mathtt{T}_1 \triangleleft \mathtt{T}_2 X)_2 = \mathtt{T}_2 X_2$$

The action on arrows is

$$(\mathtt{T}_1 \triangleleft \mathtt{T}_2 (f : X \to Y))_1 = \mathtt{T}_1 f_1 \times f_2 \qquad (\mathtt{T}_1 \triangleleft \mathtt{T}_2 f)_2 = \mathtt{T}_2 f_2$$

A $\mathtt{T}_1 \triangleleft \mathtt{T}_2$-coalgebra is nothing but a pair of sets (X_1, X_2), together with a $\mathtt{T}_1$-coalgebra on X_1, a $\mathtt{T}_2$-coalgebra on X_2, and a map from X_1 to X_2. To see this formally, note that a functor $X : 2 \to \mathtt{Set}$ is just a pair of sets (X_1, X_2). For (X, f) a $\mathtt{T}_1 \triangleleft \mathtt{T}_2$-coalgebra, f is a natural transformation, therefore a pair of maps $(f_1 : X_1 \to (\mathtt{T}_1 X_1) \times X_2, f_2 : X_2 \to \mathtt{T}_2 X_2)$. Moreover, f_1 can be decomposed (by the universal property of the product) in the pairing of $f_1^1 : X_1 \to \mathtt{T}_1 X_1$ and $f_1^2 : X_1 \to X_2$.

A coalgebra morphism $h : (X, f) \to (Y, g)$ is in turn a pair of functions (h_1, h_2) giving rise to two separate commutativity requirements

$$((\mathtt{T}_1 h_1) \times h_2) \circ f_1 = g_1 \circ h_1 \qquad \mathtt{T}_2 h_2 \circ f_2 = g_2 \circ h_2$$

As f_1 and g_1 are pairings, we see that h being a coalgebra morphism amounts to commutativity of the three diagrams in Figure 1

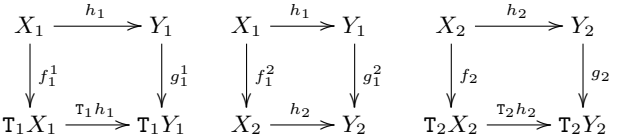

Fig. 1. Commutativity requirements of dependent behavioural equivalence.

Throughout the paper, we are mostly interested into equivalence in the first sort of a coalgebra in $\mathtt{Set}^2$. However, we remark that behavioural equivalence in $\mathtt{Set}^2$ is a two sorted relation, which we denote by $\simeq_i$ for $i \in \{1, 2\}$. We shall omit the index i when clear from the context.

Definition 12 (Final $\mathtt{T}_1 \triangleleft \mathtt{T}_2$-coalgebra). *Assume that $\mathtt{T}_2$ has a final coalgebra (Z_2, z_2). Suppose that the functor $\mathtt{F}(X) = \mathtt{T}_1(X) \times Z_2$ has a final coalgebra (Z_1, z_1) in turn. Define the $\mathtt{T}_1 \triangleleft \mathtt{T}_2$-coalgebra (Z, z) as $Z = (Z_1, Z_2)$, $z = (z_1, z_2)$.*

Proposition 3 (The final coalgebra is final). *Assume the coalgebras from Definition 12. For (X, f) a $\mathtt{T}_1 \triangleleft \mathtt{T}_2$-coalgebra, define a morphism $h : (X, f) \to (Z, z)$ as follows. The arrow h_2 is the unique morphism from (X_2, f_2) to (Z_2, z_2). The arrow h_1 is the unique morphism from the $\mathtt{F}$-coalgebra (X_1, f_1') to (Z_1, z_1), where $f_1'(x) = (f_1^1(x), h_2(f_2^1(x)))$. Such h is unique as a coalgebra morphism.*

Proof. First, observe that h fulfils the requirements in Figure 1. The rightmost diagram commutes since h_2 is a coalgebra homomorphism. The leftmost and middle diagrams commute simultaneously since h_1 is an $\mathtt{F}$-coalgebra homomorphism.

3.4 Regular languages of non-empty words

As we discussed, $Loop(x)$ is a regular language in C^+. To describe DMAs as coalgebras, we need a precise characterisation of such regular languages of non-empty words. This can be done by a variant of the coalgebraic description of classical DFAs. In doing so, one omits the initial state in the spirit of coalgebraic presentations of automata, where one speaks of the language of a given state instead of the language of an automaton.

Definition 13 (DFAs accepting non-empty words). *Consider the endofunctor in $\mathtt{Set}$ $\mathtt{D}(X) = (X \times 2)^C$. Let (X, γ, F) a DFA having unspecified initial state, with states in X, transition function $\gamma : X \to X^C$ and final states $F \subseteq X$. Define the $\mathtt{D}$-coalgebra (X, f_γ), where $f_\gamma xc = (\gamma xc, 1)$ if $\gamma xc \in F$, $f_\gamma xc = (\gamma xc, 2)$ otherwise.*

States of finite $\mathtt{D}$-coalgebras are intended to represent regular languages of non-empty words; thus it is sensible to define acceptance on them.

Definition 14 (Language of x). *For $u \in C^+$, (X, f) a $\mathtt{D}$-coalgebra and $x \in X$, define the language $\mathcal{L}_\mathtt{D}(x)$, by induction on the length of u. If $u = c$ has length one, we let $u \in \mathcal{L}_\mathtt{D}(x) \iff \pi_2(fxc) = 1$. If $u = cv$, with $v \in C^+$, we let $u \in \mathcal{L}_\mathtt{D}(x) \iff v \in \mathcal{L}_\mathtt{D}(\pi_1(fxc))$.*

The following proposition is easy to prove.

Proposition 4 (Languages of non-empty words). *Given an uninitialised DFA D and the corresponding coalgebra $\mathcal{C}$, a non-empty word u is is accepted from x in $\mathcal{C}$ if and only if it is accepted from x in D.*

Corollary 2. *Each regular language of non-empty words is accepted by some state in a finite* $\mathtt{D}$*-coalgebra.*

A construction in the reverse direction, obtaining a DFA accepting non-empty words from a finite $\mathtt{D}$-coalgebra is possible. This proves that the states of the final coalgebra, that are images of states of finite $\mathtt{D}$-coalgebras, are isomorphic to the regular languages of non-empty words. We omit the details of the construction. It is worth noting that, just like in the standard case, the structure map of the final $\mathtt{D}$-coalgebra computes a form of "derivative" of a language; however, since the languages in question are of non-empty words, the derivative behaves accordingly, thus being equal to the standard derivative of languages, where, in addition, the empty word is excluded from the result. In the following we will sometimes blur the distinction between DFAs of non-empty words and the corresponding $\mathtt{D}$-coalgebras.

3.5 Deterministic Muller automata as coalgebras

The characterization of language equivalence given by Section 3.1 can be refined, by incorporating in the definition of coalgebraic bisimilarity the ability to check that two looping languages are the same. This can be done using the characterisation of languages of non-empty words in Section 3.4.

First, we define the functor Ω, using the construction for dependent functors (Definition 11) and the functor $\mathtt{D}$ for DFAs of non-empty words (Definition 13).

Definition 15 (Functor Ω). *We define the functor* $\Omega : \mathtt{Set}^2 \to \mathtt{Set}^2$ *as*

$$\Omega = (-)^C \triangleleft \mathtt{D}.$$

For an intuition about Ω-coalgebras, recall that a $\mathtt{T}_1 \triangleleft \mathtt{T}_2$-coalgebra f is composed of of a $\mathtt{T}_1$-coalgebra (X_1, f_1^1), a $\mathtt{T}_2$-coalgebra (X_2, f_2), and a map f_2^1 from X_1 to X_2. Roughly, in our case, the $\mathtt{T}_1$-coalgebra can be thought of as the structure of a DMA, while the $\mathtt{T}_2$-coalgebra can be used to represent the DFAs that accept $\mathit{Loop}(x)$ for $x \in X_1$. The map f_2^1 provides a link between the two. Before showing our first result, we give a characterisation of behavioural equivalence in an Ω-coalgebra (X, f).

Proposition 5 (Dependent bisimilarity). *For* (X, f) *an* Ω*-coalgebra,* $x, y \in X_1$ *are behaviourally equivalent if and only if:*

1. $\mathcal{L}_{\mathtt{D}}(f_1^2(x)) = \mathcal{L}_{\mathtt{D}}(f_1^2(y))$*;*
2. *For all* $c \in C$*,* $f_1^1 xc$ *and* $f_1^1 yc$ *are behaviourally equivalent in turn.*

Proof. Two states x and y are behaviourally equivalent if and only if there is a coalgebra (Y, g) and a morphism $h : (X, f) \to (Y, g)$ obeying to the requirements in Figure 1. The leftmost square is turned into the usual coinductive back-and-forth condition on the transition function f_1^1 (Condition 2). The middle square imposes that $f_1^2(x)$ and $f_1^2(y)$ are identified by h_2. By the rightmost square, this

is equivalent to behavioural equivalence of these two states. By Proposition 4 we know that behavioural equivalence in the second sort is language equivalence (Condition 1).

In Definition 16 below, we introduce the Ω-coalgebra that corresponds to a given DMA. The correspondence is then made precise by Theorem 1, showing that the proposed translation turns language equivalence into behavioural equivalence.

Definition 16 (Ω-coalgebra of a DMA). *Let $A = (X_1, \delta, \mathcal{M})$ be a DMA. For $x \in X_1$, let $(T_x, \gamma_x, F_x, t^0_x)$ be the DFA[1] for x from Definition 10. Let (T_x, f_x) be the* D*-coalgebra of (T_x, γ_x, F_x) (from Definition 14). We define the Ω-coalgebra*

$$\mathcal{C}(A) = ((X_1, X_2), f) \qquad \text{with } X_2 = \coprod_{x \in X_1} T_x$$

where the natural transformation $f = (f_1, f_2)$ is defined by

$$f_1(x) = (\delta(x), t^0_x) \qquad f_2 = f_x.$$

As we discussed, an Ω-coalgebra is a two-sorted structure. In the first sort of $\mathcal{C}(\mathcal{A})$ the structure of the DMA A is replicated. The second sort corresponds to a DFA accepting non-empty words, having initial states that accept the looping languages of the states of A. The map f^2_1 associates to each state x in the main sort precisely the initial state of the DFA for $\mathit{Loop}(x)$. Thus, Proposition 1 is recovered in coalgebraic form.

Theorem 1 (Language equivalence is dependent bisimilarity). *Let A be a DMA. Behavioural equivalence in the first sort of $\mathcal{C}(A)$ coincides with language equivalence in A. That is, for every two states x, y in X_1 we have that*

$$x \simeq y \iff \mathcal{L}(x) = \mathcal{L}(y).$$

Proof. Consider two states x and y in X_1. By Proposition 5, they are behaviourally equivalent if and only if the usual back-and-forth condition holds, where the "observations" $x' = \pi_2(f_1(x))$ and $y' = \pi_2(f_1(y))$ are not required to be equal, but rather to be bisimilar (then, coinductively, this is extended to all states reachable from x and y). By Proposition 4, x' and y' are bisimilar if and only if they are language equivalent. By Proposition 2, x' and y' accept respectively $\mathit{Loop}(x)$ and $\mathit{Loop}(y)$. By Proposition 1 we get the thesis.

[1] Even though only the accepting states depend on x, it is convenient in Definition 16 to consider the DFA for each x as a distinguished structure. We consider all the T_x as non-intersecting sets to simplify the notation for elements of their coproduct. This can be concretely achieved by letting e.g. $T_x = \{x\} \times T$.

4 A characterising property

Ω-coalgebras can be considered acceptors of *lasso languages*. We call *balanced* those whose states accept lasso languages corresponding to ω-regular languages (in a sense made precise in the following). In this section we show that these form a proper subclass, and provide a characterising property.

We have seen that Muller automata can be encoded as Ω-coalgebras, and the encoding turns language equivalence into behavioural equivalence. Thus, each ω-regular language can be represented as a state in the first sort of some Ω-coalgebra. Such a state represents a set of lassos, according to Definition 17 below, where we use the notation for the pairing from Remark 1.

Definition 17 (Loops and lassos in a coalgebra). *Given an Ω-coalgebra (X, f), for $x \in X_1$, we define the language of lassos*

$$\mathit{Lasso}(x) = \{(s, l) \mid \exists x'.\, \widehat{f_1^1} xs = x' \wedge l \in \mathcal{L}_{\mathtt{D}}(f_1^2 x')\}$$

and the regular language of non-empty words

$$\mathit{Loop}(x) = \mathcal{L}_{\mathtt{D}}(f_1^2 x).$$

The lasso language of a state x is the set of pairs (s, l) such that one may consume s starting from x and staying in the first sort of the coalgebra, then follow l from the corresponding state in the second sort, and reach an accepting state. The overloading of notation is justified, as $\mathit{Lasso}(x)$ (respectively, $\mathit{Loop}(x)$) is the same when x is considered a state of a DMA A, or of the corresponding coalgebra $\mathcal{C}(A)$. Consider states x and y belonging to the first sort of possibly distinguished Ω-coalgebras. Clearly, the image of states x and y along the final morphism coincides if and only if $\mathit{Lasso}(x) = \mathit{Lasso}(y)$. In an Ω-coalgebra, for $x \in X_1$, it is not necessary that $\mathit{Lasso}(x)$ is obtained from the ultimately periodic fragment of some ω-regular language, as shown by the following example.

Example 1. Consider $c \in C$, and a DFA (T, γ, F, t^0) accepting the language $\{c\}$. Let (X, f) be an Ω-coalgebra with $X_1 = \{x\}$, $X_2 = T$, $f_1^1 xc = x$ for all $c \in C$, $f_1^2 x = t^0$, f_2 obtained from γ and F. Then (ϵ, c) belongs to $\mathit{Lasso}(x)$, but (ϵ, cc) does not. However, both lassos represent the ultimately periodic word c^ω. There is no ω-regular language $\mathcal{L}$ such that $\mathit{Lasso}(x) = \mathit{Lasso}(\mathcal{L})$, since c^ω belongs to $\mathcal{L}$ if and only if both (ϵ, c) and (ϵ, cc) belong to $\mathit{Lasso}(\mathcal{L})$.

We shall call *balanced* those Ω-coalgebras whose states correspond to ω-regular languages in the following sense.

Definition 18 (balanced coalgebra). *An Ω-coalgebra (X, f) is* balanced *if, for all $x \in X_1$, there is an ω-regular language $\mathcal{L}_x$ such that $\mathit{Lasso}(x) = \mathit{Lasso}(\mathcal{L}_x)$.*

Clearly, for each DMA A, $\mathcal{C}(A)$ is a (sort-wise) finite balanced coalgebra. However, not all finite balanced coalgebras are equal to $\mathcal{C}(A)$ for some DMA A, as illustrated by the following example.

Example 2. Let $C = \{a, b\}$. Consider the language of infinite words $\mathcal{L} = \{u\alpha | u \in C^* \wedge (\alpha = a^\omega \vee \alpha = b^\omega)\}$. $\mathcal{L}$ is ω-regular, as it is accepted by the two states DMA $A = (\{x_1, x_2\}, \delta, \mathcal{M})$ with $\delta x_1 a = x_1$, $\delta x_1 b = x_2$, $\delta x_2 a = x_1$, $\delta x_2 b = x_2$, $\mathcal{M} = \{\{x_1\}, \{x_2\}\}$. Notice that $UP(\mathcal{L}) = \mathcal{L}$. Consider a $\mathtt{D}$-coalgebra (Y, g) accepting $\{a\}^+ \cup \{b\}^+$ from state t^0. Define the Ω-coalgebra (X, f) where $X_1 = \{x\}$, $X_2 = Y$. For $c \in C$, we let $f_1^1(x)(c) = x$. We let $f_1^2(x) = t^0$, and $f_2 = g$. We have that $Lasso(x) = Lasso(\mathcal{L})$, but there is no DMA B such that $\mathcal{C}(B) = (X, f)$: by construction, the structure defined by the set of states X_1 and transition function f_1^1 has to be the structure of B. But the only choice for an accepting condition is $\mathcal{M} = \{\{x\}\}$. Obviously, such an automaton accepts the whole C^ω.

The relation between DMAs and balanced Ω-coalgebras is slightly more subtle than expected. Example 2 shows that there are DMAs having language equivalent states that can not be quotiented. Thus, DMAs (set aside finiteness constraints) do not admit a universal model, or at least not one which is "minimal" in the traditional sense. Similar considerations apply to parity and Büchi automata. In contrast, each Ω-coalgebra has a quotient with respect to behavioural equivalence (its image in the final coalgebra). Moreover, as DMAs give rise to balanced coalgebras, the subobject of the final Ω-coalgebra, where all DMAs are mapped into, must be balanced. This coalgebra does not correspond to any DMA, as it has infinite states (there are infinitely many distinct ω-regular languages). However, we easily get the following statement, which paves the way to Proposition 6, explaining the relationship between DMAs and balanced coalgebras.

Lemma 3 (Quotients are balanced). *If (X, f) is a balanced Ω-coalgebra, and $h : (X, f) \to (Y, g)$ is an epimorphism of coalgebras, then (Y, g) is balanced.*

Proof. Consider $y \in Y_1$; since h is epic, there is $x \in X_1$ such that $h(x) = y$. Being unique, the final morphism from (X, f) factors trough h. Then, by finality, x and y accept the same lasso language.

By construction, the coalgebra corresponding to a given DMA is sort-wise finite. Proposition 6 below clarifies that finiteness plays a key role in representing DMAs; however, it is not necessary that a coalgebra has a finite number of states, but rather it is sufficient that the equivalence classes induced by bisimilarity are in a finite number, that is, the image in the final coalgebra is sort-wise finite.

Proposition 6 (DMAs and balanced coalgebras). *For each balanced coalgebra whose image (X, f) in the final coalgebra is finite, there is a DMA A such that the image in the final coalgebra of $\mathcal{C}(A)$ coincides with (X, f).*

Proof. Since (X, f) is balanced (Lemma 3), for each state $x \in X_1$, there certainly is a DMA A_x having a state y such that $Lasso(y) = Lasso(x)$. In a DMA, it is obvious that only the states reachable from y concur to the construction of $\mathcal{L}(y)$, thus we assume w.l.o.g. that all the states of A_x are reachable from y. When y is seen as a state in the first sort of $\mathcal{C}(A) = (Y_x, g_x)$, x and y are equated by the final morphism, and the image in the final coalgebra of $\mathcal{C}(A)$ coincides with the subcoalgebra of (X, f) reachable from x. The disjoint union of all the A_x

for $x \in X_1$ is then a witness DMA that proves the thesis. Such DMA exists by finiteness of X_1.

A lasso (s, l) can be interpreted as a finite, pointed coalgebra having finite carrier, for the functor $C \times (-)$. The formal definition is useful, since coalgebraic bisimilarity precisely corresponds to equality of the represented word sl^ω.

Definition 19 (Coalgebra of a lasso). *Consider a lasso (s, l) and define a coalgebra $(X, f : X \to C \times X)$ where X is the finite cardinal $|s| + |l|$, and f is the pairing of f^1 and f^2. We let $f^1(i) = s_i$ if $1 \leq i \leq |s|$, $f^1(i) = l_{i-|s|}$ if $|s| < i \leq |s| + |l|$, $f^2(i) = i + 1$ if $1 \leq i < |s| + |l|$, $f^2(|s| + |l|) = |s| + 1$. Call the state* 1 initial *in (X, f), making it a pointed coalgebra.*

Remark 4. Consider the coalgebras corresponding to two lassos (s, l) and (s', l'). Their initial states are bisimilar if and only if they denote the same infinite word, that is $sl^\omega = s'(l')^\omega$. We will just call (s, l) and (s', l') bisimilar.

Proposition 7 (Invariance). *Consider $L = Lasso(x)$ for x a state in a coalgebra whose image in the final coalgebra is finite. There is an ω-regular language $\mathcal{L}$ such that $L = Lasso(\mathcal{L})$ if and only if L is* invariant under bisimilarity*: whenever (s, l) and (s', l') are bisimilar, $(s, l) \in L$ if and only if $(s', l') \in L$.*

Proof. The languages of states in finite subobjects of the final coalgebra are in one to one correspondence with the regular languages in $C^*\$C^+$, with $\$ \notin C$. A proof can be sketched as follows. For each such coalgebra (X, f), view it as a DFA over $C \cup \{\$\}$ by letting f_1^2 be the derivative with respect to \$. Conversely, define a coalgebra from a DFA by letting the first sort coincide with the structure of the DFA, excluding the \$-labelled transitions; the second sort is defined by the derivatives w.r.t. \$ of the languages of states in the DFA. This established, observe that there certainly is a state z in the final coalgebra whose language is L. Its reachable part is also finite, hence it is a regular language in $C^*\$C^+$. The thesis is obtained by direct application of the following result from [4]: a *regular* language in $C^*\$C^+$ is $\{s\$l | sl^\omega \in \mathcal{L}\}$ for some ω-regular $\mathcal{L}$ if and only if it is bisimulation invariant.

An obvious question at this point is to find a "structural" characterisation of bisimulation invariance for Ω-coalgebras. An answer is provided by Theorem 2.

Theorem 2 (Circular and coherent). *An Ω-coalgebra (X, f) is bisimulation invariant if and only if it is:*

Circular: $\forall x \in X_1. \forall k > 0. \forall u \in C^+. u \in Loop(x) \iff u^k \in Loop(x)$
Coherent: $\forall x \in X_1. Loop(x) = \{cu \mid uc \in Loop(f_1^1(x)(c))\}$

Proof. First, assume bisimulation invariance. To prove circularity, notice that for all u and k, $u^\omega = (u^k)^\omega$; then use bisimulation invariance. To prove coherence, suppose $cu \in Loop(x)$. Then $(\epsilon, cu) \in Lasso(x)$; by bisimulation invariance, since $(cu)^\omega = c(uc)^\omega$, we have $(c, uc) \in Lasso(x)$. Then $uc \in Loop(f_1^1 xc)$. For the

other direction, assume circularity and coherence. Suppose (s, l) is bisimilar to (s', l'). We can define a lasso bisimilar to both of them, and having "longer spoke and loop". Formally, there are a lasso (t, v), integers k, h, k', h', and finite words l_1, l_2, l'_1, l'_2, such that: $sl^\omega = tv^\omega = s'(l')^\omega$; $l = l_1 l_2$; $l' = l'_1 l'_2$; $t = sl^k l_1 = s'(l')^{k'} l'_1$; $v = (l_2 l_1)^h = (l'_2 l'_1)^{h'}$. We now prove $(t, v) \in \mathit{Lasso}(x) \iff (s, l) \in \mathit{Lasso}(x)$. A similar argument can be used to prove $(t, v) \in \mathit{Lasso}(x) \iff (s', l') \in \mathit{Lasso}(x)$, obtaining the thesis.
We have $(t, v) \in \mathit{Lasso}(x) \iff (sl^k l_1, (l_2 l_1)^h) \in \mathit{Lasso}(x) \iff (l_2 l_1)^h \in \mathit{Loop}(\widehat{f_1^1}(sl^k l_1)) \iff$ {by circularity} $l_2 l_1 \in \mathit{Loop}(\widehat{f_1^1}(sl^k l_1)) \iff l_2 l_1 \in \mathit{Loop}(\widehat{f_1^1}(s(l_1 l_2)^k l_1)) \iff$ {by coherence} $l_1 l_2 \in \mathit{Loop}(\widehat{f_1^1}(s(l_1 l_2)^k l_1 l_2)) \iff l \in \mathit{Loop}(\widehat{f_1^1}(sl^{k+1})) \iff$ {by coherence} $l \in \mathit{Loop}(\widehat{f_1^1}(s)) \iff (s, l) \in \mathit{Lasso}(x)$.

Corollary 3. *Consider an Ω-coalgebra C whose image in the final coalgebra is finite. C is balanced if and only if it is circular and coherent.*

Remark 5. The definition of a balanced coalgebra has been formulated in terms of the lasso languages it accepts (taking each of its respective states as initial), and the conditions of circularity and coherence are also phrased in terms of languages (here, languages of finite words). One may wonder whether these notions can be defined more directly, in terms of the structure of the coalgebra and its associated DFAs. We leave this as a question for further research.

5 Decision procedures

Two-sorted coalgebras provide a natural model for lasso languages. In this section, we spell out decision procedures for computing language equivalence (Section 5.1) and boolean operations (Section 5.2), on finite Ω-coalgebras. By Fact 1, when acting on balanced coalgebras, these procedures actually compute with (presentations of) ω-regular languages. By Proposition 6, the results we present yield an alternate proof of closure of DMAs under boolean operations.

5.1 Equivalence checking

Using Ω-coalgebras, language equivalence is reduced to behavioural equivalence. In coalgebras, whenever a final coalgebra exists, this relation can be computed by a generalised form of partition refinement that exploits the final sequence and finiteness. Therefore, the coalgebraic formulation by itself, and the fact that the states in $\mathcal{C}(A)$ for a DMA A are finite, already prove decidability of language equivalence in DMAs.

Deciding behavioural equivalence in an Ω-coalgebra corresponds to computing the kernel of the final morphism. This is done by a fixed point iteration, roughly described as follows. Consider a coalgebra (X, f). Consider a morphism $h : X \to Y$ for some object Y in $\mathtt{Set}^2$; notice that h is not necessarily a homomorphism of coalgebras, but assume that it satisfies $x \simeq_i y \Rightarrow h_i(x) = h_i(y)$, for

$i \in 2$; that is, the kernel of h includes behavioural equivalence. If, in addition, h happens to be a coalgebra homomorphism for some coalgebra over Y, then the reverse inclusion holds, since $h_i(x) = h_i(y) \Rightarrow x \simeq_i y$ by definition of $\simeq$, making the kernel of h coincide with behavioural equivalence. Coalgebraic partition refinement uses this fact and finiteness of X to approximate the kernel of the final morphism "from above", starting from an arrow h whose kernel includes behavioural equivalence. In subsequent steps, h is *refined*, maintaining the property, until a coalgebra morphism is obtained. In the reminder of this section, we give a formal specification of this procedure for finite Ω-coalgebras.

Briefly, the definition is based on the idea of "refining" an approximation of the unique morphism in the final coalgebra until one reaches a fixed point. The starting approximation is the unique morphism from the object of states into the final object of the base category. This morphism equates all the states, thus its kernel includes bisimilarity. Each refinement preserves this property; the termination condition of the algorithm implies that the computed morphism is a coalgebra homomorphism, making it the desired bisimilarity quotient.

Definition 20 (Kernel). *The kernel* $\ker h$ *of a morphism* $h : X \to Y$ *is the pullback object of the diagram* $X \xrightarrow{h} Y \xleftarrow{h} X$*, thus it is a subobject of the product* $X \times X$*. In* $\mathtt{Set}^2$*, limits are computed point-wise, and* $\ker h$ *is the pair of relations* $(\ker h_1, \ker h_2)$*, where each component is just the set theoretical notion:* $\ker h_i = \{(x, y) \in X_i \times X_i | h_i(x) = h_i(y)\}$.

Definition 21 (Refinement). *Given an* Ω*-coalgebra* (X, f) *and an arrow* $h : X \to Y$ *in* $\mathtt{Set}^2$*, call* $(\Omega h) \circ f$ *the* refinement of h with respect to f.

Proposition 8 (Partition refinement). *Given an* Ω*-coalgebra* (X, f)*, define the sequence of morphisms, indexed by* ω*, where* $h_0 : X \to 1$ *is the unique morphism into the final object, and* h_{i+1} *is the refinement of* h_i *with respect to* f*. Then define the sequence* $K_i = \ker h_i$*. If both* X_1 *and* X_2 *are finite, then* K_i *converges at some finite step* j*. In that case, let* $\bar{X}$ *denote the target of* h_j*. There is a coalgebra structure* $\bar{f} : \bar{X} \to \Omega\bar{X}$ *such that* h_j *is a coalgebra homomorphism from* (X, f) *to* $(\bar{X}, \bar{f})$*, and* K_j *is behavioural equivalence in* (X, f)*. More precisely,* $(\bar{X}, \bar{f})$ *is (up-to isomorphism) the image of* (X, f) *in the final coalgebra.*

The algorithm starts from the unique morphism from X into the final object of $\mathtt{Set}^2$ (which is not required to be a homomorphism of coalgebras). Recall that in $\mathtt{Set}^2$ colimits are computed point-wise. Thus the final morphism from X is the pair of final morphisms from X_1 and X_2 in $\mathtt{Set}$. Its kernel is the two sorted relation $(X_1 \times X_1, X_2 \times X_2)$ making all the states equivalent. The kernel of this morphism necessarily includes behavioural equivalence; the two are equal only if all states are (sort-wise) equivalent; thus it is an approximation of the unique morphism into the final coalgebra. The approximation is refined in subsequent steps, obtaining at most the identity function (up-to isomorphism). This limit case is reached (in a finite number of steps), only if no states are equivalent. Correctness and convergence of the algorithm comes from convergence of the terminal sequence (see [6]). Convergence in a finite number of steps is just a consequence of finiteness of the state space.

5.2 Boolean operations

Boolean operations on Ω-coalgebras are particularly simple and reminiscent of the corresponding operations on classical DFAs. First recall that the set 2 has the structure of a boolean algebra, with the obvious definition of the boolean operations $\neg$, $\wedge$, $\vee$ denoting complementation, conjunction and disjunction, respectively. In order to be coherent with Definition 10, we assume 1 to denote the truth value *true* and 2 to denote *false*. In the following, let (X, f) and (Y, g) be Ω-coalgebras.

Definition 22 (Complement). *Define the* complement *coalgebra*

$$(X, \bar{f})$$

where

$$\bar{f}_1 = f_1 \qquad \bar{f}_2 xc = (\pi_1(f_2 xc), \neg\pi_2(f_2 xc)).$$

Definition 23 (Union). *Define the* union *coalgebra*

$$(X \times Y, f \cup g)$$

where

$$(f \cup g)^1_1(x, y)(c) = (f^1_1 xc, g^1_1 yc) \qquad (f \cup g)^2_1(x, y) = (f^2_1 x, f^2_1 y)$$

$$(f \cup g)_2(x, y)c = ((\pi_1(f_2 xc), \pi_1(g_2 yc)), \pi_2(f_2 xc) \vee \pi_2(g_2 yc)).$$

Definition 24 (Intersection). *Define the* intersection *coalgebra*

$$(X \times Y, f \cap g)$$

where

$$(f \cap g)^1_1(x, y)(c) = (f^1_1 xc, g^1_1 yc) \qquad (f \cap g)^2_1(x, y) = (f^2_1 x, f^2_1 y)$$

$$(f \cap g)_2(x, y)c = ((\pi_1(f_2 xc), \pi_1(g_2 yc)), \pi_2(f_2 xc) \wedge \pi_2(g_2 yc)).$$

The operations we define actually implement boolean operations on lasso languages, as shown in Proposition 9 below. Furthermore, boolean operations on balanced coalgebras yield balanced coalgebras. This fact is proved in Proposition 10, and is a direct application of Theorem 2. Thus, by Proposition 6, we get a simple proof that ω-regular languages are closed under boolean operations. We state this fact in Corollary 4, concluding the section.

Proposition 9 (Boolean algebra). *For $x \in X_1$ and $y \in Y_1$, let $L_x = Lasso(x)$ and $L_y = Lasso(y)$ in the coalgebras (X, f) and (Y, g), respectively; let $L_{\bar{x}} = Lasso(x)$ in $(X, \bar{f})$; let $L_{x \cup y} = Lasso(x, y)$ in $(X \times Y, f \cup g)$; let $L_{x \cap y} = Lasso(x, y)$ in $(X \times Y, f \cap g)$. We have $L_{\bar{x}} = C^* \times C^+ \setminus L_x$, $L_{x \cup y} = L_x \cup L_y$ and $L_{x \cap y} = L_x \cap L_y$.*

Proof. Recall that for $x \in X_1$ and $y \in Y_1$, $f_1^2(x)$ and $g_1^2(y)$ are initial states of DFAs. For complementation, observe that $(s,l) \in \mathit{Lasso}(x) \iff l \in \mathcal{L}_{\mathtt{D}}(f_1^2(\widehat{f_1^1}(x))$. For all $x \in X_1$, the construction complements the DFA of non-empty words rooted at $f_1^2(x)$. Binary operations are also simple. Notice that the structure of $(X \times Y, f \cup g)$ is that of their parallel composition, thus it also includes the parallel composition of the DFAs rooted at $f_1^2(x)$ and $g_1^2(y)$ for all $(x,y) \in (X_2 \times Y_2)$. The obtained DFA is rooted at $(f \cup g)_1^2(x,y)$, and it accepts $\mathit{Loop}(x) \cup \mathit{Loop}(y)$ by construction. A similar argument holds for $(X \times Y, f \cap g)$.

Proposition 10 (Boolean operations are balanced). *If (X,f) is balanced and image-finite in the final coalgebra, then so is $(X, \bar{f})$. If, additionally, (Y,g) is in the same conditions, then also $(X \times Y, f \cup g)$ and $(X \times Y, f \cap g)$ are balanced and image-finite along the final morphism.*

Proof. By construction, boolean operations preserve finiteness, thus also image-finiteness along the final morphism. We prove that circularity and coherence are preserved in turn. Suppose (X,f) is circular, and consider $x \in X_1$, $k > 0$, $u \in C^+$. Call $\mathit{Loop}(\bar{x})$ the language $\mathit{Loop}(x)$ in the complement coalgebra. We have $u \in \mathit{Loop}(\bar{x}) \iff u \notin \mathit{Loop}(x) \iff u^k \notin \mathit{Loop}(x) \iff u^k \in \mathit{Loop}(\bar{x})$, thus the complement coalgebra is circular. For the union, we have $u \in \mathit{Loop}(x,y) \iff u \in \mathit{Loop}(x) \cup \mathit{Loop}(y) \iff u \in \mathit{Loop}(x) \vee u \in \mathit{Loop}(y) \iff u^k \in \mathit{Loop}(x) \vee u^k \in \mathit{Loop}(y) \iff u^k \in \mathit{Loop}(x,y)$. Similarly for the intersection (or just by de Morgan laws). Coherence is dealt with in the same way.

Corollary 4. *Deterministic Muller automata, thus also ω-regular languages, are closed under boolean operations.*

Proof. By Proposition 10, Proposition 6, and the fact that the coalgebra of a DMA is finite and balanced.

As a final remark, we notice that the combination of boolean operations and equivalence checking allows one to derive many set-theoretical operations on lasso languages; as a fundamental example, it yields a proof of decidability of language inclusion.

6 Future work

In [4], the authors hoped "*that a number of constructions which are presently outwardly performed on Büchi automata can be performed on simple DFAs*". Their goal is somewhat obfuscated by the fact that DFAs do not precisely characterise lasso languages; thus, for example, bisimulation invariant lasso languages are not closed under complementation of DFAs. This hints at the possibility of more precise structures, that characterise ω-regular languages better. Ω-coalgebras are perfectly fitted: in Section 5 we see that typical operations on automata become fairly standard using them; still, they maintain the definitional simplicity of DFAs, and their accepting condition is local to states.

Lasso languages provide a *finite* syntax for ω-regular languages. We expect that a deeper study of balanced coalgebras may help understanding the issues of finite memory in the realm of (languages of) streams, which are infinite structures by their own nature. In this respect, the study of pumping lemmas for lasso and ω-regular languages seems an interesting research direction to take.

The importance of circularity and coherence should not be underestimated: it allows the proof of Proposition 10 to be carried out directly, without assuming closure under boolean operations of ω-regular languages. In combination with Proposition 6, this yields a "lasso-language-theoretical" alternate proof of closure of ω-regular languages under boolean operations. The definition of the two properties also suggests that circular and coherent coalgebras may be treated as a covariety. The impact of such a statement has to be framed in the more general question of what can the theory of coalgebras add to the theory of stream automata.

Future work in this sense may be directed in at least three ways. First, the theory of Ω-coalgebras has just been introduced. Many results in coalgebras have not been explored yet in conjunction with them (covarieties, coalgebraic modal logics, and trace semantics come to mind). Second, as it is typical in coalgebras, one could seek for generalisation of lasso languages and ω-regular languages, as it has been done for the case of DFAs. An interesting case study is that of automata over infinite alphabets. Similarly to [5, 2], one could use coalgebras in the category of nominal sets, which are then finitely represented using a categorical equivalence. Third, lassos can in turn be seen as coalgebras, of which Ω-coalgebras are classifying structures. Coalgebra automata [10] enhance streams by making them elements of the final coalgebra for a certain functor which can be varied, providing a general notion of modal fixpoint logics for infinite structures. The relationship between stream languages and their ultimately periodic fragments should be investigated in this more general setting.

Acknowledgements

The authors wish to thank Jean-Eric Pin for pointing them to the work [4].

References

1. Bartels, F., Sokolova, A., de Vink, E.P.: A hierarchy of probabilistic system types. Theoretical Computer Science 327(1-2), 3–22 (2004)
2. Bojanczyk, M., Klin, B., Lasota, S.: Automata with group actions. In: LICS. pp. 355–364. IEEE Computer Society (2011)
3. Bonchi, F., Bonsangue, M., Rutten, J., Silva, A.: Deriving syntax and axioms for quantitative regular behaviours. In: CONCUR. Lecture Notes in Computer Science, vol. 5710, pp. 146–162. Springer-Verlag, Berlin, Heidelberg (2009)
4. Calbrix, H., Nivat, M., Podelski, A.: Ultimately periodic words of rational ω-languages. In: MFPS. Lecture Notes in Computer Science, vol. 802, pp. 554–566. Springer (1993)

5. Ciancia, V., Tuosto, E.: A novel class of automata for languages on infinite alphabets. Tech. rep., University of Leicester (2009)
6. James, Worrell: Terminal sequences for accessible endofunctors. In: CMCS. Electronic Notes in Theoretical Computer Science, vol. 19, pp. 24 – 38. Elsevier (1999)
7. Pin, J.E., Perrin, D.: Infinite Words: Automata, Semigroups, Logic and Games. Elsevier (2004)
8. Rutten, J.J.M.M.: Automata and coinduction (an exercise in coalgebra). In: CONCUR. Lecture Notes in Computer Science, vol. 1466, pp. 194–218. Springer (1998)
9. Rutten, J.J.M.M.: Universal coalgebra: a theory of systems. Theoretical Computer Science 249(1), 3–80 (2000)
10. Venema, Y.: Automata and fixed point logic: A coalgebraic perspective. Information and Computation 204(4), 637–678 (2006)