Automata on flows

MSc Thesis (*Afstudeerscriptie*)

written by

Petter Remen
(born 5th August 1980 in Tønsberg, Norway)

under the supervision of **Yde Venema**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
29th August, 2007	Yde Venema
	Peter van Emde Boas
	Jan Rutten
	Johan van Benthem

Institute for Logic, Language and Computation

Abstract

We introduce a coalgebraic generalization of an infinite word, namely a *sourced flow*, followed by a definition of what it means for an automaton to accept a sourced flow, thereby generalizing the notion of ω-regularity. We show that this definition yields a finitary description of acceptance for finite sourced flows. We end by presenting three characterization results on regular classes of finite sourced flows.

1 Introduction

There is a deep connection linking the fields of automata and logic. On the conceptual level, both use a relation (acceptance/satisfaction) to let finite objects (automata/formulas) describe properties of other, possibly infinite structures (e.g. infinite words/models). Intuitively, one may see automata as a generalization of formulas, in the sense that the syntactic structure of the latter is a tree, whereas automata admit *loops*, i.e., their underlying structure is a finite *graph*.

And although the terminology differs, the questions asked are often similar. Can we characterize those classes $\mathcal{C}$ of structures which are such that there exists a formula φ (an automaton $\mathbb{A}$) such that $M \in \mathcal{C}$ if and only if M satisfies φ (M is accepted by $\mathbb{A}$)? How are the expressive powers of different logics (classes of automata) related, when compared over a class of structures?

In some cases, the connection is very strong indeed. In the 60's, Büchi obtained a decision procedure for the monadic second order logic of one successor, S1S [2], by showing that S1S and automata are equally strong in expressive power over the class of infinite words. In fact, he showed that there is a constructive algorithm transforming a given S1S sentence φ into an automaton $\mathbb{A}_\varphi$ such that φ is true for an infinite word u if and only if u is accepted by $\mathbb{A}_\varphi$. Since the *emptiness problem* for automata – i.e., the question of whether a given automaton $\mathbb{A}$ accepts *any* infinite words at all – is computable, this gave an effective procedure for deciding satisfiability of S1S formulas. The techniques used by Büchi were later used by Rabin in [16], in which he showed that the stronger theory of monadic second order logic of two successors, S2S, is decidable (over the class of binary trees). The result, also called Rabin's theorem, has been successfully applied to obtain decidability results for other logics and problems by reducing them to questions of S2S.

1.1 Modal logic and automata

Both Büchi and Rabin treated infinite words and binary trees as colored transition structures when seeing them as second-order models. It therefore makes sense to compare the expressive power of automata to that of a natural logical language for transition structures, namely *modal logic*. It is not difficult to see that classical modal logic is quite weak in comparison. Since every modal formula only can "see" finitely many steps ahead, such properties as "there are only finitely many states which satisfy the propositional variable p" or even "each state which satisfies p is eventually followed by a state which satisfies q" are not expressible, even though they are easily characterized by automata. However, it turns out that adding adding *fixpoint operators* to the classic modal language adds precisely the strength required in order for the logic to gain the same expressive power as automata. In fact, there is a back-and-forth translation between what is called the *modal μ-calculus* [9] and *alternating tree automata* [24]. This has turned out to be quite a success story for computer science. Viewing Kripke models as process graphs for computer programs with non-deterministic behavior, one can use the modal μ-calculus as a highly expressive specification language. Then, using the translation into automata, one can *check* whether a given graph satisfies the specification, yielding a field of research called *model checking*.

For a modal logician, the modal μ-calculus serves as a very beautiful exten-

sion to the classic modal language, being remarkably strong but still preserving two important properties, namely the *finite model property* and *bisimulation invariance*. Many questions from modal logic can be lifted to this setting, such as whether there exists a characterization theorem for the modal μ-calculus in the style of van Benthem's result [21] for the classical modal logic. This question was answered in [8], when Janin & Walukiewicz showed that it is equal in expressive power to the bisimulation invariant fragment of monadic second order logic. There are still open questions in this area, such as whether the result of Janin & Walukiewicz still holds if we restrict ourself to the class of *finite* models.

1.2 The coalgebraic perspective

It turns out that the strong connection between automata and logic holds for many different classes of transition structures (in this introduction, we have already briefly mentioned infinite words, binary trees and Kripke models). Moreover, the techniques used are often similar, indicating the possibility of a more general setting in which to develop these theories.

The fact that transition systems can be seen as *coalgebras* was observed in [1]. It also turns out that the familiar notions of bounded morphisms and bisimulation are natural in the coalgebraic setting. Moreover, modal logic, which classically applies to Kripke models (which are coalgebras of a special signature), can be generalized to coalgebras [13]. Combining this with the link between the modal μ-calculus and automata, a natural question arises: is it possible to develop a theory of *coalgebraic automata* and add fixpoints to coalgebraic logics in such a way that this link is preserved?

This question was positively answered by Venema in [22], in which a general theory of coalgebraic fixpoint logic and coalgebraic automata was developed, and in [10] together with Kupke, where standard results from automata theory was shown to hold in this, more general, setting.

1.3 The aim of this thesis

In this thesis we investigate automata operating on one of the simplest classes of coalgebras, namely *sourced flows*. Sourced flows are labeled transition structures with a well-defined initial state and where each state has a unique successor. As such, they constitute a natural generalization of an infinite word. In fact, it turns out that we can view the "ordinary" infinite words as precisely the sourced flows which have infinitely many states. Therefore, the generalization can be seen as a way of adding *finite models* to the existing theory. In fact, our primary focus will be to find characterization theorems for those classes $\mathcal{L}$ of finite sourced flows which are *regular*, in the sense that there is an automaton $\mathbb{A}$ such that $\mathcal{L}$ consists of precisely the finite sourced flows which $\mathbb{A}$ accepts. Implicit in our investigation is the connection between automata and the corresponding modal calculus with fixpoint operators, so, actually we will be investigating the *finite model theory* of this calculus.

Seeing how we want to fit into the coalgebraic setting, our definitions, statements and proofs will try to use the toolkit of universal coalgebra, such as the notions of *homomorphisms*, *bisimilarity* and *generated subcoalgebra*. This has the added advantage of making it easier to see how the ideas in this thesis might

possibly be generalized to other classes of coalgebras with more complicated signatures than ours.

2 Preliminaries

We assume that the reader is proficient with the standard toolset of regular languages and automata operating on finite words.

We begin by presenting a characterization theorem for regular languages in terms of a pumping property. Both the result in itself and its proof will be used to obtain a characterization theorem for regular languages of lassos (defined in section 3).

The subsequent part of this section is devoted to introducing automata on infinite words; however, it can hardly be seen as a tutorial. Hence, for the reader who wants to have a better grip on the subject, we recommend the works of Thomas in [20] & [19], Perrin & Pin in [15], and the comprehensive work in [5].

Before we begin, know that from now on and throughout the entire thesis, Σ denotes some fixed finite alphabet of *symbols*. We tend to use lowercase a's, b's, and c's as variables ranging over Σ. Elements of Σ^* are called *finite words* and elements of Σ^ω are called *infinite words*.

2.1 Regular languages and the block-removal property

It is well-known that regular languages have a "pumping" property. That is, if $\mathcal{L} \subseteq \Sigma^*$ is a regular language and x is a sufficiently large string in $\mathcal{L}$, then there is some substring of x which can be removed or repeated arbitrarily many times, the result always remaining in $\mathcal{L}$. Eurenfeucht, Parikh and Rozenberg showed in [3] that there is a pumping condition which is equivalent to regularity. We present here the main argument, referring the reader to the paper for the full proofs.

Let $w \in \Sigma^*$ be a finite word, and let $y_1, y_2, \ldots, y_k$ be a sequence of consecutive (possibly empty) subwords of w, i.e., $w = xy_1y_2 \ldots y_kz$ for some $x, z \in \Sigma^*$. We consider those strings w' which are obtained from w by removing "blocks" of consecutive y's. Formally, we call a word $w' \in \Sigma^*$ a *cut* of $xy_1y_2 \ldots y_kz$, if $w' = xy_1y_2 \ldots y_iy_{j+1} \ldots y_kz$ for some $0 \leq i < j \leq k$. Note that w' is a cut relative to the "factorization" $w = xy_1y_2 \ldots y_kz$. Also note that there are no "empty" cuts, since we forced $i < j$.

$$w = xy_1y_2 \ldots y_{i-1}y_i \overbrace{y_{i+1} \cdots y_{j-1}y_j}^{\text{the removed block}} y_{j+1} \ldots y_kz$$
$$w' = xy_1y_2 \ldots y_{i-1}y_iy_{j+1} \ldots y_kz$$

Figure 1: An example of a cut

Definition 2.1. Let $\mathcal{L} \subseteq \Sigma^*$ be set of finite words and let $k \in \omega$ be a natural number. We say that $\mathcal{L}$ has the k-blockremoval property if for every $w \in \Sigma^*$ and sequence $y_1, \ldots, y_k$ of length k such that $w = xy_1y_2 \ldots y_kz$ for some $x, z \in \Sigma^*$, there is a cut w' of $xy_1y_2 \ldots y_kz$ such that $w \in \mathcal{L}$ if and only if $w' \in \mathcal{L}$.

Theorem 2.2 (Eurenfeucht, Parikh & Rozenberg)**.** A language $\mathcal{L}$ is regular if and only if it has the k-blockremoval property.

The proof of this theorem is by means of three lemmas, the first of which is of independent interest to us and is used in Theorem 5.11. The third shows how to create a deterministic automaton from a finite congruence, a construction from which we take inspiration in Theorem 5.26.

Lemma 2.3. For every $k \in \omega$, there are only finitely many languages $\mathcal{L} \subseteq \Sigma^*$ with the k-blockremoval property.

Proof. The proof uses Ramsey's theorem to show that for every $k \in \omega$ there is a number $r(k)$ such that every language $\mathcal{L}$ with the k-blockpumping property is uniquely determined by its set of strings of length $\leq r(k)$. Hence, there are at most $|\Sigma|^{r(k)}$ many languages with the k-blockpumping property. □

Lemma 2.4. If $\mathcal{L} \subseteq \Sigma^*$ has the k-blockpumping property, then so do all of its derivatives, i.e., $\mathcal{L}_a = \{ x \mid ax \in \mathcal{L} \}$ has the k-blockpumping property for each $a \in \Sigma$.

Proof. Let $xy_1y_2 \dots y_kz$ be a finite word. Then there are $i < j \leq k$ such that $axy_1y_2 \dots y_kz \in \mathcal{L}$ if and only if $axy_1 \dots y_iy_{j+1} \dots y_kz \in \mathcal{L}$, and hence $xy_1y_2 \dots y_kz \in \mathcal{L}_a$ if and only if $xy_1 \dots y_iy_{j+1} \dots y_kz \in \mathcal{L}_a$. □

Lemma 2.5. Let $\mathbb{L}$ be a finite set of languages such that $\mathcal{L} \in \mathbb{L}$ implies that $\mathcal{L}_a \in \mathbb{L}$ for all $a \in \Sigma$. Then every language in $\mathbb{L}$ is regular.

Proof. We let $\mathbb{A}$ be the deterministic automaton (without initial states) where

- the set of states is $\mathbb{L}$,
- the transition function is such that for every $\mathcal{L} \in \mathbb{L}$ and $a \in \Sigma$, the unique a-successor of $\mathcal{L}$ is $\mathcal{L}_a$, and
- a state $\mathcal{L}$ is accepting if and only if $\mathcal{L}$ contains the empty word, i.e., $\epsilon \in \mathcal{L}$.

It is now straightforward to check that the language accepted by setting the initial point of $\mathbb{A}$ to a state $\mathcal{L}$ is precisely $\mathcal{L}$ itself. □

Proof of Theorem 2.2. If $\mathcal{L}$ is a regular language, then there is a deterministic automaton $\mathbb{A}$ which accepts $\mathcal{L}$. Letting k be the number of states of $\mathbb{A}$, it is easy to check that $\mathcal{L}$ has the k-blockpumping property.

For the other direction, the three lemmas show that every language which has the k-blockpumping property is regular. □

2.2 Automata on infinite words

As in the finitary case, the underlying structure for an automaton operating on infinite words is a finite labeled transition system. The structure necessary to define acceptance in the infinitary case is a bit more involved. Instead of a single set of accepting states, we need a *set* of sets of states, the reason for which will be explained shortly.

Definition 2.6. An automaton $\mathbb{A}$ consists of

1. a finite set of *states* $Q_{\mathbb{A}}$,
2. a *transition function* $\Delta_{\mathbb{A}} \colon Q \to \wp(\Sigma \times Q)$,

3. a set $I_{\mathbb{A}} \subseteq Q$ of *initial states*, and

4. an *acceptance condition* $Acc_{\mathbb{A}} \subseteq \wp(Q)$.

We omit the subscript $\mathbb{A}$ when the automaton is implicit from context.

One often sees the underlying labeled transition systems of an automaton formalized by means of a transition *relation* $R \subseteq Q \times \Sigma \times Q$. The two different definitions are equivalent through the isomorphism mapping such a relation R to the transition function $\Delta_R = \lambda q.\{\, (a, q') \mid (q, a, q') \in R \,\}$.

Let $\mathbb{A} = (Q, \Delta, I, Acc)$ be an automaton. We introduce some convenient shorthands for reasoning about transitions.

- We write $q \xrightarrow{a} q'$ for $(a, q') \in \Delta(q)$.

- We inductively define a terniary relation $q \overset{x}{\twoheadrightarrow} s$, where q and s are elements of Q and $x \in \Sigma^*$ is a finite word, the induction being performed on the length of x.

  In the base case when $x = \epsilon$, i.e., x is the empty word, we have that $q \overset{x}{\twoheadrightarrow} q'$ if and only if $q = s$. In the inductive case when $x = a_0 \dots a_{n-1}$, we let $x' = a_1 \dots a_{n-1}$. We then have that $q \overset{x}{\twoheadrightarrow} s$ if and only if there is an $r \in Q$ such that $q \xrightarrow{a_0} r$ and $r \overset{x'}{\twoheadrightarrow} s$.

- Given two states $q, q' \in Q$, we write $[q \twoheadrightarrow q']$ for the set of all finite words $x \in \Sigma^*$ such that $q \overset{x}{\twoheadrightarrow} q'$.

As in the finitary case, the acceptance of an infinite word is witnessed by a (in our case infinite) *trace* through the automaton.

Definition 2.7. Let $\mathbb{A}$ be an automaton and let $u = a_0 a_1 \dots \in \Sigma^\omega$ be an infinite word. A u-labeled *trace* through $\mathbb{A}$ is a sequence of pairs $(q_0, a_0), (q_1, a_1) \dots$, such that $q_i \xrightarrow{a_i} q_{i+1}$ for all $i \in \omega$. We often write

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \cdots$$

for traces, calling the sequence $q_0 q_1 \dots$ the *states* of the trace, and q_0 the *starting state*.

We now use the acceptance condition to specify which traces are successful. In the finitary case, a finite path through the automaton is successful if it ends in an accepting state. As traces are one-way infinite, we do not have the luxury of a final state; instead, we define the success of trace based on its infinitary behavior.

Definition 2.8. Given an automaton $\mathbb{A}$ and an trace $(q_0, a_0), (q_1, a_1), \dots$ through $\mathbb{A}$, we let

$$\inf(\tau) = \{\, q \in Q \mid q = q_i \text{ for infinitely many } i \in \omega \,\}.$$

Definition 2.9. A trace τ through an automaton $\mathbb{A}$ with acceptance condition Acc is said to be *successful* if $\inf(\tau) \in Acc$.

Consider now a point q in an automaton. If we take the collection of successful traces which start in q and take their labels, we obtain a subset of Σ^ω, i.e., a *language*.

Definition 2.10. Let $\mathbb{A}$ be an automaton. The (infinitary) language *defined* or *accepted* by a point $q \in \mathbb{A}$, written $\mathcal{L}_\omega(\mathbb{A}, q)$, or simply $\mathcal{L}_\omega(q)$ if the automaton is implicit, is defined as the set of all $u \in \Sigma^\omega$ such that there is a successful u-labeled trace τ with starting state q.

Letting I be the set of initial states of $\mathbb{A}$, the language *defined* or *accepted* by $\mathbb{A}$, written $\mathcal{L}_\omega(\mathbb{A})$, is defined as

$$\mathcal{L}_\omega(\mathbb{A}) := \bigcup_{q \in I} \mathcal{L}_\omega(q),$$

By calling successful traces which start in an initial state *accepting*, we get that an infinite word u is *accepted* by an automaton $\mathbb{A}$ (i.e., $u \in \mathcal{L}_\omega(\mathbb{A})$) if and only if u is the label of an accepting trace through $\mathbb{A}$.

Definition 2.11. A language $\mathcal{L} \subseteq \Sigma^\omega$ is called *ω-regular* if there is some automaton $\mathbb{A}$ which accepts it, i.e., $\mathcal{L} = \mathcal{L}_\omega(\mathbb{A})$.

It is worth noting that there are a plethora of different acceptance conditions in the literature, all of which are defined in terms of the infinitary behavior of traces. While defining our automata, we have implicitly chosen the most general of these, namely *Muller acceptance*, but will more often use two other, at first sight more restrictive, conditions. The first one, called *Büchi acceptance*, is the one most reminiscient of the finitary case and the one which will be used most often in this thesis.

Definition 2.12 (Büchi automata)**.** Let $\mathbb{A}$ be an automaton with states Q and acceptance condition Acc. We say that Acc is a *Büchi acceptance condition*, and $\mathbb{A}$ a *Büchi automaton*, if there is a set $F \subseteq Q$ such that

$$Acc = \{\, X \subseteq Q \mid X \cap F \neq \emptyset \,\}.$$

The elements of F are called *accepting states* .

If $\mathbb{A}$ is a Büchi automaton with accepting states F, we have that a trace $q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \dots$ is successful if and only if $q_i \in F$ for infinitely many $i \in \omega$; and since automata always have finitely many states, this is equivalent to requiring that there is *some* state $q_F \in F$ such that $q_i = q_F$ for infinitely many $i \in \omega$. We will often write $F_{\mathbb{A}}$ for the set of accepting states of a Büchi automaton $\mathbb{A}$. Figure 2 shows an example of a Büchi automaton. The initial states are pointed to with large ingoing arrows, and the accepting states are marked with double circles. The automaton in Figure 2, has q_0 and q_1 as initial states, and q_1 and q_2 as accepting states.

The language accepted by a Büchi automaton can easily be described in terms of finitary regular languages.

Definition 2.13. Let $\mathcal{L} \subseteq \Sigma^*$ be a set of finite words. We define

$$\mathcal{L}^\omega := \{\, x_0 x_1 x_2 \cdots \mid x_i \in \mathcal{L} \text{ for all } i \in \omega \,\}.$$

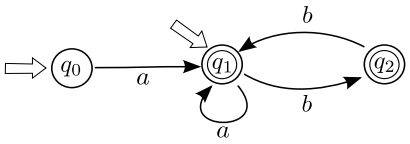


Figure 2: A Büchi automaton

Proposition 2.14 (Büchi)**.** Let $\mathbb{A}$ be a Büchi automaton with initial states I and accepting states F. The language $\mathcal{L}(\mathbb{A})$ accepted by $\mathbb{A}$ is precisely

$$\bigcup_{q_I \in I} \bigcup_{q_F \in F} [q_I \twoheadrightarrow q_F] \circ [q_F \twoheadrightarrow q_F]^\omega$$

Proof. An infinite word u is accepted by $\mathbb{A}$ if and only if there is an u-labeled successful trace which starts in an initial state $q_I \in I$. However, a trace $(q_I q_1 q_2 \ldots, u)$ is successful if and only if there is some state $q_F \in F$ such that $q_i = q_F$ for infinitely many $i \in \omega$. From this the result follows easily. □

The second condition is a bit more involved. If Q is the set of states of some automaton and n is some natural number, we call a function $p\colon Q \to \{1, 2, \ldots, n\}$ a *parity map*.

Definition 2.15 (Parity automata)**.** Let $\mathbb{A}$ be an automaton with states Q and acceptance condition Acc. We say that Acc is a *parity acceptance condition* and $\mathbb{A}$ is a *parity automaton* if there is a parity map $p\colon Q \to \{1, 2, \ldots, n\}$ such that

$$Acc = \{\, X \subseteq Q \mid \max(p[X]) \text{ is even} \,\}.$$

In other words, a trace τ through an automaton $\mathbb{A}$ with parity map p is successful if and only if the element of $\inf(\tau)$ with highest parity is even.

Both of these conditions are useful because the success of a trace is witnessed by a *single* state in its infinite behavior; a fact which makes reasoning about these sort of automata much easier. Surpringly, by the next theorem we can always assume our automata to be of this kind, and in the case of parity automata, we can also assume the underlying transition system to be *deterministic*.

Definition 2.16. An automaton $\mathbb{A}$ is said to be *deterministic* if the set I of initial states is a singleton set and for every $q \in Q$, $a \in \Sigma$, there is *exactly one* $q' \in Q$ such that $q \xrightarrow{a} q'$ in $\mathbb{A}$.

Theorem 2.17. Let $\mathcal{L} \subseteq \Sigma^\omega$ be a set of infinite words. Then the following are equivalent:

(a) $\mathcal{L}$ is ω-regular, i.e., $\mathcal{L}$ is accepted by an automaton.

(b) $\mathcal{L}$ is accepted by a (possibly non-deterministic) Büchi automaton.

(c) $\mathcal{L}$ is accepted by a deterministic parity automaton.

Proof. Both the proof of (a) ⇒ (b) and (b) ⇒ (c) are non-trivial (the second one being by far the hardest). For a full proof, see [4] and [17]. □

Using the (a) ⇒ (b) part of the theorem, we get the following characterization theorem for regular languages.

Theorem 2.18 (Büchi)**.** A language $\mathcal{L}$ is ω-regular if and only if there is a finite set $\{ X_0, Y_0, X_1, Y_1, \ldots, X_{n-1}, Y_{n-1} \}$ of regular languages such that

$$\mathcal{L} = \bigcup_{i<n} X_i \circ Y_i^\omega$$

Proof. If $\mathcal{L}$ is regular, then by Theorem 2.17 there is a Büchi automaton which accepts $\mathcal{L}$. The result then follows from Proposition 2.14.

We omit the proof of the other direction, since a similar construction is used when proving the analogous Theorem 5.3. □

Note that if $\mathbb{A}$ is a deterministic automaton whose single initial state is q_I, then, for every infinite word u, there is *exactly one* u-labeled trace which starts in q_I. Since Büchi automata are the simplest to work with, we would have hoped that deterministic Büchi automata would suffice, but unfortunately there are ω-regular languages which are not accepted by deterministic Büchi automata. However, there is another class of automata which almost do the trick, and which also correspond to the "syntactic" automata for finitary regular languages, namely prophetic automata.

2.3 Prophetic automata

Definition 2.19. An automaton $\mathbb{A}$ is said to be *prophetic* if for every infinite word u, there is *exactly* one successful u-labeled trace through $\mathbb{A}$.

Now, if we assume that our automata are *trim*[1], meaning that for every state q, the language $\mathcal{L}_\omega(q)$ is non-empty, we get the following proposition.

Proposition 2.20. Let $\mathbb{A}$ be an automaton. Then $\mathbb{A}$ is prophetic if and only if

$$\bigcup_{q \in \mathbb{A}} \mathcal{L}_\omega(q) = \Sigma^\omega \tag{1}$$

and

$$\mathcal{L}_\omega(q) \cap \mathcal{L}_\omega(q') = \emptyset \text{ for all distinct } q, q' \in \mathbb{A}. \tag{2}$$

In other words, $\mathbb{A}$ is prophetic if and only if the sets $\mathcal{L}_\omega(q)$ form a *partition* of Σ^ω.

Notice the almost deterministic quality of prophetic automata. As long as we are only interested in *successful* traces, there is but one choice. It is therefore quite surprising that the class of prophetic Büchi automata is equally strong in expressive power as the class of all automata.

Theorem 2.21 (Carton & Michel)**.** For every ω-regular language $\mathcal{L}$, there is a prophetic Büchi automaton $\mathbb{A}$ which accepts $\mathcal{L}$.

[1]It should be fairly clear that this is safe to assume since every state which defines the empty language can be safely removed.

Proof. This was proven in [12] where the term "complete and unambiguous" is used instead of "prophetic", the latter borrowed from [15]. □

As we shall see in Theorem 5.26, prophetic automata can also be seen as syntactic automata for ω-regular languages, similar to the construction used in Lemma 2.5 in the section on regular languages and the block-removal property.

3 (Sourced) Flows

This chapter defines the primary objects of our investigation, namely *sourced flows*. These can be seen in two ways; as a natural coalgebraic generalization of an infinite word, or as the natural class of structures for which the modal μ-calculus of one modal "next"-operator applies.

Definition 3.1. A *flow* $\mathcal{S}$ consists of a non-empty set S, called the *carrier* of $\mathcal{S}$, together with a function $\sigma\colon S \to \Sigma \times S$ called the *transition function* or *successor function* of $\mathcal{S}$. Elements of S are usually designated with letters r, s, t, and are called the *states* of (or *points in*) $\mathcal{S}$. If $\sigma(s) = (a, s')$ we call a the *color* of s and s' the *successor* of s .

Alternatively, we can think of flows as the special case of a labeled transition system in which each point has a unique successor, i.e., for every state s there is a unique pair $(a, s') \in \Sigma \times S$ such that $s \xrightarrow{a} s'$. This point of view will be very helpful to us later when we consider automata operating on flows, since it enables us to see the structural similarities between the two classes of objects.

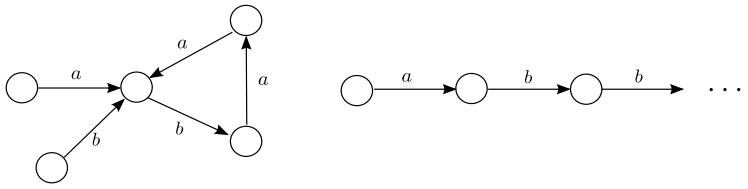


Figure 3: A typical flow.

Figure 3 shows a typical flow over the alphabet $\{a, b\}$. Note that it is not *connected* being the disjoint union of two flows. Another illustrative example is the two way infinite flow

$$\ldots \xrightarrow{a} -2 \xrightarrow{b} -1 \xrightarrow{a} 0 \xrightarrow{b} 1 \xrightarrow{a} 2 \xrightarrow{b} \ldots$$

showing that it is quite possible for every point to have a predecessor.

Since flows can be seen as a special case of labeled transition systems, we adopt the same notation.

Definition 3.2. Let $\mathcal{S}$ be a flow. We write $s \xrightarrow{a} s'$ for $\varphi(s) = (a, s')$ and $s \overset{x}{\twoheadrightarrow} s'$ if and only if either

- $x = \epsilon$ and $s = s'$, or
- $x = ax'$ and there is an $s'' \in S$ such that $s \xrightarrow{a} s''$ and $s'' \overset{x'}{\twoheadrightarrow} s'$.

The objects satisfying formulas of the modal μ-calculus of one modal "next"-operator are points in flows, or alternatively, *pointed flows*, i.e., pairs $(\mathcal{S}, s)$, where s is a point in $\mathcal{S}$. However, in this thesis, it will be very convenient to restrict our attention to pointed flows of a certain kind, namely those where the flow is *generated* by the designated point.

Definition 3.3. Let $\mathcal{S} = (S, \sigma)$ be a flow and let $s \in \mathcal{S}$. The flow $\mathcal{S}_s = (S_s, \sigma_s)$ is defined by letting $S_s = \{\, s' \mid \exists x\, s \overset{x}{\twoheadrightarrow} s' \,\}$ and $\sigma_s = \sigma \restriction S_s$. We say that $\mathcal{S}_s$

is the *subflow of* $\mathcal{S}$ *generated by* s. If $\mathcal{S} = \mathcal{S}_s$ for some $s \in \mathcal{S}$, we say that $\mathcal{S}$ is *point-generated* (by s).

Definition 3.4. A *sourced flow* $\mathbb{S}$ is a structure $(\mathcal{S}, s)$, where $\mathcal{S}$ is generated by s, i.e., $\mathcal{S} = \mathcal{S}_s$. We call s the *source* of $\mathbb{S}$.

One might think that since the underlying flow is generated by the source, the extra mention of s is superfluous. The reason why we need to explicitely mention it is that a point which generates a point-generated flow need not be unique. For instance, the flow $\mathcal{S}$ in Figure 4 is generated by each of its points.

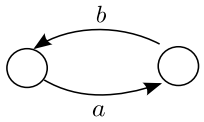


Figure 4: A flow with several generators.

It is well-known that if $(\mathcal{S}, s)$ is a pointed flow and φ is a modal μ-calculus formula, then φ is true for $(\mathcal{S}, s)$ if and only if φ is true for $(\mathcal{S}_s, s)$. So, our restriction to sourced flows as opposed to pointed flows does not cause any loss in information in this sense.

3.1 The coalgebraic perspective

As can be seen from its definition, a flow is a coalgebra over the functor $\Sigma \times \mathrm{id}$ (see Appendix A for a brief introduction to universal coalgebra). The benefit of having this perspective is that it enables us to use the standard toolkit of universal coalgebra in our arguments. We have already seen one of these tools, namely the definition of a *generated subflow* above. This, together with the following definitions of homomorphism, bisimulation, final flow and behavioral equivalence are precisely what the definitions from universal coalgebra boil down to if spelled out for our specific functor $\Sigma \times \mathrm{id}$.

First, to simplify notation, if $\mathcal{S}$ and $\mathcal{T}$ are flows, we will often write $f\colon \mathcal{S} \to \mathcal{T}$ for a function between the *carriers* of $\mathcal{S}$ to $\mathcal{T}$, and similarily for relations. The same applies to sourced flows, in the sense that if $\mathbb{S} = (\mathcal{S}_s, s)$ and $\mathbb{T} = (\mathcal{T}_t, t)$, we write $f\colon \mathbb{S} \to \mathbb{T}$ for a function between the carriers of the $\mathcal{S}_s$ and $\mathcal{T}_t$.

Definition 3.5. Let $\mathcal{S}$ and $\mathcal{T}$ be two flows. A function $f\colon \mathcal{S} \to \mathcal{T}$ is a *homomorphism* from $\mathcal{S}$ to $\mathcal{T}$ if $s \xrightarrow{a} s'$ in $\mathcal{S}$ implies $f(s) \xrightarrow{a} f(s')$ in $\mathcal{T}$ for all $s, s' \in S$.

For *sourced* flows, we also require that the function f maps initial points to initial points. I.e., if $\mathbb{S} = (\mathcal{S}_s, s)$ and $\mathbb{T} = (\mathcal{T}_t, t)$ are two sourced flows, then $f\colon \mathbb{S} \to \mathbb{T}$ is a homomorphism if and only if it is a homomorphism from $\mathcal{S}$ to $\mathcal{T}$ such that $f(s) = t$.

Note that in the absence of multiple successors, the "back" condition which occurs when defining bounded morphisms in the setting of Kripke frames becomes superfluous. For the same reason, the definition of a bisimulation simplifies to the following.

Definition 3.6. A relation $R \subseteq \mathcal{S} \times \mathcal{T}$ is called a *bisimulation* between $\mathcal{S}$ and $\mathcal{T}$ if $(s, t) \in R$ implies that s and t have the same color and their respective successors s' and t' are R related, i.e., $(s', t') \in R$.

Let s and t be points in two flows $\mathcal{S}$ and $\mathcal{T}$, respectively. We say that s and t are *bisimilar* , if there exists a bisimulation $R \subseteq \mathcal{S} \times \mathcal{T}$ such that $(s,t) \in R$.

We now lift this relation on points of flows to sourced flows by saying that two sourced flows $(\mathcal{S}_s, s)$ and $(\mathcal{T}_t, t)$ are *bisimilar*, written $(\mathcal{S}_s, s) \leftrightarrows (\mathcal{T}_t, t)$, if s and t are bisimilar.

The primary aim of this thesis will be to define acceptance of a sourced flow by an automaton. We get our inspiration from the fact that infinite words can themselves be seen as a special case of sourced flows, namely the infinite ones.

3.2 Infinite words as sourced flows

An infinite word u is a function from ω to Σ, and can thus be seen as a *coloring* of the set ω. However, in our case, the essential point of the matter is that ω can be seen as a *transition structure* with an initial point (namely 0) and where each element has a unique successor. Indeed, we *identify* an infinite word $u\colon \omega \to \Sigma$ with the sourced flow whose carrier is the set ω, where the transition function is $\lambda i.(u(i), i+1)$ and where the initial point is 0.

$$0 \overset{u(0)}{\to} 1 \overset{u(1)}{\to} 2 \overset{u(2)}{\to} \dots$$

Seen through this perspective, the infinite words have a special place amongst the pointed flows as seen by the following two propositions.

Proposition 3.7. Two infinite words u and v are bisimilar if and only if they are identical.

Proof. By simple induction using the definition of bisimilarity in the successor steps. □

Proposition 3.8. For every sourced flow $\mathbb{S}$ there is a unique bisimilar infinite word $\overrightarrow{\mathbb{S}}$, called the *unraveling* of $\mathbb{S}$. In fact, there is a unique homomorphism $U_{\mathbb{S}}\colon \overrightarrow{\mathbb{S}} \to \mathbb{S}$.

The "U" in the above proposition is for "unravel", although, strictly speaking, the unraveling is in the other direction. See Figure 5.

Proof. Let $\mathbb{S} = (\mathcal{S}_{s_0}, s_0)$ be a sourced flow. We inductively create a sequence $s_0 s_1 \dots$ of states of $\mathcal{S}$ and a sequence $a_0 a_1 \dots$ of symbols of Σ by setting a_i and s_{i+1} to be the unique pair such that $s_i \overset{a_i}{\to} s_{i+1}$ (note that s_0 is already given). The unraveling $\overrightarrow{\mathbb{S}}$ of $\mathbb{S}$ is the infinite word $a_0 a_1 \dots$, i.e.,

$$0 \overset{a_0}{\to} 1 \overset{a_1}{\to} 2 \overset{a_2}{\to} \dots$$

and the homomorphism is quite simply the sequence $s_0 s_1 \dots$ when seen as a function from ω to $\mathcal{S}$.

The uniqueness of $a_0 a_1 \dots$ follows from the preceding proposition and the uniqueness of the homomorphism follows from the fact that *any* homomorphism $f\colon \mathbb{S} \to \mathbb{T}$ must be unique, since the definition of homomorphism on sourced flows yields a unique inductive definition of f. □

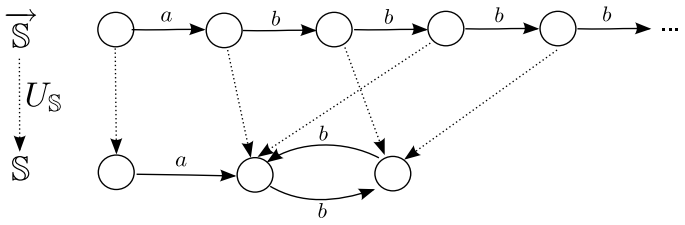


Figure 5: Unwinding a sourced flow $\mathbb{S}$

From this we obtain a simple way of checking whether two sourced flows are bisimilar.

Corollary 3.9. Two sourced flows $\mathbb{S}$ and $\mathbb{T}$ are bisimilar if and only if $\overrightarrow{\mathbb{S}} = \overrightarrow{\mathbb{T}}$.

Proof. Assume that $\mathbb{S}$ and $\mathbb{T}$ are bisimilar. It is easy to check that bisimilarity is a transitive relation over sourced flows, and hence $\overrightarrow{\mathbb{S}}$ and $\overrightarrow{\mathbb{T}}$ are bisimilar as well. But then $\overrightarrow{\mathbb{S}} = \overrightarrow{\mathbb{T}}$. The other direction is similarily straight forward. □

In fact, there is another way to view the unraveling of a pointed flow $\mathbb{S}$ with initial point s, namely as the *behavior* of s. The notion of behavioral equivalence is a fairly abstract notion (see Appendix A), but in our case it becomes simplified, since the category of flows has a *final* object.

Fact 3.10. We write $\mathbf{\Sigma}^\omega$ for the flow whose elements are infinite words $u \in \Sigma^\omega$ and where the transition function is defined as $au \xrightarrow{a} u$ for all $a \in \Sigma$, $u \in \Sigma^\omega$.

The flow $\mathbf{\Sigma}^\omega$ is final in the category of all flows, i.e., for every flow $\mathcal{S}$ there is a unique homomorphism $!_{\mathcal{S}} \colon \mathcal{S} \to \mathbf{\Sigma}^\omega$, which is defined by

$$!_{\mathcal{S}}(s) = \overrightarrow{(\mathcal{S}_s, s)}.$$

We say that two pointed flows $(\mathcal{S}, s)$ and $(\mathcal{T}, t)$ are *behaviorally equivalent* if $!_{\mathcal{S}}(s) = !_{\mathcal{T}}(t)$. However, this is equivalent to saying that $\overrightarrow{(\mathcal{S}_s, s)} = \overrightarrow{(\mathcal{T}_t, t)}$ which in turn is equivalent to saying that the two sourced flows $(\mathcal{S}_s, s)$ and $(\mathcal{T}_t, t)$ are bisimilar.

It is easy to see that a source-pointed flow $\mathbb{S}$ is infinite (has an infinite carrier) if and only if it is (isomorphic to) an infinite word. Now, since one of the aims of our thesis is to come up with a natural definition of an automaton accepting a sourced flow, and we already know how to deal with infinite words, we are more interested in the *finite* sourced flows. These are the ones in which the successor function eventually *loops*. If we go back again to the modal μ-calculus, these can be seen as the finite models of the logic and are therefore interesting in their own right.

3.3 Lassos

Definition 3.11. A finite sourced flow $\mathbb{S} = (\mathcal{S}_s, s)$ (i.e., where the carrier of $\mathcal{S}_s$ is finite) is called a *lasso*. The least subset of $\mathcal{S}_s$ which is closed under successor is called the *loop* of $\mathbb{S}$. The set of remaining points is called the *spoke* of the lasso.

We give a special name to the point of the loop which is closest to the source.

Definition 3.12. Let $\mathbb{S}$ be a sourced flow with source s. For each point t of $\mathbb{S}$ we say that $d(s) = \min\{|x| \mid s \overset{x}{\twoheadrightarrow} t\}$ is the *distance* from the source to t.

Definition 3.13. If $\mathbb{S}$ is a lasso with source s, we call the point t of the loop with the least distance from s the *knot point* of $\mathbb{S}$.

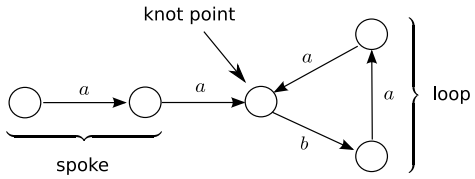


Figure 6: A lasso showing its parts.

Let $\mathbb{S}$ be a lasso with source s and knot point r. Let x be the least string such that $s \overset{x}{\twoheadrightarrow} r$ and let y be the least string such that $r \overset{y}{\twoheadrightarrow} r$. It is easy to see that $\overrightarrow{\mathbb{S}} = xy^\omega$ and that x and y uniquely determine $\mathbb{S}$ up to isomorphism, in the sense that the lasso (x, y) defined below will be isomorphic to $\mathbb{S}$.

Definition 3.14. Given finite strings x and y, where y is nonempty, we write (x, y) for the lasso whose carrier is the set $\{0, 1, \ldots, |xy|-1\}$; where the transition function σ is defined by

$$\sigma(i) = \begin{cases} (x(i), i+1) & \text{if } i < |x|, \\ (y(i), i+1) & \text{if } |x| \leq i < |xy| - 1, \\ (y(i), |x|) & \text{if } i = |xy| - 1; \end{cases}$$

and where the source is 0. For an example, see figure 7.

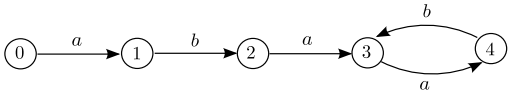


Figure 7: The lasso (aba, ab).

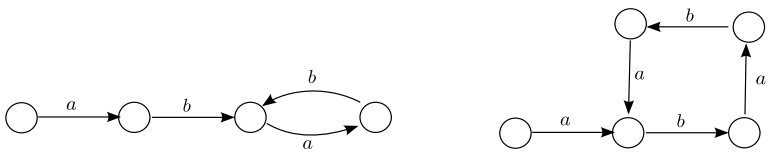


Figure 8: Two bisimilar but non-isomorphic lassos.

Since the representation (x, y) uniquely determines a lasso and also gives all the information we need about the spoke and loop part, we will henceforth call the string x the *spoke* of (x, y) and similarily the string y its *loop* . Please note

that this is a name conflict, but we will try to be clear so as to not cause any confusion.

By Corollary 3.9 we know that two lassos (x, y) and (z, w) are bisimilar if and only if $xy^\omega = zw^\omega$. However, we could still have $x \neq z$ or $w \neq y$. For instance, (ab, ab) is bisimilar to $(a, baba)$ (see Figure 8), but they are not isomorphic.

We will not differentiate between isomorphic structures in this thesis. Hence, we can safely speak of the *set* of sourced flows, which we write SFlow . We write Lasso for the set of all finite sourced flows, i.e., lassos, and Stream for the set of all infinite sourced flows, i.e., infinite words.

4 Automata on flows

We are now ready to define what it means for an automaton to accept an arbitrary sourced flow $\mathbb{S}$. There are some requirements that we want this definition to satisfy. We want it to be conservative with regards to infinite words (seen as sourced flows) and we want it to be *bisimulation invariant*, i.e., if $\mathbb{S}$ and $\mathbb{T}$ are bisimilar, then an automaton $\mathbb{A}$ should accept $\mathbb{S}$ if and only if it accepts $\mathbb{T}$. The second requirement is to guarantee that we keep the equivalence in expressive power between automata and the modal fixed point logic when expanding from infinite words to the set of *all* flows. These two requirements also makes our definition conform with the general one in [22], although ours is different in that is does not use *games* to define acceptance, instead using homomorphisms and label-preserving maps. There are two primary reasons for this. One is that it enables us to give a definition of acceptance which for *finite* sourced flows is purely finitary, i.e., it does not need infinite objects as witnesses for acceptance. Secondly, using maps and homomorphisms allow us to see structural similarities between the automata and the sourced flows it accepts, giving us more control when pursuing our goal of finding characterization theorems.

4.1 Traces and runs

The idea of our definition is taken from the game theoretic semantics given in [22] and goes as follows. Take an automaton $\mathbb{A}$ and a sourced flow $\mathbb{S}$. We want to stepwise identify states in $\mathbb{S}$ with states of $\mathbb{A}$ in a label-preserving way. So, starting by setting s_0 to be the source of $\mathbb{S}$, we pick some element q_0 of $\mathbb{A}$. Next, we have that $s_0 \xrightarrow{a_0} s_1$ for some $a_0 \in \Sigma$, $s_1 \in \mathbb{S}$. We then pick some element $q_1 \in \mathbb{A}$ *such that* $q_0 \xrightarrow{a_0} q_1$. Continuing in this fashion, we obtain a sequence

$$(s_0, q_0, a_0), (s_1, q_1, a_1), (s_2, q_2, a_2), \dots \tag{3}$$

such that $s_i \xrightarrow{a_i} s_{i+1}$ and $q_i \xrightarrow{a_i} q_{i+1}$ holds for all $i \in \omega$. In the game theoretic perspective, this sequence is often called a *play* or a *match*.

Now, if we rewrite the sequence in (3) as

$$(q_0, s_0, 0) \xrightarrow{a_0} (q_1, s_1, 1) \xrightarrow{a_1} (q_2, s_2, 2) \xrightarrow{a_2} \cdots$$

we see that this is in fact a sourced flow, which we will designate $\mathbb{R}$. The accute reader might notice that it is isomorphic to the infinite word $a_0 a_1 \dots$ which is also the *unraveling* of the sourced flow $\mathbb{S}$. This, however, is a bit outside the point (although it will become important later on). The main observation is that the following two statements hold:

- The function $\rho \colon \mathbb{R} \to \mathbb{A}$ sending a state (q_n, s_n, n) to q_n is *label-preserving* in the sense that if $r \xrightarrow{a} r'$ in $\mathbb{R}$, then $\rho(r) \xrightarrow{a} \rho(r')$ in $\mathbb{A}$.
- The function $h \colon \mathbb{R} \to \mathbb{S}$ sending a state (q_n, s_n, n) to s_n is a *homomorphism*.

Hence, this "stepwise" identification of states yielded the following situation.

$$\begin{array}{ccc} & \mathbb{R} & \\ {\scriptstyle \rho}\swarrow & & \searrow{\scriptstyle h} \\ \mathbb{A} & & \mathbb{S} \end{array} \tag{4}$$

Notice how this diagram is similar, but not identical, to the general coalgebraic notion of a bisimulation. The difference is that the automaton $\mathbb{A}$ is not a sourced flow and ρ is not a homomorphism (and could not reasonably be required to be one, since the automaton $\mathbb{A}$ has a different coalgebraic signature).

We now use the diagram in (4) to give us our definition of a *run*, beginning by giving the function ρ a special name.

Definition 4.1. Let $\mathbb{A}$ be an automaton and $\mathbb{S}$ be a sourced flow. We say that a function $\tau\colon \mathbb{S} \to \mathbb{A}$ is an $\mathbb{S}$-labeled *trace* through $\mathbb{A}$ if

$$s \xrightarrow{a} s' \implies \tau(s) \xrightarrow{a} \tau(s')$$

for all $s, s' \in \mathbb{S}$, $a \in \Sigma$.

Definition 4.2. Let $\mathbb{A}$ be an automaton and $\mathbb{S}$ a sourced flow. A *run of* $\mathbb{A}$ *on* $\mathbb{S}$ is a trace $\rho\colon \mathbb{R} \to \mathbb{A}$ together with a homomorphism $h\colon \mathbb{R} \to \mathbb{S}$.

Note that any homomorphism between two sourced flows must be unique if it exists, since it maps the source of the first to the source of the other, and successor points to successor points. So, in the definition above, the actual nature of the homomorphism h does not matter, as long as it exists. Hence, as a convention, *we will instead say* "let $\rho\colon \mathbb{R} \to \mathbb{A}$ be a run of $\mathbb{A}$ on $\mathbb{S}$", implying that $\mathbb{R}$ is a homomorphic preimage of $\mathbb{S}$. Often (but not always), we will make a typographical distinction and write ρ for traces which are also runs, and τ otherwise.

To determine whether a run ρ of $\mathbb{A}$ on a sourced flow $\mathbb{S}$ is accepting or not, we need to look at the *infinite behavior* of ρ, when seen as a trace. Recall that in the case for infinite words, acceptance was determined by looking at the set of states of the automaton that were was visited infinitely many times during the run. In our generalized case, this definition won't do as it stands, but we can reformulate it in such a way that it applies. Instead of asking "which states of the automaton gets mapped to infinitely often by ρ", we ask ourselves "which states q of the automaton are such that wherever you start in the run, q will eventually show up". Formalizing this gives us the following definition.

Definition 4.3. Let $\tau\colon \mathbb{S} \to \mathbb{A}$ be a trace. For each $s \in \mathbb{S}$, let S_s be the set of all points $s' \in \mathbb{S}$ such that $s \xrightarrow{x} s'$ for some $x \in \Sigma^*$. The *infinite behavior of* τ, $\inf(\tau)$, is then defined by

$$\inf(\tau) := \bigcap_{s \in \mathbb{S}} \tau[S_s].$$

The definition of whether a trace is successful (accepting) or not becomes the same as the one we saw for infinite words in Definition 2.9.

Definition 4.4. A trace $\tau\colon \mathbb{S} \to \mathbb{A}$ is said to be *successful* if $\inf(\tau) \in Acc_{\mathbb{A}}$. It is called *accepting* if it is successful and starts in an initial point of $\mathbb{A}$, i.e., if $\tau(s_0) \in I_{\mathbb{A}}$, where s_0 is the source of $\mathbb{S}$.

These definitions "lift" directly to runs by saying that a run $\rho\colon \mathbb{R} \to \mathbb{A}$ is successful (accepting) if the *trace* ρ is successful (accepting). From this we obtain the following definition of acceptance.

Definition 4.5. Let $\mathbb{A}$ be an automaton. We say that a sourced flow $\mathbb{S}$ is *accepted* by a point q in $\mathbb{A}$ if there is a successful run ρ of $\mathbb{A}$ on $\mathbb{S}$ which starts in q. We write $\mathcal{L}(q, \mathbb{A})$ (or simply $\mathcal{L}(q)$, if the automaton $\mathbb{A}$ is implicit) for the set of all sourced flows accepted by q.

Similarily, we say that $\mathbb{S}$ is *accepted* by $\mathbb{A}$ if there is an accepting run ρ of $\mathbb{A}$ on $\mathbb{S}$. We write $\mathcal{L}(\mathbb{A})$ for the set of all sourced flows accepted by $\mathbb{A}$.

For an automaton $\mathbb{A}$ with initial states I, we immediately get that

$$\mathcal{L}(\mathbb{A}) = \bigcup_{q_I \in I} \mathcal{L}(q_I).$$

We now show that the definition yields the results we want, namely being conservative with regards to the standard definition over the set of infinite words; being bisimulation invariant; and giving a finitary definition of acceptance for finite sourced flows.

Proposition 4.6. A automaton $\mathbb{A}$ accepts the coalgebraic representation of an infinite word u (in the sense of Definition 4.5) if and only if it accepts the infinite word u in the ordinary sense (i.e., in the sense of Definition 2.10).

Proof. The result follows from a simple comparison of the two definitions, together with the observation that for infinite sourced flows, Definition 4.3 reduces to the ordinary notion of "infinite behavior". □

Now, in order to prove that the language accepted by an automaton is closed under bisimilarity, we prove the following statement.

Proposition 4.7. Let $\rho\colon \mathbb{R} \to \mathbb{A}$ be a run of an automaton $\mathbb{A}$ on a sourced flow $\mathbb{S}$ and let f be a homomorphism from some sourced flow $\mathbb{T}$ to $\mathbb{R}$. Then $\tau = \rho \circ f$ is a also a run of $\mathbb{A}$ on $\mathbb{S}$. Moreover, τ is successful (accepting) if and only if ρ is successful (accepting).

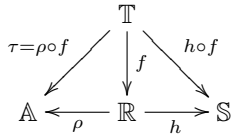


Figure 9: Runs are preserved by taking homomorphic preimages.

Proof. Since ρ is a run of $\mathbb{A}$ on $\mathbb{S}$, we know that $\mathbb{R}$ is a homomorphic preimage of $\mathbb{S}$, i.e., there is a homomorphism $h\colon \mathbb{R} \to \mathbb{S}$. It is easy to check that the composition of two homomorphisms is still a homomorphism, and hence $\mathbb{T}$ is a homomorphic preimage of $\mathbb{S}$, using the homomorphism $h \circ f$. Furthermore, it is easy to see that $\tau = \rho \circ f$ is a trace, since it is the composition of two label-preserving functions.

We now show that τ and ρ have the same infinite behavior, which shows that τ is successful if and only if ρ is. Using the notation of Definition 4.3, we first observe that $f[T_t] = R_{f(t)}$ for every $t \in \mathbb{T}$. This is because f is a

homomorphism and hence must map successor points to successor points. Our second observation is that f must be *surjective*. This again follows from the fact that it maps successors to successors, but now combined with the fact that it also is required to map the source of $\mathbb{T}$ to the source of $\mathbb{S}$. But then we must have

$$\begin{aligned}\inf(\tau) &= \bigcap_{t\in\mathbb{T}} \tau[T_t] \\ &= \bigcap_{t\in\mathbb{T}} (\rho \circ f)[T_t] \\ &= \bigcap_{t\in\mathbb{T}} \rho[f[T_t]] \\ &= \bigcap_{t\in\mathbb{T}} \rho[R_{f(t)}] \\ &= \bigcap_{r\in\mathbb{R}} \rho[R_r] \\ &= \inf(\rho).\end{aligned}$$

Now, since the success of a run is determined in terms of its infinite behavior this means that τ is successful if and only if ρ is. To realize that τ is accepting if and only if ρ is, it is enough to observe that if t_0 and r_0 are the sources of $\mathbb{T}$ and $\mathbb{R}$, respectively, then $\tau(t_0) = \rho(f(t_0)) = \rho(r_0)$. This is because f is a homomorphism, and therefore, by definition, must map the source of $\mathbb{T}$ to the source of $\mathbb{R}$. □

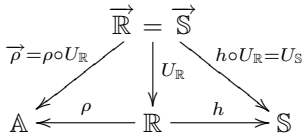
Corollary 4.8. Let $\mathbb{A}$ be an automaton and $\mathbb{S}$ a sourced flow. Then the following are equivalent:

(a) $\mathbb{S}$ is accepted by $\mathbb{A}$.

(b) $\overrightarrow{\mathbb{S}}$ is accepted by $\mathbb{A}$

(c) There is an $\overrightarrow{\mathbb{S}}$-labeled accepting trace through $\mathbb{A}$.

Proof. $(a) \Rightarrow (c)$. Assume that $\mathbb{A}$ accepts $\mathbb{S}$. Then there is some run $\rho\colon \mathbb{R} \to \mathbb{A}$ of $\mathbb{A}$ on $\mathbb{S}$. But then, according to Proposition 4.7, the function $\overrightarrow{\rho}\colon \overrightarrow{\mathbb{R}} \to \mathbb{A}$ defined by $\overrightarrow{\rho} := \rho \circ U_{\mathbb{R}}$ is an accepting run of $\mathbb{A}$ on $\mathbb{S}$. We then have that $\overrightarrow{\mathbb{R}}$ and $\mathbb{S}$ are bisimilar, since $\overrightarrow{\mathbb{R}}$ is a homomorphic preimage of $\mathbb{S}$. This means that $\overrightarrow{\mathbb{R}} = \overrightarrow{\mathbb{S}}$, which means that $\overrightarrow{\rho}$ is an $\overrightarrow{\mathbb{S}}$-labeled accepting trace.



$(c) \Rightarrow (b)$. Immediate.

$(b) \Rightarrow (a)$. Assume that there is an accepting run $\rho\colon \mathbb{R} \to \mathbb{A}$ of $\mathbb{A}$ on $\overrightarrow{\mathbb{S}}$. Then $\mathbb{R}$ is a homomorphic preimage of $\overrightarrow{\mathbb{S}}$ which means that $\mathbb{R} = \overrightarrow{\mathbb{S}}$ (or rather, $\mathbb{R}$ and $\overrightarrow{\mathbb{S}}$ are isomorphic, but we do not differentiate between isomorphic copies). But then $\rho\colon \mathbb{R} \to \mathbb{A}$ is also an accepting run of $\mathbb{A}$ on $\mathbb{S}$, since $\mathbb{R} = \overrightarrow{\mathbb{S}}$ is a homomorphic preimage of $\mathbb{S}$. □

Proposition 4.9. If $\mathbb{S}$ and $\mathbb{T}$ are two bisimilar sourced flows and $\mathbb{A}$ is an automaton, then $\mathbb{A}$ accepts $\mathbb{S}$ if and only if $\mathbb{A}$ accepts $\mathbb{T}$.

Proof. We know that $\mathbb{S}$ and $\mathbb{T}$ are bisimilar if and only if $\overrightarrow{\mathbb{S}} = \overrightarrow{\mathbb{T}}$. The result then follows from Corollary 4.8. □

Having shown that our definition behaves nicely with respect to infinite words and bisimulation, we are now ready to show our last claim, namely that for lassos, the definition can be made purely finitary. We will do this by defining the *finitary fragment of a regular language of sourced flows*, and looking at its properties. First, we introduce a special subset of the language accepted by an automaton, namely those sourced flows which are the domain of an accepting trace, as they will play an important part in the rest of the thesis.

4.2 Labels

In Proposition 4.8, we saw that an automaton $\mathbb{A}$ accepts a sourced flow $\mathbb{S}$ if and only if there is an accepting trace $\rho\colon \overrightarrow{\mathbb{S}} \to \mathbb{A}$. Hence, if $\mathbb{S}$ is infinite, i.e., if $\mathbb{S} = \overrightarrow{\mathbb{S}}$, then there is an accepting trace ρ whose label is $\mathbb{S}$. The reason why these are to become important is that the trace ρ will enable us to reason about similarities in structure between $\mathbb{A}$ and $\mathbb{S}$.

Definition 4.10. Let $\mathbb{A}$ be an automaton and q a state in $\mathbb{A}$. If $\mathbb{S}$ is the label of a successful trace through $\mathbb{A}$ which starts in q, i.e., if there is a successful trace $\rho\colon \mathbb{S} \to \mathbb{A}$, we say that $\mathbb{S}$ is a *successful label of* q. The set of successful labels of q is denoted $\mathrm{Label}(q, \mathbb{A})$, or simply $\mathrm{Label}(q)$ if the automaton $\mathbb{A}$ is implicit.

Similarily, we call $\mathbb{S}$ an *accepting label of* $\mathbb{A}$ if it is a successful label of an initial point of $\mathbb{A}$, i.e., if there exists an accepting trace $\rho\colon \mathbb{S} \to \mathbb{A}$. We write $\mathrm{Label}(\mathbb{A})$ for the set of accepting labels of $\mathbb{A}$.

Using this definition, we obtain the following proposition which is just a reformulation of Definition 4.5.

Proposition 4.11. The language $\mathcal{L}(\mathbb{A})$ accepted by an automaton $\mathbb{A}$ is the closure of the set $\mathrm{Label}(\mathbb{A})$ under homomorphic images.

Since $\mathrm{Label}(\mathbb{A})$ is closed under homomorphic preimages, it is not difficult to see that this is equivalent to saying that $\mathcal{L}(\mathbb{A})$ is the closure of $\mathrm{Label}(\mathbb{A})$ under bisimilarity.

If we let $\mathbb{A}$ be the automaton displayed in Figure 10, we get an example of when the two languages $\mathcal{L}(\mathbb{A})$ and $\mathrm{Label}(\mathbb{A})$ don't coincide. To see why, consider the lasso (a, b). Certainly the infinite word ab^ω is accepted by $\mathbb{A}$, so the lasso (a, b) is as well. However, there are *no* (a, b)-labeled traces through $\mathbb{A}$ as can be realized by asking which state of $\mathbb{A}$ the single point in the loop of (a, b) would be mapped to. In a sense, the righthand "loop" in $\mathbb{A}$ is too big. In Section 4.4 of this chapter, we will see that for *prophetic* automata, the two languages always coincide.

4.3 Regular lasso languages

We are very interested in the set of *lassos* accepted by an automaton $\mathbb{A}$, since, as we have mentioned, lassos can be seen as the *finite* models of the modal

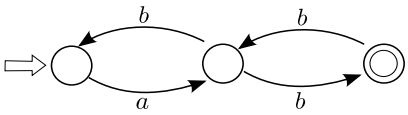


Figure 10: An example of when $\mathrm{Label}(\mathbb{A}) \neq \mathcal{L}(\mathbb{A})$

μ-calculus of one modal operator. Hence, by studying these in particular, we are doing *finite model theory* for this logic.

Recalling how we defined Lasso to be the set of finite sourced flows, and Stream as the set of infinite ones, we define the finitary and infinitary fragments of a language of sourced flows in the following way.

Definition 4.12. If $\mathcal{L}$ is a language of sourced flows, we define

$$\mathcal{L}_{\mathrm{fin}} := \mathcal{L} \cap \mathsf{Lasso}$$
$$\mathcal{L}_{\omega} := \mathcal{L} \cap \mathsf{Stream}.$$

These are respectively called the *finitary* and *infinitary* fragments of the language $\mathcal{L}$.

Proposition 4.13. The class of regular languages of sourced flows is closed under intersection, union and complementation.

Proof. This follows from the fact that the statement holds for the infinitary fragments. As an example we show closure under intersection.

Let $\mathcal{L}$ and $\mathcal{L}'$ be two regular languages of sourced flows. Consider the sets $\mathcal{L}_{\omega}$ and $\mathcal{L}'_{\omega}$ of infinite sourced flows, *seen as sets of infinite words*. Then both $\mathcal{L}_{\omega}$ and $\mathcal{L}'_{\omega}$ are ω-regular languages, and since ω-regular languages are closed under intersection, we have that $\mathcal{L}_{\omega} \cap \mathcal{L}'_{\omega}$ is ω-regular. But then the automaton which accepts $\mathcal{L}_{\omega} \cap \mathcal{L}'_{\omega}$ also accepts $\mathcal{L} \cap \mathcal{L}'$, since $\mathcal{L} \cap \mathcal{L}'$ is the closure of $\mathcal{L}_{\omega} \cap \mathcal{L}'_{\omega}$ under homomorphic images. □

From this we obtain the well-known result (although rephrased in our terminology) that regular languages are *determined* by their finitary fragment.

Theorem 4.14. Let $\mathcal{L}, \mathcal{L}'$ be regular languages of sourced flows. Then the following are equivalent:

(a) $\mathcal{L} = \mathcal{L}'$

(b) $\mathcal{L}_{\omega} = \mathcal{L}'_{\omega}$

(c) $\mathcal{L}_{\mathrm{fin}} = \mathcal{L}'_{\mathrm{fin}}$

Proof. That (a) ⇒ (b) is clear. For the case of (b) ⇒ (c), if $\mathcal{L}'_{\mathrm{fin}} \neq \mathcal{L}'_{\mathrm{fin}}$, we can, without loss of generality, assume that there is some lasso $\mathbb{S} \in \mathcal{L} \setminus \mathcal{L}'$. But then $\overrightarrow{\mathbb{S}} \in \mathcal{L} \setminus \mathcal{L}'$, since both $\mathcal{L}$ and $\mathcal{L}'$ are closed under bisimilarity. Hence, $\mathcal{L}_{\omega} \neq \mathcal{L}'_{\omega}$.

To prove that $(c) \Rightarrow (a)$, assume that $\mathcal{L} \neq \mathcal{L}'$. Without loss of generality, we can assume that $\mathcal{L} \setminus \mathcal{L}'$ is non-empty, and we want to show that $\mathcal{L} \setminus \mathcal{L}'$ contains some lasso, from which the result follows.

Since the class of regular languages is closed under complementation and intersection, $\mathcal{L} \setminus \mathcal{L}'$ is regular, and hence there is some Büchi automaton $\mathbb{A}$

which accepts it. Now, since $\mathbb{A}$ is finite and accepts *some* sourced flow $\mathbb{S}$ (since we assumed that $\mathcal{L} \setminus \mathcal{L}'$ was non-empty), it is not difficult to see that there must be some finite path from an initial point q_I to some accepting point q_F (say with label x), and a non-empty finite path from q_F back to itself (say with label y). But then the lasso (x, y) is an element of $\mathcal{L} \setminus \mathcal{L}'$. □

We define the following shorthands for the finitary fragments of the languages defined by an automaton $\mathbb{A}$.

Definition 4.15. Let $\mathbb{A}$ be an automaton. We define

$$\begin{aligned} \mathrm{Label}_{\mathrm{fin}}(q, \mathbb{A}) &:= \mathrm{Label}(q, \mathbb{A}) \cap \mathsf{Lasso} \\ \mathrm{Label}_{\mathrm{fin}}(\mathbb{A}) &:= \mathrm{Label}(\mathbb{A}) \cap \mathsf{Lasso} \\ \mathcal{L}_{\mathrm{fin}}(q, \mathbb{A}) &:= \mathcal{L}(q, \mathbb{A}) \cap \mathsf{Lasso} \\ \mathcal{L}_{\mathrm{fin}}(\mathbb{A}) &:= \mathcal{L}(\mathbb{A}) \cap \mathsf{Lasso} \end{aligned}$$

The language $\mathcal{L}_{\mathbb{A}}$ is called the *finitary language, or lasso language, accepted by* $\mathbb{A}$. We say that a language $\mathcal{L}$ of lassos is a *regular lasso language* if there is an automaton $\mathbb{A}$ such that $\mathcal{L}$ is the lasso language accepted by $\mathbb{A}$.

What we want is a purely finitary definition of a regular lasso-language, since these correspond to the definable classes of finitary models in the modal μ-calculus. As we saw in Proposition 4.11, the language accepted by $\mathbb{A}$ is the closure of $\mathrm{Label}(\mathbb{A})$ under homomorphic images. For a finitary version of this, we would like the lasso-language $\mathcal{L}_{\mathrm{fin}}(\mathbb{A})$ accepted by an automaton $\mathbb{A}$ to be the closure of $\mathrm{Label}_{\mathrm{fin}}(\mathbb{A})$ under homomorphic images. Calling a run $\rho\colon \mathbb{R} \to \mathbb{A}$ on a sourced flow $\mathbb{S}$ *finite* if $\mathbb{R}$ is finite, this is equivalent to requiring that whenever there exists an *infinite* accepting run ρ of an automaton $\mathbb{A}$ on a *finite* sourced flow $\mathbb{S}$, then there also exists a *finite* run ρ' of $\mathbb{A}$ on $\mathbb{S}$.

In fact, we can do even better, as we can give a bound on how "big" the run has to be, in terms of the size of the automaton.

Theorem 4.16. Let $\mathbb{A}$ be an automaton with k states and let (x, y) be a lasso in $\mathcal{L}(\mathbb{A})$. Then there are $n < k + k^3$ and $m < k^3$, dependent on (x, y), such that $(xy^n, y^m) \in \mathrm{Label}_{\mathrm{fin}}(\mathbb{A})$.

These upper bounds are probably not optimal, since there is much room for "tweaking" the proof in order to obtain better results.

Proof. Assume that $\mathbb{A}$ has k states and that $(x, y) \in \mathcal{L}(\mathbb{A})$. Then there must be an accepting trace τ through $\mathbb{A}$ whose label is the infinite word xy^ω. We show that we can assume that τ is of a certain form, which will enable us to create a finite accepting trace with the required properties.

Our infinite trace τ consists of a label preserving function from the flow representation of the infinite word xy^ω. Writing $x(i)$ for the i:th symbol of x (indexing from zero), and similarily for y, we have a sequence of states $q_0, q_1, \ldots$ of the automaton such that

$$q_0 \overset{x(0)}{\to} q_1 \overset{x(1)}{\to} \cdots \overset{x(|x|-1)}{\to} q_{|x|} \overset{y(0)}{\to} q_{|x|+1} \overset{y(1)}{\to} \cdots \overset{y(|y|-1)}{\to} q_{|x|+|y|} \overset{y(0)}{\to} \cdots$$

We are especially interested in the states $q_{|x|+|y|\cdot i}$ for different values of $i \in \omega$, so we write $q^i := q_{|x|+|y|\cdot i}$ and can then write the trace above as

$$q_0 \overset{x}{\twoheadrightarrow} q^0 \overset{y}{\twoheadrightarrow} q^1 \overset{y}{\twoheadrightarrow} \dots$$

Since τ is accepting, there is a set $F \in Acc_{\mathbb{A}}$ such that $\inf(\tau) = F$. This means that after some point in the sequence $q_0, q_1, \dots$, only elements of F occurs. So we can let m_0 be the least natural number such that

$$\{\, \tau(i) \mid i \geq |x| + |y| \cdot m_0 \,\} = F.$$

Now, if we start at some point q_i where i is greater than $|x| + |y| \cdot m_0$, then there must be some $j \geq i$ such that $\{\, q_i, q_{i+1}, \dots, q_j\} = F$. Moreover, for every j' greater than j, $\{q_i, q_{i+1}, \dots, q_{j'}\}$ is equal to F as well. Writing $[q^i, q^j]$ for the inclusive interval between the points q^i and q^j, i.e.,

$$[q^i, q^j] := \{\, q_n \mid |x| + |y| \cdot i \leq n \leq |x| + |y| \cdot j \,\}$$

we can therefore inductively define m_{i+1} to be the least such that $[q^{m_i}, q^{m_{i+1}}] = F$. Then, if we consider the trace τ up to q^{m_k},

$$q_0 \overset{xy_0^m}{\twoheadrightarrow} q^{m_0} \overset{y^{m_1-m_0}}{\twoheadrightarrow} q^{m_1} \overset{y^{m_2-m_1}}{\twoheadrightarrow} q^{m_1} \overset{y^{m_3-m_2}}{\twoheadrightarrow} \cdots \overset{y^{m_k-m_{k-1}}}{\twoheadrightarrow} q^{m_k},$$

at least two of the points in the set $\{q^{m_0}, q^{m_1}, \dots, q^{m_k}\}$ must be identical, i.e., there are $a < b \leq k$ such that $q^{m_a} = q^{m_b}$. But this means that we have found a "loop" in the automaton, by which we mean that the string $y^{m_b-m_a}$ is the label of a finite path from q^{m_a} back to itself. Now, by construction, this finite path passes through every element of F. So, moving first by from q_0 to q^{m_a} by means of the finite xy^{m_a}-labeled path $q_0, q_1, \dots, q^{m_a}$ and then back from q^{m_a} to itself by means of the $y^{m_b-m_a}$-labeled path we just found, gives us an accepting trace whose label is $(xy^{m_a}, y^{m_b-m_a})$.

Finally, we show that we can assume that $m_0 < k$ and $m_k - m_0 < k^3$, concluding our proof. For the first statement, observe the trace up to q^{m_0}:

$$q_0 \overset{x}{\twoheadrightarrow} q^0 \overset{y}{\twoheadrightarrow} q^1 \overset{y}{\twoheadrightarrow} q^2 \overset{y}{\twoheadrightarrow} \cdots \overset{y}{\twoheadrightarrow} q^{m_0}$$

If $m_0 \geq k$, then two of the states in the set $\{q^0, q^1, \dots, q^{m_0}\}$ must be equal, and hence we can *remove* the segment between these two points without changing the label or the acceptance of the trace. Hence, it is safe to assume that $m_0 < k$.

For the second statement, we prove that for every $i < k$, $m_{i+1} - m_i$ can be assumed to be less than k^2. This suffices because then $m_k - m_0 = (m_1 - m_0) + (m_2 - m_1) + \ldots + (m_k - m_{k-1}) < k^3$.

Consider the trace between q^{m_i} and $q^{m_{i+1}}$ for some choice of i:

$$q^{m_i} \overset{y}{\twoheadrightarrow} q^{m_i+1} \overset{y}{\twoheadrightarrow} \cdots \overset{y}{\twoheadrightarrow} q^{m_{i+1}-1} \overset{y}{\twoheadrightarrow} q^{m_{i+1}}$$

The idea is that this path must "accumulate" states from F one by one, allowing us to split it into subpaths which we then show can be assumed to be bounded in length. We start by letting $l_0 = m_i$ and inductively setting l_{j+1} to be the least natural number $\geq l_j$ such that

$$[q^{l_0}, q^{l_j}] \subsetneq [q^{l_0}, q^{l_{j+1}}].$$

In other words, we start at q^{l_0}, move stepwise through the trace, and increase the "counter" variable j each time we are at a point q^n (for some n), having encountered an element of F which we haven't seen before. Now, since we start by having "seen" q^{l_0}, the counter variable j can only increase to some number $h < k$. In other words, we know that the sizes of the sets

$$[q^{l_0}, q^{l_0}] \subsetneq [q^{l_0}, q^{l_1}] \subsetneq \cdots \subsetneq [q^{l_0}, q^{l_h}]$$

are strictly increasing, but are bounded by $|F| \leq k$. By our choice of m_{i+1} we must have a situation which looks like this:

$$q^{m_i} = \underbrace{q^{l_0} \overset{y^{l_1-l_0}}{\twoheadrightarrow} q^{l_1} \overset{y^{l_2-l_1}}{\twoheadrightarrow} q^{l_2} \overset{y^{l_3-l_2}}{\twoheadrightarrow} \cdots \overset{y^{l_h-l_{h-1}}}{\twoheadrightarrow} q^{l_h}}_{\leq k \text{ intervals}} = q^{m_{i+1}}$$

The final step is now to give a bound for $l_{j+1} - l_j$. Consider the segment of the trace between the points q^{l_j} and $q^{l_{j+1}}$:

$$\underbrace{q^{l_j} \overset{y}{\twoheadrightarrow} q^{l_j+1} \overset{y}{\twoheadrightarrow} \cdots \overset{y}{\twoheadrightarrow} q^{l_{j+1}-1}}_{\text{no new points of } F \text{ here}} \overset{y}{\twoheadrightarrow} q^{l_{j+1}}$$

Certainly, if $l_{j+1} - l_j \geq k$, then the set $\{q^{l_j}, q^{l_j+1}, \ldots, q^{l_{j+1}}\}$ would have more than k elements. Therefore we should be able to reason as in the case of m_0, and remove the segment of the trace between these two points. However, in doing so we might accidentally remove a point such that the interval $[q^{m_i}, q^{m_{i+1}}]$ no longer contains every element of F. Luckily, the minimality of l_{j+1} guarantees that $[q^{m_i}, q^{l_j}] = [q^{m_i}, q^{m_{j+1}-1}]$, so removing segments in the interval $[q^{l_j}, q^{l_{j+1}-1}]$ is safe. So, we can argue as follows: if $l_{j+1} - l_j$ is *strictly* greater than k, then the set $\{q^{l_j}, q^{l_j+1}, \ldots, q^{l_{j+1}-1}\}$ has cardinality greater than k and hence there are two points which are equal. By the above discussion, removing the segment between these two points does not alter the label of the trace, nor the fact that $[q^{m_i}, q^{m_{i+1}}] = F$. Hence, we may safely assume that $l_{j+1} - l_j \leq k$. But then

$$\begin{aligned} m_{i+1} - m_i &= \sum_{j=0}^{j=h-1} (l_{j+1} - l_j) \\ &\leq \sum_{j=0}^{j=h-1} k \\ &< k^2, \end{aligned}$$

as required. □

Having these upper bounds enables us to give a *uniform* size for the witnessing lasso in Label(𝔸).

Corollary 4.17. Let 𝔸 be an automaton with k states. Then

$$(x, y) \in \mathcal{L}(\mathbb{A}) \iff (xy^{k+k^3}, y^{k^3!}) \in \text{Label}(\mathbb{A}).$$

Proof. The main observation to make here is that if we take $x \in \Sigma^*$, $y \in \Sigma^+$, $n \geq 0, m \geq 1$, and consider the lasso (xy^n, y^m), then for any $n' \geq n$ and any multiple m' of m, (xy^n, y^m) is a homomorphic image of $(xy^{n'}, y^{m'})$.

Hence, the direction from right to left is immediate, since (x, y) is a homomorphic image of $(xy^{k+k^3}, y^{k^3!})$. For the other direction, assume that (x, y) is accepted by $\mathbb{A}$. Then there are $n < k+k^3, m < k^3$ such that $(xy^n, y^m) \in \mathrm{Label}(\mathbb{A})$. By choosing $n' = k + k^3$ and $m' = k^3!$, we get $n' > n$ and that m' is a multiple of m; hence we obtain the wanted result. □

We also get the result we want, namely that a finite sourced flow $\mathbb{S}$ is accepted by an automaton $\mathbb{A}$ if and only if there is a *finite* run $\rho\colon \mathbb{R} \to \mathbb{A}$ of $\mathbb{A}$ on $\mathbb{S}$.

Corollary 4.18. If $\mathbb{A}$ is an automaton, then $\mathcal{L}_{\mathrm{fin}}(\mathbb{A})$ is the closure of $\mathrm{Label}_{\mathrm{fin}}(\mathbb{A})$ under homomorphic images.

Since $\mathrm{Label}_{\mathrm{fin}}(\mathbb{A})$ is closed under homomorphic preimages, restricted to the set of lassos, it is not difficult to check that $\mathcal{L}_{\mathrm{fin}}(\mathbb{A})$ is also the closure of $\mathrm{Label}_{\mathrm{fin}}(\mathbb{A})$ under bisimilarity, restricted to the set of lassos. This can also be realized by noting that $\mathcal{L}_{\mathrm{fin}}(\mathbb{A})$ *must* be closed under bisimilarity restricted to the set of lassos, since $\mathcal{L}(\mathbb{A})$ is closed under bisimilarity over the entire class of sourced flows.

4.4 Prophetic automata and labels

As we have seen, it is not in general the case that the set $\mathrm{Label}(\mathbb{A})$ of accepting labels of an automaton $\mathbb{A}$ coincides with the set of *accepted* sourced flows. It turns out that the set $\mathrm{Label}(\mathbb{A})$ is easier to investigate than $\mathcal{L}(\mathbb{A})$, because of the similarities in structure between the lassos in $\mathrm{Label}(\mathbb{A})$ and the automaton $\mathbb{A}$ itself.

Here we show that for a *prophetic* Büchi automaton $\mathbb{A}$, as defined in section 2.3, the two sets $\mathcal{L}(\mathbb{A})$ and $\mathrm{Label}(\mathbb{A})$ are equal. In fact, we get a stronger property, namely that the sets $\mathrm{Label}(q)$ for each $q \in \mathbb{A}$, constitutes a partition of the class of all pointed flows, where each class is closed under bisimulation.

Theorem 4.19. Let $\mathbb{A}$ be a Büchi automaton. Then $\mathbb{A}$ is prophetic if and only if for every sourced flow $\mathbb{S}$, there is a unique successful trace $\tau\colon \mathbb{S} \to \mathbb{A}$.

Proof. Note that an automaton is prophetic if and only if this theorem holds when restricted to the class of infinite words, i.e., the infinite sourced flows. Hence, the direction from right to left is immediate.

For the direction from left to right, assume that $\mathbb{A}$ is prophetic. Then, for every infinite word u there is a unique successful trace $\tau_u\colon u \to \mathbb{A}$ (remember that we identify infinite words with their sourced flow representation).

Now, take an arbitrary sourced flow $\mathbb{S}$ and let $u = \overrightarrow{\mathbb{S}}$ be its unraveling. We know that there is a unique homomorphism $U_{\mathbb{S}}\colon u \to \mathbb{S}$, and by proposition 4.7, we know that *if* there is a successful trace

$$\tau\colon \mathbb{S} \to \mathbb{A},$$

then

$$\tau \circ U_{\mathbb{S}}\colon u \to \mathbb{A}$$

is a successful trace as well. By the uniqueness of τ_u, this would imply that $\tau_u = \tau \circ U_{\mathbb{S}}$, and hence there is really just one choice for τ, namely the function which maps a state $s \in \mathbb{S}$ to the unique state $q \in \mathbb{A}$ such that if $U_{\mathbb{S}}(i) = s$ for any $i \in \overrightarrow{\mathbb{S}}$, then $\tau(i) = q$ (provided that such a state q exists).

So, it suffices to show that if $i, j \in \overrightarrow{S}$ are such that $U_{\mathbb{S}}(i) = U_{\mathbb{S}}(j)$, then $\tau(i) = \tau(j)$. However, if $U_{\mathbb{S}}(i) = U_{\mathbb{S}}(j)$, then the points i and j are behaviorally equivalent, and hence the suffixes of u generated by i and j, respectively, are identical.

$$0 \xrightarrow{a_0} 1 \xrightarrow{a_1} \dots \xrightarrow{a_{i-1}} i \xrightarrow{a_i} i+1 \xrightarrow{a_{i+1}} \dots \xrightarrow{a_{j-1}} j \xrightarrow{a_j} j+1 \xrightarrow{a_{j+1}} \dots$$

Because of the uniqueness of the successful traces, if $v \in \Sigma^\omega$ is a suffix of u, then τ_v must conform with τ_u, i.e., $\tau_v = \tau_u \restriction v$. Hence, $\tau(i) = \tau(j)$. □

From this we obtain a corollary which is analoguous to Proposition 2.20.

Corollary 4.20. An automaton $\mathbb{A}$ is prophetic if and only if $\{ \operatorname{Label}(q) \mid q \in \mathbb{A} \}$ partitions SFlow.

We also obtain the result which was mentioned in the beginning of this subsection, namely that for prophetic automata $\mathbb{A}$, $\operatorname{Label}(\mathbb{A})$ is closed under bisimilarity. This is implied by the following corollary.

Corollary 4.21. If $\mathbb{A}$ is prophetic, then $\operatorname{Label}(q)$ is closed under bisimilarity for every $q \in \mathbb{A}$.

Proof. Take a prophetic automaton $\mathbb{A}$ and consider two bisimilar sourced flows $\mathbb{S}$ and $\mathbb{T}$. There are unique $q, q' \in \mathbb{A}$ such that $\mathbb{S} \in \operatorname{Label}(q)$ and $\mathbb{T} \in \operatorname{Label}(q')$. Then $\overrightarrow{\mathbb{S}} \in \operatorname{Label}(q)$ and $\overrightarrow{\mathbb{T}} \in \operatorname{Label}(q')$. Because $\mathbb{S}$ and $\mathbb{T}$ are bisimilar, we have $\overrightarrow{\mathbb{S}} = \overrightarrow{\mathbb{T}}$ and hence we must have $q = q'$. □

Prophetic automata can be characterized in yet another way, showing a connection to the final flow (the final object in the category of flows). The result will not be used in the rest of the thesis, but is mentioned here anyway since we think it is an interesting result on its own. Let $\mathbb{A}$ be an automaton and $\mathcal{S}$ be a (non-pointed) flow. We temporarily call a function $\tau \colon \mathcal{S} \to \mathbb{A}$ a *successful trace* if, for every $s \in \mathcal{S}$, τ restricted to the sourced flow $(\mathcal{S}_s, s)$ is a successful trace in the ordinary sense.

Theorem 4.22. An automaton $\mathbb{A}$ is prophetic if and only if there is surjective successful trace $\tau \colon \mathbf{\Sigma}^\omega \to \mathbb{A}$ which is unique with respect to being successful.

Proof. The unique successful trace is the one mapping an infinite word u to the unique point q such that $u \in \mathcal{L}(q)$. We omit the details since they are straightforward and reminiscient of the proof of Theorem 4.19. □

5 Characterizating regular lasso languages

Recall from Definition 4.15 that a set $\mathcal{L} \subseteq \mathsf{Lasso}$ is called a *regular lasso language* if it is the lasso fragment of a regular language of pointed flows, i.e., if there is an automaton $\mathbb{A}$ such that $\mathcal{L} = \mathcal{L}_{\text{fin}}(\mathbb{A})$.

Our aim in this section is to look for interesting properties shared by these regular lasso languages, ultimately seeking out those properties which are *equivalent* to regularity.

We will often define sets of lassos by specifying their sets of spokes and loops. For this reason, we introduce the following conveniant shorthand.

Definition 5.1. Let $X, Y \subseteq \Sigma^*$ be sets of finite strings, where $\epsilon \notin Y$. We let

$$\text{Lassos}(X, Y) := \{\, (x, y) \mid x \in X, y \in Y \,\},$$

saying that $\text{Lassos}(X, Y)$ is the set (or language) of lassos *generated by* X and Y.

5.1 Characterization using finitary regular languages

Our first characterization theorem is an analogue of Theorem 2.18 of the preliminaries in which ω-regular languages were defined in terms of finitary regular languages. To establish a similar result for regular languages of lassos, we first prove an analogue of Proposition 2.14. For this, let $\mathbb{A}$ be a Büchi automaton, (x, y) be a lasso, and consider a (x, y)-labeled trace τ through $\mathbb{A}$. Such a trace is accepting if and only if

- the initial state of (x, y) to some initial state q_I of $\mathbb{A}$,
- the knot point of (x, y) to some state q of $\mathbb{A}$,
- some point of the loop of (x, y) to some accepting state q_F.

In other words, $(x, y) \in \text{Label}_{\text{fin}}(\mathbb{A})$ if and only if

$$q_I \overset{x}{\twoheadrightarrow} q \overset{y_1}{\twoheadrightarrow} q_F \overset{y_2}{\twoheadrightarrow} q$$

for some $q_I \in I_{\mathbb{A}}$, $q \in Q_{\mathbb{A}}$, $q_F \in F_{\mathbb{A}}$, and $y_1, y_2 \in \Sigma^*$ such that $y_1 y_2 = y$. Reformulated, we have the following.

Proposition 5.2. Let $\mathbb{A}$ be a Büchi automaton. Then

$$\text{Label}_{\text{fin}}(\mathbb{A}) = \bigcup_{q_I \in I_{\mathbb{A}}} \bigcup_{q \in \mathbb{A}} \bigcup_{q_F \in F_{\mathbb{A}}} \text{Lassos}\left([q_I \twoheadrightarrow q], ([q \twoheadrightarrow q_F] \circ [q_F \twoheadrightarrow q]) \setminus \{\epsilon\}\right)$$

As a reminder, Theorem 2.18 states that a language $\mathcal{L} \subseteq \Sigma^\omega$ is ω-regular if and only if

$$\mathcal{L} = \bigcup_{i<n} X_i \circ Y_i^\omega$$

for some finite set of regular languages $\{X_0, Y_0, \ldots, X_{n-1}, Y_{n-1}\}$. We will prove something similar, but the result will be a bit more complicated. This is in part due to the fact that $X \circ Y^\omega = X \circ Y^* \circ Y^\omega$, and so, very losely speaking,

the separation of the "loop-part" Y and the "spoke-part" X isn't as clear cut. What we get instead is the following.

Theorem 5.3. A bisimulation closed set of lassos $\mathcal{L}$ is regular if and only if

$$\mathcal{L} = \bigcup_{i<n} \text{Lassos}(X_i Y_i^*, Y_i^+)$$

for some finite set $\{X_0, Y_0, X_1, Y_1, \ldots, X_{n-1}, Y_{n-1}\}$ of regular languages (over Σ^*) such that Y_i does not contain the empty string, ϵ, for any $i < n$.

The proof from right to left will use the following lemma.

Lemma 5.4. For every pair (X, Y) of regular languages $X, Y \subseteq \Sigma^*$ such that $\epsilon \notin Y$, there is an automaton $\mathbb{A}$ such that $\mathcal{L}_{\text{fin}}(\mathbb{A})$ is the closure of $\text{Lassos}(XY^*, Y^+)$ under bisimilarity, restricted to the set of lassos.

This in turn will be proven using two well-known results about regular languages over Σ^*. The same sort of construction can for instance be found in the proof of Theorem 5.4 in [15], page 28. They are proved here for the sake of completeness.

The first of the lemmas gives us an automaton of a special shape for the "spoke"-part of our lasso language.

Definition 5.5. We call a state q in an automaton $\mathbb{A}$ a *sink* if there are no outgoing transitions from q, i.e., $q \xrightarrow{a} q'$ does *not* hold for any $a \in \Sigma$, $q' \in \mathbb{A}$.

Lemma 5.6. For every regular language $X \subseteq \Sigma^*$, there is an automaton $\mathbb{A}_s$ which accepts X, such that $\mathbb{A}_s$ has a single accepting state q_F which is a sink.

Proof. Since X is a regular language, there is some automaton

$$\mathbb{A} = (Q_{\mathbb{A}}, \Delta_{\mathbb{A}}, I_{\mathbb{A}}, F_{\mathbb{A}})$$

which accepts X. We construct our automaton

$$\mathbb{A}_s = (Q_{\mathbb{A}_s}, \Delta_{\mathbb{A}_s}, I_{\mathbb{A}_s}, F_{\mathbb{A}_s})$$

by adding a new point r to $\mathbb{A}$, setting it as the single accepting point and adding transitions to it from states which used to see an accepting point. More formally, we let

$$\begin{aligned} Q_{\mathbb{A}_s} &= Q_{\mathbb{A}} \uplus \{r\} \\ I_{\mathbb{A}_s} &= \begin{cases} I_{\mathbb{A}} & \text{if } I_{\mathbb{A}} \cap F_{\mathbb{A}} = \emptyset, \text{ i.e., } \epsilon \notin \mathcal{L}(\mathbb{A}) \\ I_{\mathbb{A}} \cup \{r\} & \text{otherwise.} \end{cases} \\ F_{\mathbb{A}_s} &= \{r\} \end{aligned}$$

and define the transition function by setting

$$q \xrightarrow{a} q' \text{ in } \mathbb{A}_s \iff \begin{cases} q \xrightarrow{a} q' \text{ in } \mathbb{A}, \text{ or} \\ q' = r \text{ and } \exists q_F \in F_{\mathbb{A}} \text{ s.t. } q \xrightarrow{a} q_F \text{ in } \mathbb{A} \end{cases}$$

Now, it is clear that by construction, $\mathbb{A}$ accepts the empty word if and only if $\mathbb{A}_s$ accepts it. For the case of a non-empty word x we have that, for an arbitrary $q \in Q_{\mathbb{A}}$, $q \xrightarrow{x} q_F$ for some $q_F \in F_{\mathbb{A}}$ if and only if $q \xrightarrow{x} r$ in $\mathbb{A}_s$. Hence, $\mathcal{L}(\mathbb{A}) = \mathcal{L}(\mathbb{A}_s)$ as required. □

The second lemma gives us an automaton for the "loop"-part.

Lemma 5.7. If $Y \subseteq \Sigma^*$ is a regular language such that $Y = Y^*$, then there is an automaton $\mathbb{A}_l$ which accepts Y such that $\mathbb{A}_l$ has a single initial and accepting state which coincide, i.e., $I_{\mathbb{A}_l} = F_{\mathbb{A}_l} = \{q_F\}$ for some $q_F \in \mathbb{A}_l$.

Proof. The proof is similar to the previous one. We take an existing automaton $\mathbb{A}$ with initial states $I_{\mathbb{A}}$ and accepting states $F_{\mathbb{A}}$, add a new point r which will be the new single initial and accepting state. Again, we add transitions to r from those states of $\mathbb{A}$ which see an accepting point, but we also add transitions *from* r to those states of $\mathbb{A}$ which are *seen* by an initial point. Formally, we let $\mathbb{A}_l$ be an automaton defined by

$$\begin{aligned} Q_{\mathbb{A}_l} &= Q_{\mathbb{A}} \uplus \{r\} \\ I_{\mathbb{A}_l} &= \{r\} \\ F_{\mathbb{A}_l} &= \{r\} \end{aligned}$$

and we define the transition function by letting

$$q \xrightarrow{a} q' \text{ in } \mathbb{A}_l \iff \begin{cases} q \xrightarrow{a} q' \text{ in } \mathbb{A}, \text{ or} \\ q' = r \text{ and } \exists q_F \in F_{\mathbb{A}} \text{ s.t. } q \xrightarrow{a} q_F \text{ in } \mathbb{A}, \text{ or} \\ q = r \text{ and } \exists q_I \in I_{\mathbb{A}} \text{ s.t. } q_I \xrightarrow{a} q' \text{ in } \mathbb{A}, \text{ or} \\ q = q' = r \text{ and } \exists q_I \in I_{\mathbb{A}}, q_F \in F_{\mathbb{A}} \text{ s.t. } q_I \xrightarrow{a} q_F \text{ in } \mathbb{A}. \end{cases}$$

It is easy to see that $\mathcal{L}(\mathbb{A}) \subseteq \mathcal{L}(\mathbb{A}_l)$, since any accepting path

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \dots \xrightarrow{a_{n-2}} q_{n-1} \xrightarrow{a_{n-1}} q_n$$

in $\mathbb{A}$ yields an accepting path

$$r \xrightarrow{a_0} q_1 \xrightarrow{a_1} \dots \xrightarrow{a_{n-2}} q_{n-1} \xrightarrow{a_{n-1}} r$$

in $\mathbb{A}_l$. For the other direction, note that for every path in $\mathbb{A}_l$ which is of the form

$$r \xrightarrow{a_0} q_1 \xrightarrow{a_1} \dots \xrightarrow{a_{n-2}} q_{n-1} \xrightarrow{a_{n-1}} r \tag{5}$$

where all of the states $q_1, \dots, q_{n-1}$ are also in $\mathbb{A}$ (i.e., are distinct from r), there are $q_I \in I_{\mathbb{A}}$ and $q_F \in F_{\mathbb{A}}$ such that

$$q_I \xrightarrow{a_0} q_1 \xrightarrow{a_1} \dots \xrightarrow{a_{n-2}} q_{n-1} \xrightarrow{a_{n-1}} q_F$$

is a path in $\mathbb{A}$. Given the requirement that $\mathcal{L}(\mathbb{A}) = Y = Y^*$, the result now follows from the observation that any accepting path through $\mathbb{A}_l$ must either be the empty path or be a concatenation of paths of the form in (5). □

Proof of Lemma 5.4. Let $X, Y \subseteq \Sigma^*$ be regular languages where $\epsilon \notin Y$ and let $\mathcal{L}$ be the closure of $\text{Lassos}(XY^*, Y^+)$ under bisimilarity, restricted to the set of lassos. Our aim is to build an automaton $\mathbb{A}$ such that $\mathcal{L}_{\text{fin}}(\mathbb{A}) = \mathcal{L}$.

According to Lemma 5.6 there is an automaton $\mathbb{A}_s$ accepting X with a single accepting state q_F which is a sink. Similarily, Lemma 5.7 provides us with an automaton $\mathbb{A}_s$ with a single initial and accepting state q'_F. We now combine these two automata by first letting $\mathbb{A}_s$ and $\mathbb{A}_l$ have disjoint states *except* that

we let $q_F = q'_F$, thereby *identifying* the accepting state of $\mathbb{A}_s$ with the initial and accepting state of $\mathbb{A}_l$. Next, we let $\mathbb{A}$ be the automaton defined by

$$\begin{aligned} Q_{\mathbb{A}} &= Q_{\mathbb{A}_s} \cup Q_{\mathbb{A}_l} \\ \Delta_{\mathbb{A}} &= \Delta_{\mathbb{A}_s} \cup \Delta_{\mathbb{A}_l} \\ I_{\mathbb{A}} &= I_{\mathbb{A}_s} \\ F_{\mathbb{A}} &= \{q_F\}. \end{aligned}$$

We now show that $\mathcal{L} = \mathcal{L}_{\text{fin}}(\mathbb{A})$ by showing that

(a) $\text{Lassos}(XY^*, Y^+) \subseteq \text{Label}_{\text{fin}}(\mathbb{A})$; and

(b) for every lasso in $\text{Label}_{\text{fin}}(\mathbb{A})$ there is a bisimilar one in $\text{Lassos}(XY^*, Y^+)$.

This suffices, since then the bisimulation closure of $\text{Lassos}(XY^*, Y^+)$ is equal to the bisimulation closure of $\text{Label}_{\text{fin}}(\mathbb{A})$ (restricted to the set of lassos). But the former is precisely $\mathcal{L}$ and the latter is precisely $\mathcal{L}_{\text{fin}}(\mathbb{A})$.

(a) Consider a lasso $(xy, y') \in \text{Lassos}(XY^*, Y^+)$ such that $x \in X, y \in Y^*$ and $y' \in Y^+$. By the construction of our automaton $\mathbb{A}$, x is the label of a finite path from an initial state q_I to q_F, y is the label of a finite path from q_F to q_F and y' is the label of a non-empty finite path from q_F to q_F. Hence $q_I \overset{xy}{\twoheadrightarrow} q_F \overset{y'}{\twoheadrightarrow} q_F$. But this means that (xy, y') is the label of an accepting trace through $\mathbb{A}$, i.e., $(xy, y') \in \text{Label}_{\text{fin}}(\mathbb{A})$.

(b) Let (w, z) be a lasso in $\text{Label}_{\text{fin}}(\mathbb{A})$. Our aim is to construct a bisimilar lasso (xy, y') such that $x \in X, y \in Y^*$ and $y' \in Y^+$.

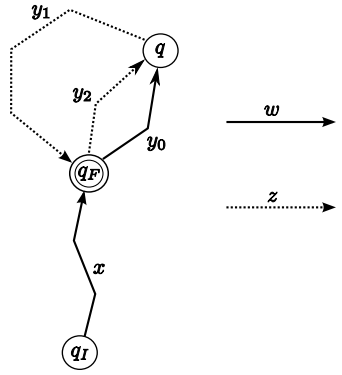


Figure 11: Deconstructing a lasso $(w, z) \in \text{Label}_{\text{fin}}(\mathbb{A})$

Keeping Figure 11 in mind, we argue as follows. Since (w, z) is the label of an accepting path through $\mathbb{A}$, we have that w is the label of a finite path from some initial state q_I to some state q and z is the label of a non-empty finite path from q back to itself which passes through the accepting state q_F. In other words, there must be $y_1, y_2 \in \Sigma^*$ such that $z = y_1 y_2$ and

$$q_I \overset{w}{\twoheadrightarrow} q \overset{y_1}{\twoheadrightarrow} q_F \overset{y_2}{\twoheadrightarrow} q.$$

Our first observation is that all the states in the finite path

$$q \overset{y_1}{\twoheadrightarrow} q_F \overset{y_2}{\twoheadrightarrow} q$$

must be in the "loop-part" of the automaton $\mathbb{A}$, i.e., they must be elements in $Q_{\mathbb{A}_l}$. This is because there are no outgoing transitions from q_F "back" into $Q_{\mathbb{A}_s} \setminus \{q_F\}$. In particular, q is in the loop-part, and therefore, the finite path $q_I \overset{w}{\twoheadrightarrow} q$ must pass through the accepting state q_F, which means that there are $x, y_0 \in \Sigma^*$ such that $x \in X$ and $w = xy_0$. Note that y_0 can be the empty string if $q = q_F$ and the entire path $q_I \overset{w}{\twoheadrightarrow} q$ is in the "spoke-part" of the automaton (i.e., in $Q_{\mathbb{A}_s}$).

If we let $y = y_0y_1$ and $y' = y_1y_2$, we therefore have that $(w, z) = (xy_0, y_1y_2)$ is bisimilar to $(xy_0y_1, y_2y_1) = (xy, y')$. What remains is to show that $y \in Y^*$ and $y' \in Y^+$. However, since $z = y_1y_2$ is the loop of the lasso (z, w), it cannot be the empty string which means that $y' = y_2y_1$ is not the empty string either. Therefore, it suffices to show that both y and y' are in Y^*. But this is follows from the fact that

$$y, y' \in [q_F \twoheadrightarrow q_F]_{\mathbb{A}} = [q_F \twoheadrightarrow q_F]_{\mathbb{A}_l} = Y^*.$$

□

Proof of Theorem 5.3. $\Rightarrow$) Assume that $\mathcal{L}$ is a regular language of lassos, and let $\mathbb{A}$ be a prophetic Büchi automaton accepting $\mathcal{L}$. Combining Corollary 4.21, which states that the set $\mathrm{Label}(\mathbb{A})$ is closed under bisimilarity, with Propositon 5.2, we get that

$$\mathcal{L}_{\mathrm{fin}}(\mathbb{A}) = \mathrm{Label}_{\mathrm{fin}}(\mathbb{A}) = \bigcup_{\substack{q_I \in I_{\mathbb{A}} \\ q \in Q_{\mathbb{A}} \\ q_F \in F_{\mathbb{A}}}} \mathrm{Lassos}\left([q_I \twoheadrightarrow q], ([q \twoheadrightarrow q_F] \circ [q_F \twoheadrightarrow q]) \setminus \{\epsilon\}\right).$$

Fixing a triplet $(q_I, q, q_F) \in I_{\mathbb{A}} \times Q_{\mathbb{A}} \times F_{\mathbb{A}}$ we let

$$\begin{aligned} X &:= [q_I \twoheadrightarrow q] \\ Y &:= ([q \twoheadrightarrow q_F] \circ [q_F \twoheadrightarrow q]) \setminus \{\epsilon\}. \end{aligned}$$

Then X and Y are both regular languages and Y does not contain the empty string. Furthermore, it is not difficult to see that $Y = Y^+$ since the concatenation of two paths from q to q which passes through q_F must itself be a path from q to q which passes through q_F. Similarily, we have $X = X \circ Y^*$, since the concatenation of a path $q_I \twoheadrightarrow q$ with a path $q \twoheadrightarrow q_F \twoheadrightarrow q$ is a path from q_I to q. Therefore, $\mathrm{Lassos}(X, Y) = \mathrm{Lassos}(XY^*, Y^+)$ and the result follows from the simple observation that there are only finitely many triplets (q_I, q, q_F) in $I_{\mathbb{A}} \times Q_{\mathbb{A}} \times F_{\mathbb{A}}$.

$\Leftarrow$) Assume that $\mathcal{L}$ is a lasso language which is closed under bisimilarity (restricted to the set of lassos), and that there are regular languages $\{X_0, Y_0, X_1, Y_1, \ldots, X_{n-1}, Y_{n-1}\}$, where $\epsilon \notin Y_i$ for any $i < n$, such that

$$\mathcal{L} = \bigcup_{i<n} \mathrm{Lassos}(X_iY_i^*, Y_i^+).$$

Our aim is to construct an automaton $\mathbb{A}$ such that $\mathcal{L}_{\mathrm{fin}}(\mathbb{A}) = \mathcal{L}$. We will do this by taking the disjoint union of automata obtained by applying Lemma 5.4. Formally, if $\mathbb{A}$ and $\mathbb{B}$ are two Büchi automata, then $\mathbb{C} = \mathbb{A} \uplus \mathbb{B}$ is defined by letting $Q_{\mathbb{C}} = Q_{\mathbb{A}} \uplus Q_{\mathbb{B}}$, $\Delta_{\mathbb{C}} = \Delta_{\mathbb{A}} \uplus \Delta_{\mathbb{B}}$, $I_{\mathbb{C}} = I_{\mathbb{A}} \uplus I_{\mathbb{B}}$, and $F_{\mathbb{C}} = F_{\mathbb{A}} \uplus F_{\mathbb{B}}$.

Now, according to Lemma 5.4, for each pair X_i, Y_i, there is an automaton $\mathbb{A}_i$ which accepts the bisimilarity closure of $\text{Lassos}(X_i Y_i^*, Y^+)$. So, we let $\mathbb{A} = \biguplus_{i<n} \mathbb{A}_i$ and proceed to show that this automaton will do.

For this, take a lasso $\mathbb{S} \in \mathcal{L}_{\text{fin}}(\mathbb{A})$. By definition there is an $i < n$ such that $\mathbb{S} \in \mathcal{L}_{\text{fin}}(\mathbb{A}_i)$. But then $\mathbb{S}$ is bisimilar to some $\mathbb{T} \in \text{Lassos}(X_i Y_i^*, Y_i^+)$. Since $\mathcal{L} = \bigcup_{i<n} \text{Lassos}(X_i Y_i^*, Y_i^+)$, $\mathbb{T} \in \mathcal{L}$; and since $\mathcal{L}$ is closed under bisimilarity, $\mathbb{S} \in \mathcal{L}$.

For the other direction, assume that $\mathbb{S} \in \mathcal{L} = \bigcup_{i<n} \text{Lassos}(X_i Y_i^*, Y_i^+)$. Then there is some $i < n$ such that $\mathbb{S} \in \text{Lassos}(X_i Y_i^*, Y_i^+)$. But then $\mathbb{S} \in \mathcal{L}_{\text{fin}}(\mathbb{A}_i)$, and therefore $\mathbb{S} \in \mathcal{L}_{\text{fin}}(\mathbb{A})$. □

Notice how the proof above made use of the fact that prophetic Büchi automata have the property that the set of accepting labels is equal to the set of accepted sourced flows. In the next characterization theorem, we will be working with deterministic parity automata, for which we have no result which equates the set of accepting labels with the set of accepted sourced flows. What we instead obtain is a result characterizing regular lasso languages in terms of the k-blockpumping property.

5.2 Characterization using the k-blockremoval property

Definition 5.8. Let $\mathcal{L}$ be a lasso language. For an arbitrary $x \in \Sigma^*$, we let

$$\text{Loop}_x(\mathcal{L}) := \{\, y \mid (x, y) \in \mathcal{L} \}.$$

We say that $\mathcal{L}$ has the *relativized loop concatenation property* (RLCP), if for every $x \in \Sigma^*$, $\text{Loop}_x(\mathcal{L}) = \text{Loop}_x(\mathcal{L})^+$.

As an example, if X and Y are any sets of finite strings, where Y doesn't contain the empty string, then $\text{Lassos}(X, Y^+)$ has the loop concatenation property. To see why, pick an arbitrary $x \in \Sigma^*$ and let (x, y) and (x, y') be two lassos in $\text{Lassos}(X, Y^+)$. Then $x \in X$ and $y, y' \in Y^+$. Since Y^+ is closed under concatenation, we have $yy' \in Y^+$ as well, and so $(x, yy') \in Y^+$.

Note, however that we do *not* in general have that regular lasso languages have the RLCP. As a simple example let

$$\mathcal{L} := \{(\epsilon, a), (\epsilon, aa), (\epsilon, aaa), \dots\} \cup \{(\epsilon, b), (\epsilon, bb), (\epsilon, bbb), \dots\}.$$

Then $\mathcal{L}$ is a regular lasso language, but $\text{Loop}_\epsilon(\mathcal{L})$ is not closed under concatenation.

Definition 5.9. If $x, y \in \Sigma^*$ are such that $\text{Loop}_x(\mathcal{L}) = \text{Loop}_y(\mathcal{L})$, then we call x and y *loop equivalent*. This is clearly an equivalence relation and we write $[x]$ for the equivalence class to which x belongs, i.e., the set of all y such that x and y are loop equivalent (relative to $\mathcal{L}$, of course).

Definition 5.10. We say that a language $\mathcal{L}$ of lassos has the *loop unwinding property* (LUP), if

$$y \in \text{Loop}_x(\mathcal{L}) \implies [x] = [xy],$$

for all $x, y \in \Sigma^*$.

Having the LUP is a quite strong property and can be seen as a sort of "right congruence" property of the equivalence classes $[x]$. It is therefore perhaps not so surprising that it is linked to determinism, as we shall see in the proof of Theorem 5.11 below.

Theorem 5.11. A lasso language $\mathcal{L}$ is regular if and only if there is some $k \in \omega$ and some language of lassos $\mathcal{L}'$ such that

(a) $\mathcal{L}$ is the bisimulation closure of $\mathcal{L}'$,

(b) $\mathcal{L}'$ has the RLCP

(c) $\mathcal{L}'$ has the LUP

(d) for all $x \in \Sigma^*$, both $\mathrm{Loop}_x(\mathcal{L}')$ and $[x]$ have the k-blockremoval property.

Proof. $\Rightarrow$). Let $\mathcal{L}$ be a regular lasso language and let $\mathbb{A}$ be a deterministic parity automaton with parity map $p\colon Q_{\mathbb{A}} \to \omega$ such that $\mathcal{L}_{\mathrm{fin}}(\mathbb{A}) = \mathcal{L}$. We will make heavy use of the fact that $\mathbb{A}$ is deterministic, amongst others in the following sense: For $x \in \Sigma^*$, we inductively define the state q_x inductively by letting q_ϵ be the initial state of $\mathbb{A}$, and in the induction step letting q_{xa} be the unique state such that $q_x \xrightarrow{a} q_{xa}$.

We will now show that $\mathcal{L}' = \mathrm{Label}_{\mathrm{fin}}(\mathbb{A})$ has the required properties. Recall that $\mathrm{Label}_{\mathrm{fin}}(\mathbb{A})$ are the set of lassos (x, y) such that there exists an accepting trace $\tau\colon (x, y) \to \mathbb{A}$. For parity automata, this means that

$$q_I \overset{x}{\twoheadrightarrow} q \overset{y}{\twoheadrightarrow} q,$$

and that in the (unique) finite path $q \overset{y}{\twoheadrightarrow} q$, the highest parity encountered is even.

Our first observation is that for any $x \in \Sigma^*$, the sets $[x]$ and $\mathrm{Loop}_x(\mathcal{L}')$ are uniquely determined by the state q_x. More precisely, if x and y are finite strings such that $q_x = q_y$, then it is not difficult to see that $\mathrm{Loop}_x(\mathcal{L}') = \mathrm{Loop}_y(\mathcal{L}')$, which also means that $[x] = [y]$.

(a) We know that $\mathcal{L} = \mathcal{L}_{\mathrm{fin}}(\mathbb{A})$ is the bisimulation closure of $\mathcal{L}' = \mathrm{Label}_{\mathrm{fin}}(\mathbb{A})$.

(b) Take an arbitrary $x \in \Sigma^*$ and consider two lassos (x, y) and (x, y') in $\mathrm{Label}_{\mathrm{fin}}(\mathbb{A})$. By the definition of $\mathrm{Label}_{\mathrm{fin}}(\mathbb{A})$, we have that

$$q_I \overset{x}{\twoheadrightarrow} q_x \overset{y}{\twoheadrightarrow} q_x \text{ and } q_I \overset{x}{\twoheadrightarrow} q_x \overset{y'}{\twoheadrightarrow} q_x.$$

Moreover, the (unique) paths $q_x \overset{y}{\twoheadrightarrow} q_x$ and $q_x \overset{y'}{\twoheadrightarrow} q_x$ are both such that the highest parity encountered on the path is even. But then concatenating these two paths yields a path from q_x back to itself which has the property that the highest parity encountered is even. Hence $(x, yy') \in \mathrm{Label}_{\mathrm{fin}}(\mathbb{A})$, which shows that $\mathcal{L}' = \mathrm{Label}_{\mathrm{fin}}(\mathbb{A})$ has the RLCP.

(c) Let $x, y \in \Sigma^*$ be such that $y \in \mathrm{Loop}_x(\mathcal{L}')$. This means that there is a unique y-labeled finite path from q_x back to itself. But then $q_x = q_{xy}$ and so $[x] = [xy]$.

(d) Now, our task is to find a $k \in \omega$ which works for every $x \in \Sigma^*$, in the sense that for each $x \in \Sigma^*$, both $\mathrm{Loop}_x(\mathcal{L}')$ and $[x]$ has the k-blockremoval property. We do this by proving that

(i) There are only finitely many different sets $\mathrm{Loop}_x(\mathcal{L}')$ and $[x]$, i.e., both $\{\, \mathrm{Loop}_x(\mathcal{L}') \mid x \in \Sigma^* \,\}$ and $\{\, [x] \mid x \in \Sigma^* \,\}$ are finite;

(ii) for each $x \in \Sigma^*$, there is a $k \in \omega$ such that $\mathrm{Loop}_x(\mathcal{L}')$ has the k-blockremoval property; and

(iii) for each $x \in \Sigma^*$, there is a $k \in \omega$ such that $[x]$ has the k-blockremoval property.

This suffices because then we can take k to be the maximum of those found in (ii) and (iii). Furthermore, by Theorem 2.2, in order to show (ii) and (iii), it is sufficient to prove that both $\mathrm{Loop}_x(\mathcal{L}')$ and $[x]$ are regular languages, for each choice of $x \in \Sigma^*$.

(i) As mentioned, the sets $\mathrm{Loop}_x(\mathbb{A})$ are determined by the *states* q_x. Since there are only finitely many states in $\mathbb{A}$, this implies that $\{\, \mathrm{Loop}_x(\mathbb{A}) \mid x \in \Sigma^* \,\}$ is finite. This, in turn, implies that the set $\{\, [x] \mid x \in \Sigma^* \,\}$ is finite as well, since two strings x and x' belong to the same equivalence class iff $\mathrm{Loop}_x(\mathbb{A}) = \mathrm{Loop}_{x'}(\mathbb{A})$.

(ii) Let $x \in \Sigma^*$ be an arbitrary finite word. Since $\mathbb{A}$ is deterministic, we know that for every $y \in \Sigma^*$, there is a unique y-labeled finite path starting in q_x. We now set $f(y)$ to be the highest parity encountered in this finite path. Formally, this can be expressed as

$$f(y) = \max\{\, p(q_{xy'}) \mid y' \text{ a substring of } y \,\}.$$

Using this definition, we have that $\mathrm{Loop}_x(\mathcal{L}')$, which by definition is equal to $\{\, y \mid (x, y) \in \mathrm{Label}_{\mathrm{fin}}(\mathbb{A}) \,\}$, consists of precisely those non-empty strings y such that $q_x \overset{y}{\twoheadrightarrow} q_x$ and $f(y)$ is even. Now, $[q_x \twoheadrightarrow q_x] \setminus \{\epsilon\}$ is a regular language and since regular languages are closed under intersection, it therefore suffices to show that the set $\{\, y \mid f(y) \text{ is even} \,\}$ is regular.

Now, since $\mathbb{A}$ is finite, there is a highest possible parity, say n. This means that

$$\{\, y \mid f(y) \text{ is even} \,\} = \bigcup_{\{i \mid 2i \leq n\}} \{\, y \mid f(y) = 2i \,\}.$$

Since the class of regular languages is closed under finite union, it therefore suffices to show that $L_i := \{\, y \mid f(y) = i \,\}$ is regular for each $i \leq n$.

Now, a string y is in L_i if and only if y is the label of a path from q_x which passes through a state with parity i, *without passing through any state with higher parity.* In other words, y is an element of L_i if and only if $y = y_1 y_2$, for some strings y_1 and y_2 such that q_{xy_1} has parity i *and* y is *not* an element of L_j for any $j > i$. This can be summarized in the following equation.

$$L_i = \left(\bigcup_{q \in \mathbb{A}, p(q) = i} \{\, y_1 y_2 \mid q_x \overset{y_1}{\twoheadrightarrow} q \,\} \right) \setminus \bigcup_{i < j \leq n} L_j.$$

This implies that L_n is regular, since the equation then reduces to

$$\begin{aligned} L_n &= \bigcup_{q \in \mathbb{A}, p(q)=n} \{\, y_1 y_2 \mid q_x \overset{y_1}{\twoheadrightarrow} q \,\} \\ &= \bigcup_{q \in \mathbb{A}, p(q)=n} [q_x \twoheadrightarrow q] \circ \Sigma^*, \end{aligned}$$

which is regular since the class of regular languages is closed under finite unions and composition.

An inductive argument now shows that all the L_i's are regular, for $i \leq n$.

(iii) That $[x]$ is regular for every $x \in \Sigma^*$ is somewhat simpler. A string $x' \in \Sigma^*$ is an element of $[x]$ if and only if $\mathrm{Loop}_{x'}(\mathcal{L}') = \mathrm{Loop}_x(\mathcal{L}')$. We already noted that $\mathrm{Loop}_x(\mathcal{L}')$ is uniquely determined by the state q_x of the automaton. Making this more notationally precise, we have that if we write

$$\mathrm{Loop}(q) := \{\, y \mid (\epsilon, y) \in \mathrm{Label}_{\mathrm{fin}}(q) \,\}$$

then

$$\mathrm{Loop}_x(\mathcal{L}') = \mathrm{Loop}(q_x).$$

Hence

$$\begin{aligned} [x] &= \{\, x' \mid \mathrm{Loop}_{x'}(\mathcal{L}') = \mathrm{Loop}_x(\mathcal{L}') \,\} \\ &= \{\, x' \mid \mathrm{Loop}(q_{x'}) = \mathrm{Loop}(q_x) \,\} \\ &= \bigcup_{\mathrm{Loop}(q)=\mathrm{Loop}(q_x)} [q_\epsilon \twoheadrightarrow q], \end{aligned}$$

which is a finite union of regular sets, and is therefore regular itself.

$\Leftarrow$). Assume that we have a set $\mathcal{L}'$ and a k with the required properties. Now, since $\mathrm{Loop}_x(\mathcal{L}')$ and $[x]$ both have the k-blockremoval property, we know that they are regular. However, we also know that there are only finitely many languages with the k-blockremoval property, which implies that

$$\{\, [x] \mid x \in \Sigma^* \,\} = \{X_0, X_1, \ldots, X_{n-1}\}$$

for some finite set of regular languages $\{X_0, X_1, \ldots, X_{n-1}\}$. Furthermore, since each $X_i = [x]$ for some x, there is some associated set

$$Y_i := \mathrm{Loop}_x(\mathcal{L}').$$

We now show that

$$\mathcal{L}' = \bigcup_{i<n} \mathrm{Lassos}(X_i, Y_i).$$

For this, take a lasso $(x, y) \in \mathcal{L}'$. There is some $i < n$ such that $[x] = X_i$. By definition, we have $y \in \mathrm{Loop}_x(\mathcal{L}')$, which means that $y \in Y_i$. But this means that $(x, y) \in \mathrm{Lassos}(X_i, Y_i)$. For the other direction, assume that a lasso (x, y) is an element of $\mathrm{Lassos}(X_i, Y_i)$ for some $i < n$. This is equivalent to saying that $[x] = X_i$ and $y \in \mathrm{Loop}_x(\mathcal{L}')$, which implies that $(x, y) \in \mathcal{L}'$.

Now, we know that $\mathcal{L}'$ has the RLCP, which means that for every x, $\mathrm{Loop}_x(\mathcal{L}') = \mathrm{Loop}_x(\mathcal{L}')^+$. In other words, we have $Y_i = Y_i^+$ for every $i < n$. Furthermore, we know that $\mathcal{L}'$ has the LUP, which means that if $y \in \mathrm{Loop}_x(\mathcal{L}')$, then $[x] = [xy]$. In other words, if $x \in X_i$ and $y \in Y_i$, then $xy \in X_i$, for every $i < n$. But this means that $X_i = X_i \circ Y_i^*$, for every $i < n$. These two observations taken together gives us that

$$\mathcal{L}' = \bigcup_{i<n} \mathrm{Lassos}(X_i Y_i^*, Y_i^+)$$

where, for each $i < n$, X_i, Y_i are regular languages over Σ^*, and Y_i doesn't contain the empty string (since Y_i consists of loops, which are always non-empty). The proof is now nearly identical to the last part of the proof of Theorem 5.3. □

5.3 Characterization using a finite congruence

In this section we show that not only does a prophetic automaton partition the set of all lassos, but this partition is a special sort of congruence. In fact, we can turn this into a characterization theorem for regular lasso languages. This is comparable to the situation for regular languages over finite words, where it is well known that a set X of finite strings is regular if and only if X is an equivalence class of a congruence of finite index.

The theorem can also be applied as a characterization result for ω-regular languages. As such, it is a natural, and arguably simple, consequence of the result by Carton and Michel (our Theorem 2.21). For this reason, it is quite possibly a well known result, but we were unable to find a reference.

Definition 5.12. Let $\mathbb{S}$ be a sourced flow and $x \in \Sigma^*$ a finite word. We write $x\mathbb{S}$ for the result of "prepending" $\mathbb{S}$ with x. Formally, if $\mathbb{S}$ is an infinite word u, then $x\mathbb{S}$ is xu, and if $\mathbb{S}$ is a lasso (y, z), then $x\mathbb{S} = (xy, z)$.

Definition 5.13. Let $\mathbb{S} = (\mathcal{S}_s, s)$ be a sourced flow and $x \in \Sigma^*$ a finite word. If $s \overset{x}{\twoheadrightarrow} t$ in $\mathbb{S}$ for some t, we say that $\mathbb{S}$ *has* an x-derivative, and that the sourced flow $\mathbb{S}_x := (\mathcal{S}_t, t)$ is the x-derivative of $\mathbb{S}$.

Note that, for all $x \in \Sigma$, $(x\mathbb{S})_x = \mathbb{S}$. However, even if $\mathbb{S}$ has an x-derivative, we do not in general have that $x(\mathbb{S}_x) = \mathbb{S}$. For instance, if $\mathbb{S}$ is the lasso (ϵ, xy), for some $xy \in \Sigma^*$, then $\mathbb{S}_x = (\epsilon, yx)$ and hence $x(\mathbb{S}_x) = (x, yx)$. We do, however, have the following.

Lemma 5.14. If $\mathbb{S}$ and $\mathbb{T}$ are sourced flows such that $\mathbb{S}_x = \mathbb{T}$ for some $x \in \Sigma^*$, then $\mathbb{S}$ is bisimilar to $x\mathbb{T}$.

Proof. The result follows from the observation that $\overrightarrow{\mathbb{S}} = x\overrightarrow{\mathbb{T}} = \overrightarrow{x\mathbb{T}}$. □

We now lift these notions to the sets of sourced flows.

Definition 5.15. Let $\mathcal{L}$ be a set of sourced flows and $x \in \Sigma^*$ be a finite word. We let

$$x\mathcal{L} := \{\, x\mathbb{S} \mid \mathbb{S} \in \mathcal{L} \,\}$$

and

$$\mathcal{L}_x := \{\, \mathbb{S}_x \mid \mathbb{S} \in \mathcal{L},\ \mathbb{S} \text{ has an } x\text{-derivative} \,\}$$

By these definitions we have that $(x\mathcal{L})_x = \mathcal{L}$ for all $\mathcal{L} \subseteq \mathsf{SFlow}$ and $x \in \Sigma^*$; but not in general that $x(\mathcal{L}_x) = \mathcal{L}$. Consider, for instance, when $\mathcal{L}$ is non-empty but none of its elements are x-derivable. Then $\mathcal{L}_x = \emptyset$, and hence $x(\mathcal{L}_x) = \emptyset \neq \mathcal{L}$.

Definition 5.16. Let $\sim$ be an equivalence relation on the set of all sourced flows, and write $[\mathbb{S}]$ for the equivalence class to which an arbitrary sourced flow $\mathbb{S}$ belongs. We say that $\sim$ is a *left congruence* if

$$x[\mathbb{S}] \subseteq [x\mathbb{S}]$$

for all sourced flows $\mathbb{S}, \mathbb{S}'$ and finite words $x \in \Sigma^*$.

It is not difficult to see that $\sim$ is a left congruence if and only if $\mathbb{S} \sim \mathbb{S}' \implies x\mathbb{S} \sim x\mathbb{S}'$ for all $\mathbb{S}, \mathbb{S}' \in \mathsf{SFlow}$, $x \in \Sigma^*$, which explains the term "left congruence".

Now, if $\mathbb{A}$ is a prophetic automaton, we know that the sets $\{\mathcal{L}(q)\}_{q \in \mathbb{A}}$ form a partition of SFlow. We write $\sim_{\mathbb{A}}$ for the equivalence relation corresponding to this partition, and for an arbitrary $\mathbb{S} \in \mathsf{SFlow}$, we write $[\mathbb{S}]_{\mathbb{A}}$ for the equivalence class to which $\mathbb{S}$ belongs. The fact that it is a left congruence follows easily from the following proposition.

Proposition 5.17. Let $\mathbb{A}$ be a prophetic automaton. Then

$$q \overset{x}{\twoheadrightarrow} q' \iff \mathcal{L}(q) \supseteq x\mathcal{L}(q')$$

for any $q, q' \in \mathbb{A}$, $x \in \Sigma^*$.

Proof. If $q \overset{x}{\twoheadrightarrow} q'$ and $\mathbb{S} \in \mathcal{L}(q')$, then certainly $x\mathbb{S} \in \mathcal{L}(q)$, since prepending the accepting $\mathbb{S}$-labeled trace starting in q' with the finite x-labeled path from q to q' yields an accepting $x\mathbb{S}$-labeled trace starting in q.

For the other direction, assume $\mathcal{L}(q) \supseteq x\mathcal{L}(q')$, and take an arbitrary $\mathbb{S} \in \mathcal{L}(q')$. Then, by assumption, $x\mathbb{S} \in \mathcal{L}(q)$. However, this means that $q \overset{x}{\twoheadrightarrow} q''$ for some q'' such that $\mathbb{S} \in \mathrm{Label}(q'')$. Since $\mathbb{A}$ is prophetic, this implies $q'' = q'$. □

We get the following as a corollary. Here we use the standard terminology that an equivalence relation is said to be of finite index if it only has finitely many equivalence classes.

Corollary 5.18. If $\mathbb{A}$ is prophetic then $\sim_{\mathbb{A}}$ is a left congruence of finite index.

Proof. We have already established that $\sim_{\mathbb{A}}$ is an equivalence relation of finite index, so we only need to show that for arbitrary $x \in \Sigma^*$, $\mathbb{S} \in \mathsf{SFlow}$, $x[\mathbb{S}]_{\mathbb{A}} \subseteq [x\mathbb{S}]_{\mathbb{A}}$. Now, let q be the unique state such that $\mathcal{L}(q) = [x\mathbb{S}]_{\mathbb{A}}$ and let q' be the unique state such that $\mathcal{L}(q') = [\mathbb{S}]_{\mathbb{A}}$. Since $x\mathbb{S} \in \mathcal{L}(q)$, there must be *some* state q'' such that $\mathbb{S} \in \mathcal{L}(q'')$ and $q \overset{x}{\twoheadrightarrow} q''$. By uniqueness, we must have $q'' = q'$. Hence $q \overset{x}{\twoheadrightarrow} q'$, and the result follows from Proposition 5.17 above. □

Our final aim is to take a (special kind of) left congruence $\sim$ of finite index over SFlow and construct a Büchi automaton $\mathbb{A}_\sim$ whose states are the equivalence classes $\mathcal{L}$ of $\sim$ and where the transitions are such that $\mathcal{L} \xrightarrow{a} \mathcal{L}'$ if and only if $\mathcal{L} \supseteq a\mathcal{L}'$. The question is, which equivalence classes will act as accepting points? So, we look at the accepting points of a prophetic automaton to see if the languages they define have some defining properties.

Definition 5.19. If $\mathcal{L} \subseteq \mathsf{SFlow}$ and $x \in \Sigma^*$ are such that $\mathcal{L} \subseteq \mathcal{L}_x$, we call x a *postfixpoint* of $\mathcal{L}$. We write $\mathrm{Postfix}(\mathcal{L})$ for the set of all postfixpoints of $\mathcal{L}$.

Since we will deal with languages which are closed under bisimilarity, the following lemma gives us an alternate definition of postfixpoints.

Lemma 5.20. If $\mathcal{L}$ is a language of sourced flows closed under bisimilarity, then x is a postfixpoint of $\mathcal{L}$ if and only if $x\mathcal{L} \subseteq \mathcal{L}$.

Proof. Let $\mathcal{L}$ be a language of flows which is closed under bisimilarity. For the direction from left to right, assume that x is a postfixpoint of $\mathcal{L}$ and that $\mathbb{S} \in \mathcal{L}$. Since $\mathcal{L} \subseteq \mathcal{L}_x$, there must be some $\mathbb{T} \in \mathcal{L}$ such that $\mathbb{T}_x = \mathbb{S}$. But then $\mathbb{T}$ is bisimilar to $x\mathbb{S}$, and since $\mathcal{L}$ is closed under bisimilarity, we have $x\mathbb{S} \in \mathcal{L}$. This shows that $x\mathcal{L} \subseteq \mathcal{L}$.

For the direction from right to left, assume that $x\mathcal{L} \subseteq \mathcal{L}$. Then, if $\mathbb{S} \in \mathcal{L}$, $x\mathbb{S} \in \mathcal{L}$. But since $(x\mathbb{S})_x = \mathbb{S}$, we have that $\mathbb{S} \in \mathcal{L}_x$. This shows that $\mathcal{L} \subseteq \mathcal{L}_x$, so x is a postfixpoint. □

Now, consider a Büchi automaton $\mathbb{A}$ and a state $q \in \mathbb{A}$ such that $q \xrightarrow{x} q$ for some $x \in \Sigma^*$. If $\tau\colon \mathbb{S} \to \mathbb{A}$ is a successful trace which starts in q, then "prepending" this trace with the finite x-labeled path from q to q, yields an $x\mathbb{S}$-labeled successful trace starting in q. It is not difficult to see that this implies that if $\mathbb{S} \in \mathcal{L}(q)$, then $x\mathbb{S} \in \mathcal{L}(q)$, for all sourced flow $\mathbb{S}$. But this means that x is a postfixpoint of $\mathcal{L}(q)$ and so we get the following proposition.

Lemma 5.21. If $\mathbb{A}$ is a Büchi automaton, then

$$q \xrightarrow{x} q \implies x \in \mathrm{Postfix}(\mathcal{L}(\mathrm{q})),$$

for every $x \in \Sigma^*$, $q \in \mathbb{A}$.

We would like for the converse to hold as well, but, unfortunately, we we cannot in general draw the conclusion that $q \xrightarrow{x} q$ from the fact that x is a postfixpoint of $\mathcal{L}(q)$. Consider, for instance, the automaton of Figure 12. Here a is a postfixpoint of $\mathcal{L}(q_0)$, since if $\mathbb{S} \in \mathcal{L}(q_0)$, then certainly $a\mathbb{S} \in \mathcal{L}(q_0)$ as well. But there is no a-labeled path from q_0 to q_0.

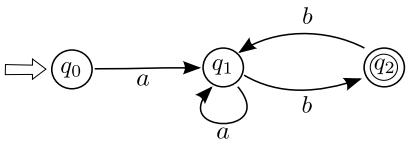


Figure 12: Not all postfixpoints yield loops

For prophetic automata, the converse *does* hold.

Lemma 5.22. If $\mathbb{A}$ be a prophetic Büchi automaton, then the following are equivalent for every finite word $x \in \Sigma^*$ and state $q \in \mathbb{A}$:

(a) $q \overset{x}{\twoheadrightarrow} q$

(b) $x \in \text{Postfix}(\mathcal{L}(q))$

(c) $x \in \text{Postfix}(\mathcal{L}_{\text{fin}}(q))$.

Proof. (a) $\Rightarrow$ (b) is Lemma 5.21.

(b) $\Rightarrow$ (c) Assume that $x \in \Sigma^*$ is a postfixpoint of $\mathcal{L}(q)$, we wish to show that it is also a postfixpoint of $\mathcal{L}_{\text{fin}}(q)$. However, this follows from the following chain of implications.

$$\begin{aligned} \mathbb{S} \in \mathcal{L}_{\text{fin}}(q) &\implies \mathbb{S} \in \mathcal{L}(q) \\ &\implies x\mathbb{S} \in \mathcal{L}(q) \text{ (since } x \text{ is a postfixpoint of } \mathcal{L}(q)) \\ &\implies x\mathbb{S} \in \mathcal{L}_{\text{fin}}(q) \text{ (since } x\mathbb{S} \text{ is finite whenever } \mathbb{S} \text{ is).} \end{aligned}$$

(c) $\Rightarrow$ (a) Let $\mathcal{L} = \mathcal{L}_{\text{fin}}(q)$ and assume that x is a postfixpoint of $\mathcal{L}$. By Lemma 5.20, we have $\mathcal{L} \subseteq x\mathcal{L}$, which, using Proposition 5.17, shows that $q \overset{x}{\twoheadrightarrow} q$. □

If we consider an accepting state q in a prophetic Büchi automaton, we get that every finite x-labeled path from q back to itself passes through an accepting state, namely q itself. By Lemma 5.22 above, we know that every such path is witnessed by a postfixpoint of $\mathcal{L}(q)$. Hence, we obtain that whenever x is a postfixpoint of $\mathcal{L}(q)$, then the lasso $(\epsilon, x) \in \mathcal{L}(q)$. We can also concatenate such finite paths, from which we get that if $x_0, x_1, \ldots$ are all postfixpoints of q, then the infinite word $x_0x_1\ldots$ is an element of $\mathcal{L}(q)$.

Definition 5.23. We say that a language $\mathcal{L}$ of sourced flows is *postfixpoint-closed* if

1. $x \in \text{Postfix}(\mathcal{L}) \implies (\epsilon, x) \in \mathcal{L}$ for every $x \in \Sigma^*$, and
2. $u \in \text{Postfix}(\mathcal{L})^\omega \implies u \in \mathcal{L}$ for every $u \in \Sigma^\omega$.

We say that a language $\mathcal{L}$ of lassos is *finitely postfixpoint-closed* if it satisfies condition 1.

The discussion preceding Definition 5.23 showed that the language accepted by an accepting state of a prophetic Büchi automaton is postfixpoint-closed. We now show the converse of this, namely that if an element q of a prophetic automaton $\mathbb{A}$ is postfixpoint-closed, then it is accepting. This requires that $\mathbb{A}$ has a maximal set of accepting states, in the sense that the set cannot be enlarged without changing the languages defined by points in the automaton.

Definition 5.24. Let $\mathbb{A}$ be a Büchi automaton. We say that the set $F_{\mathbb{A}}$ of accepting states is maximal if and only if whenever $\mathbb{B} = (Q_{\mathbb{A}}, \Delta_{\mathbb{A}}, I_{\mathbb{A}}, F')$ for some $F' \supsetneq F_{\mathbb{A}}$, then there is some $q \in Q_{\mathbb{A}}$ such that $\mathcal{L}(q, \mathbb{A}) \neq \mathcal{L}(q, \mathbb{B})$.

Lemma 5.25. Let $\mathbb{A}$ be a prophetic Büchi automaton with a maximal set of accepting states. Then the following are equivalent for any state q:

(a) q is accepting.

(b) $\mathcal{L}(q)$ is postfixpoint-closed.

(c) $\mathcal{L}_{\text{fin}}(q)$ is finitely postfixpoint-closed.

Proof. (a) $\Rightarrow$ (b) Let q be an accepting state and let $x_0, x_1, x_2, \ldots$ be a sequence of fixpoints of $\mathcal{L}(q)$. By Lemma 5.22, we have that $q \overset{x_i}{\twoheadrightarrow} q$ for all $i \in \omega$. But then concatenating these finite paths yields an accepting $x_0x_1x_2\ldots$-labeled trace through $\mathbb{A}$, which shows that $x_0x_1x_2\ldots \in \mathcal{L}(\mathbb{A})$.

(b) $\Rightarrow$ (c) This is Lemma 5.21.

(c) $\Rightarrow$ (a) Assume that $\mathcal{L}_{\text{fin}}(q)$ is finitely postfixpoint-closed. To show that q is accepting, we show that for any point q_0 in $\mathbb{A}$, and any infinite trace $\tau\colon u \to \mathbb{A}$ (where u is some infinite word) which starts in q_0 and where $q \in \inf(\tau)$, there exists a (possibly different) successful u-labeled trace starting in q_0. This is sufficient, since then q can be made accepting without changing the language accepted by q_0, and since q_0 was arbitrarily chosen, the maximality of $F_{\mathbb{A}}$ implies that q is accepting.

So, pick a point $q_0 \in \mathbb{A}$, and let $\tau\colon u \to \mathbb{A}$ be some trace starting in q_0 such that $q \in \inf(\tau)$. Then we must have that

$$q_0 \overset{x}{\twoheadrightarrow} q \overset{y_0}{\twoheadrightarrow} q \overset{y_1}{\twoheadrightarrow} q \overset{y_2}{\twoheadrightarrow} \ldots$$

where $xy_0y_1y_2\ldots = u$. By Lemma 5.22, y_i is a postfixpoint of $\mathcal{L}_{\text{fin}}(q)$ for every $i \in \omega$. Since $\mathcal{L}_{\text{fin}}(q)$ is finitely postfixpoint-closed by assumption, we must have $(\epsilon, y_i) \in \mathcal{L}_{\text{fin}}(q)$ for every $i \in \omega$. Because $\mathbb{A}$ is prophetic, this implies that $(\epsilon, y_i) \in \text{Label}_{\text{fin}}(q)$ for each $i \in \omega$.

Now, that $(\epsilon, y_i) \in \text{Label}_{\text{fin}}(q)$, means that there is a trace of the form

$$q \overset{\epsilon}{\twoheadrightarrow} q \overset{y_i}{\twoheadrightarrow} q,$$

where the path $q \overset{y_i}{\twoheadrightarrow} q$ passes through some accepting state. But then by concatenating these finite paths, we obtain a trace τ of the form

$$q \overset{y_0}{\twoheadrightarrow} q \overset{y_1}{\twoheadrightarrow} q \overset{y_2}{\twoheadrightarrow} \cdots$$

which is successful and starts in q. Since $q_0 \overset{x}{\twoheadrightarrow} q$, prepending this finite path to τ, we get a successful $u = xy_0y_1y_2\ldots$-labeled trace starting in q_0. □

We are now suited to state and prove our main theorem of this subsection. We state it simultaneously in its "full" and "restricted" version, the first being a characterization theorem for regular languages of sourced flows, and the second characterizing regular *lasso languages*.

Theorem 5.26. A language $\mathcal{L}$ of sourced flows (lassos) is regular (is a regular lasso language) if and only if there is a left congruence of finite index $\sim$ on SFlow (Lasso) such that

(a) each equivalence class $[\mathbb{S}]$ of $\sim$ is closed under bisimilarity (in the finitary case, restricted to the set of lassos),

(b) $\mathcal{L} = [\mathbb{S}_0] \cup [\mathbb{S}_1] \cup \ldots \cup [\mathbb{S}_{n-1}]$ for some finite set $\{\mathbb{S}_0, \mathbb{S}_1, \ldots, \mathbb{S}_{n-1}\}$ of sourced flows (lassos),

(c) for every lasso $\mathbb{S}$, there is an x-derivative $\mathbb{S}_x$, where $x \neq \epsilon$, such that $[\mathbb{S}_x]$ is postfixpoint-closed (finitarily postfixpoint-closed).

We prove only the finitary version of the theorem. The proof can easily be adapted to the full case.

Proof. $\Rightarrow$) Assume that $\mathcal{L}$ is a regular lasso language. Then there is some prophetic automaton $\mathbb{A}$ such that $\mathcal{L}_{\text{fin}}(\mathbb{A}) = \mathcal{L}$, and we can safely assume that its set of final states is maximal. We let the equivalence classes of our wanted relation $\sim$ be the languages $\mathcal{L}_{\text{fin}}(q)$, for $q \in \mathbb{A}$. Then (a) and (b) are immediate. To see why (c) holds, consider any lasso $\mathbb{S}$. Since $\mathbb{A}$ is prophetic, there is some state q such that $\mathbb{S} \in \text{Label}_{\text{fin}}(q)$. In other words, there is some successful trace $\tau \colon \mathbb{S} \to \mathbb{A}$ which starts in q. This trace must visit some accepting point q_F and from this it is not difficult to see that there must be some x such that $[\mathbb{S}_x] = \mathcal{L}(q_F)$. By Lemma 5.25, $\mathcal{L}(q_F)$ must be postfixpoint-closed, so we are done.

$\Leftarrow$) Assume that $\sim$ is a left congruence of finite index satisfying conditions (a), (b) and (c). We define a Büchi automaton $\mathbb{A}$ by letting

$$\begin{aligned} Q_{\mathbb{A}} &= \{\, [\mathbb{S}] \mid \mathbb{S} \in \mathsf{Lasso} \,\} \\ I_{\mathbb{A}} &= \{\, [\mathbb{S}_0], [\mathbb{S}_1], \ldots, [\mathbb{S}_{n-1}] \,\} \\ F_{\mathbb{A}} &= \{\, [\mathbb{S}] \mid \mathbb{S} \in \mathsf{Lasso}, [\mathbb{S}] \text{ is finitely postfixpoint-closed} \,\} \end{aligned}$$

and where the transition function is defined by

$$[\mathbb{S}] \xrightarrow{a} [\mathbb{T}] \iff [\mathbb{S}] = [a\mathbb{T}].$$

This is well defined, since $\sim$ is a left congruence and hence $\mathbb{T} \sim \mathbb{T}' \implies a\mathbb{T} \sim a\mathbb{T}'$ for all $\mathbb{S}, \mathbb{T} \in \mathsf{Lasso}$, $a \in \Sigma$. It is not difficult to see that it also means that

$$[\mathbb{S}] \xrightarrow{x} [\mathbb{T}] \iff [\mathbb{S}] = [x\mathbb{T}]$$

for all $x \in \Sigma^*$.

We now show that for every state $\mathcal{A} \in \mathbb{A}$ and every lasso $\mathbb{S}$

$$\mathbb{S} \in \mathcal{L}_{\text{fin}}(\mathcal{A}) \iff \mathbb{S} \in \mathcal{A},$$

from which the result follows.

For the direction from left to right, assume that $\mathbb{S}$ is an element of $\mathcal{L}_{\text{fin}}(\mathcal{A})$. Then there is some successful finite run ρ of $\mathbb{A}$ on $\mathbb{S}$ which starts in $\mathcal{A}$. If we let (x, y) be the label of the run ρ, we know that there is some state $\mathcal{B}$ and some *accepting* state $\mathcal{C}$ such that

$$\mathcal{A} \xrightarrow{x} \mathcal{B} \xrightarrow{y_1} \mathcal{C} \xrightarrow{y_2} \mathcal{B}$$

for some $y_1, y_2 \in \Sigma^*$ such that $y = y_1 y_2$. Then

$$\mathcal{A} \xrightarrow{xy_1} \mathcal{C} \xrightarrow{y_2 y_1} \mathcal{C}$$

holds as well. Now, take a lasso $\mathbb{S} \in \mathcal{C}$. Then we have that $\mathcal{C} = [\mathbb{S}]$. Since $\mathcal{C} \xrightarrow{y_2 y_1} \mathcal{C}$, our transition function states that $[\mathbb{S}] = [y_2 y_1 \mathbb{S}]$. But this means that $y_2 y_1 \mathbb{S} \in \mathcal{C}$, showing that $y_2 y_1 \mathcal{C} \subseteq \mathcal{C}$. By Lemma 5.20, this implies that

y_2y_1 is a postfixpoint of $\mathcal{C}$. Since $\mathcal{C}$ is an accepting point, it is per definition finitely postfixpoint-closed which means that (ϵ, y_2y_1) is an element of $\mathcal{C}$ and so $\mathcal{C} = [(\epsilon, y_2y_1)]$. However, since we have $\mathcal{A} \overset{xy_1}{\twoheadrightarrow} \mathcal{C}$, we must, according to the definition of our transition function, have $\mathcal{A} = [xy_1(\epsilon, y_2y_1)] = [(xy_1, y_2y_1)]$. But (xy_1, y_2y_1) is bisimilar to (x, y_1y_2), so by condition (a), we have $(x, y_1y_2) \in \mathcal{A}$, as required.

For the right to left direction, assume that $\mathbb{S}$ is a lasso in $\mathcal{A}$, i.e., $\mathcal{A} = [\mathbb{S}]$. We will construct a successful $\overrightarrow{\mathbb{S}}$-labeled trace through $\mathbb{A}$ starting in A, which suffices according to Corollary 4.8.

By property (c), there is some $x_0 \in \Sigma^+$ such that $[\mathbb{S}_{x_0}]$ is finitely postfixpoint-closed. Again, by property (c), there is some $x_1 \in \Sigma^+$ such that $[(\mathbb{S}_{x_0})_{x_1}] = [\mathbb{S}_{x_0x_1}]$ is finitely postfixpoint-closed. Continuing in this fashion, we obtain a sequence of finitely postfixpoint-closed states $[\mathbb{S}_{x_0}], [\mathbb{S}_{x_0x_1}], \ldots$

Now, for all lassos $\mathbb{T}$ and all x such that $\mathbb{T}_x$ exists, we have that $x(\mathbb{T}_x)$ is bisimilar to $\mathbb{T}$ and so $[\mathbb{T}] = [x\mathbb{T}_x]$. Hence, according to our transition function, we have that for all x such that $\mathbb{T}_x$ exists, $[\mathbb{T}] = [x\mathbb{T}_x] \overset{x}{\twoheadrightarrow} [\mathbb{T}_x]$. In particular then, we have that

$$A = [\mathbb{S}] \overset{x_0}{\twoheadrightarrow} [\mathbb{S}_{x_0}] \overset{x_1}{\twoheadrightarrow} [\mathbb{S}_{x_0x_1}] \overset{x_2}{\twoheadrightarrow} \ldots \tag{6}$$

It is not difficult to see that if a lasso $\mathbb{T}$ is derivable by some string x, then $\overrightarrow{\mathbb{T}} = xu$ for some $u \in \Sigma^*$. Now, since our strings $x_0, x_1, \ldots$ are all non-empty, and $\mathbb{S}$ is derivable by $x_0, x_0x_1, x_0x_1x_2, \ldots$, this means that $\overrightarrow{\mathbb{S}} = x_0x_1x_2\ldots$ Hence, any trace of the form (6) defines a successful $\overrightarrow{\mathbb{S}}$-labeled trace starting in A. □

Borrowing inspiration from the theory of automata operating on finite words, we call the automaton defined in the left to right direction of the proof of Theorem 5.26 the *syntactic automaton generated by* $\sim$. It is not difficult to see that the automaton produced is *prophetic* and therefore *co-deterministic*, as opposed to the deterministic automaton one obtains when constructing the syntactic automaton for a congruence over finite words.

6 Conclusions

The main objective of this thesis was to investigate automata operating on sourced flows, and in particular investigate properties of the *finitary fragments of regular languages of flows*. The rationale for this investigation is that there is a translation between automata and the modal μ-calculus which can be lifted to a translation between coalgebraic automata and coalgebraic fixed point logic (see [22]). Therefore, the regular languages from the automata perspective correspond to the definable classes of models in the logic perspective. In particular, the finitary fragments of regular languages correspond to the *definable classes of finite models*. In that sense, this thesis has been an investigation in the *finite model theory* of the coalgebraic fixed point logic, for one particular class of coalgebras.

The most important conceptual idea is the development of a *run* of an automaton on a sourced flow, as the definition gives us a purely finitary description of the finitary fragment of regular languages. This definition also showed that given an automaton $\mathbb{A}$, there are some accepted sourced flows which are more "representative" than others, namely those which are *labels of accepting traces* through $\mathbb{A}$. This subset turns out to be important, since a trace τ gives us information about the similarities in structure between the sourced flows and the automaton. In particular, it gives information about *loops* in the automaton. We then showed that for a particular class of automata, namely *prophetic Büchi automata*, the set of accepting labels is *identical* to the language accepted. Hence, prophetic automata behave very nicely in this coalgebraic perspective.

Finally, we gave three characterization results for regular lasso languages, being the finitary fragments of regular languages of sourced flows. Among these three, the arguably most enlightening one is the third (Theorem 5.26), where we showed that regularity is equivalent to being the union of equivalence classes of a special type of finite congruence.

One of the initial aims was to find a characterization result using some sort of "pumping". In that respect, the second result (Theorem 5.11) can be seen as a partial success. However, we would have wanted to develop pumping in a more "coalgebraic" way, in order for the result to be generalizable to a more general setting.

6.1 Future Research

We believe that the syntactic quality of prophetic automata as it applies to the coalgebraic setting is something which is worth further investigation. As an example, Theorem 4.22 shows a link between the *final* coalgebra Σ^ω, and the structure of prophetic automata; a result which can possibly yield important insights.

Moreover, we hope that the ideas from this thesis can be generalized to other classes of coalgebras. It would be interesting to see if, for instance, our definition of a *run* could be applied in a more general setting.

Finally, we think that there *is* a possibility of developing a good notion of pumping in order to yield a characterization result. We already have some ideas for this, which may be developed in a later paper.

A Coalgebra

Given the coalgebraic perspective in this thesis, we here provide a very short list of definitions to let the reader familiarize with the notions involved. We happily refer the reader to the works of Rutten [18] and the joint work by Jacobs & Rutten [7], especially for the theory of systems. For the connections between coalgebra and modal logic, we recommend Kurz [11], Pattinson [14] and Venema [23].

Definition A.1. Let F be an endofunctor over some category C. An *F-coalgebra*, also called a *coalgebra with signature F*, is a pair $\mathbb{A} = (A, \alpha)$, where A is some object in C and α is an arrow from A to FA.

Definition A.2. Let $\mathbb{A} = (A, \alpha)$ and $\mathbb{B} = (B, \beta)$ be two F-coalgebras for some endofunctor F. An arrow $h\colon A \to B$ is called a *homomorphism* if $\beta \circ h = \alpha \circ Fh$, i.e., if the following diagram commutes:

$$\begin{array}{ccc} A & \xrightarrow{h} & B \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ FA & \xrightarrow[Fh]{} & FB \end{array}$$

Definition A.3. With F an endofunctor, we write $\mathrm{Coalg}(F)$ for the category of all F-coalgebras, with homomorphisms acting as arrows. It is easy to check that the requirements for being a category is satisfied.

This perspective enables us to ask general category theoretic questions, such as whether a certain endofunctor F admits a final coalgebra, i.e., whether the category $\mathrm{Coalg}(F)$ has a final object.

Definition A.4. A coalgebra $\mathbb{Z} = (Z, \zeta)$ with signature F is said to be *final* in the category of all F-coalgebras if for every F-coalgebra (A, α), there is a unique homomorphism $!_{\mathbb{S}}\colon \mathbb{S} \to \mathbb{Z}$.

Now, in the following, we will assume that F is an endofunctor over the base category Set. Using the terminology of Rutten in [18], coalgebras over such functors F are called *systems* or *transition structures*. In this case, we can speak of *states* of an F-coalgebra (A, α), a state simply being an element of A. Many of the familiar notions of Kripke frames or labeled transition systems easily generalize into this setting.

Definition A.5. Let (A, α) be a system over some Set endofunctor F and let $B \subseteq A$ be a subset of A where $i\colon B \to A$ is the inclusion mapping. We say that B is a *subsystem* of $\mathbb{A}$ if there is a function $\beta\colon B \to FB$ such that $i\colon B \to A$ is a homomorphism from (B, β) to (A, α).

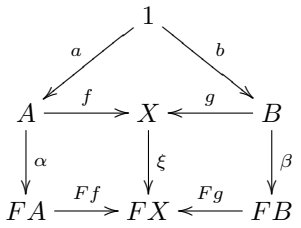
Definition A.6. Let $\mathbb{A} = (A, \alpha)$ be a system and let $a \in A$ be a state. The subsystem of $\mathbb{A}$ generated by a, written $\mathbb{A}_a$, is the smallest (in terms of $\subseteq$) subsystem of $\mathbb{A}$ which contains a.

Definition A.7. A system $\mathbb{A}$ is called *minimal* if it has no proper subsystems, i.e., the only subsystem of $\mathbb{A}$ is $\mathbb{A}$ itself.

As an example, in the category of flows defined in section 3, the minimal flows are precisely the lassos where the spoke is empty, i.e., the lassos consisting only of a loop.

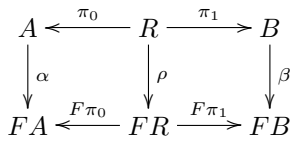
Now, thinking of systems as mathematical models of state-based dynamic systems, we can see homomorphisms as some sort of *behavior* preserving functions, i.e., if f is a homomorphism from $\mathbb{A}$ to $\mathbb{B}$ and a is some state in a, we intuitively regard a and $f(a)$ as having the same behaviour. To turn this semi-intuitive notion into an equivalence relation over states of systems, one arrives, after a bit of work, at the following definition:

Definition A.8. Let $\mathbb{A} = (A, \alpha)$ and $\mathbb{B} = (B, \beta)$ be two systems over some Set endofunctor F. We say that two points $a \in A$ and $b \in B$ are *behaviorally equivalent*, written $\mathbb{A}, a \equiv \mathbb{B}, b$, if there is some F-coalgebra $\mathbb{X} = (X, \xi)$ and two homomorphisms $f \colon \mathbb{A} \to \mathbb{X}$ and $g \colon \mathbb{B} \to \mathbb{X}$ such that $f(a) = g(b)$, i.e., such that the following diagram commutes:



A related notion, and one which is well-known from both modal logic and computer science, is that of a bisimulation relation between two structures. This can be generalized to systems in the following way:

Definition A.9. Let F be an endofunctor over set and let (A, α) and (B, β) be two F-systems. A relation $R \subseteq A \times B$ is said to be a *bisimulation* if it can be endowed with a coalgebraic structure, i.e., if there is a function $\rho \colon R \to FR$ such that the projection functions $\pi_0 \colon R \to A$ and $\pi_1 \colon R \to B$ are homomorphisms:



Fixing two points $a \in \mathbb{A}$ and $b \in \mathbb{B}$, we say that a and b are bisimilar, written $\mathbb{A}, a \leftrightarrows \mathbb{B}, b$, if there exists a bisimulation $R \subseteq A \times B$ such that $(a, b) \in R$.

Fact A.10. Bisimilarity implies behavioural equivalence, i.e.,

$$\mathbb{A}, a \leftrightarrows \mathbb{B}, b \implies \mathbb{A}, a \equiv \mathbb{B}, b$$

The reverse implication does not in general hold, see for instance the work on monotone neighborhood frames by Hansen & Kupke [6]. However, the two notions *do* coincide for a large class of functors.

Fact A.11. If F is a Set endofunctor which preserves weak pullbacks, then behavioral equivalence implies bisimililarity, i.e.,

$$\mathbb{A}, a \equiv \mathbb{B}, b \implies \mathbb{A}, a \leftrightarrows \mathbb{B}, b$$

Since Kripke polynomial functors preserve weak pullbacks, we get that for the two coalgebraic structures that are present in this thesis, viz., labeled transition systems and flows, behavioral equivalence and bisimilarity coincide.

References

[1] Peter Aczel. *Non-Well-Founded Sets*. Number 14 in CSLI Lecture Notes. Center for the Study of Language and Information, Stanford, 1988.

[2] Julius Büchi. On a decision method in restricted second-order arithmetic. In *Prooceedings of the International Congress on Logic, Method, and Philosophy of Sciences*, pages 1–12. Stanford University Press, 1962.

[3] Andrzej Ehrenfeucht, Rohit Parikh, and Grzegorz Rozenberg. Pumping lemmas for regular sets. *SIAM Journal on Computing*, 10(3):536–541, 1981.

[4] Berndt Farwer. ω-automata. In Grädel et al. [5], pages 3–20.

[5] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[6] Helle Hansen and Clemens Kupke. A coalgebraic perspective on monotone modal logic. *Electronic Notes in Theoretical Computer Science*, 106:121–143, 2004.

[7] Bart Jacobs and Jan Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of the European Association for Theoretical Computer Science*, 12, 1997. Also available at `http://citeseer.ist.psu.edu/jacobs97tutorial.html`.

[8] David Janin and Igor Walukiewicz. On the expressive completeness of the propositional μ-calculus with respect to monadic second order logic. In *CONCUR '96: Concurrency Theory, 7th International Conference*, volume 1119, pages 263–277. Springer-Verlag, 1996. Also available at `http://citeseer.ist.psu.edu/janin96expressive.html`.

[9] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[10] Clemens Kupke and Yde Venema. Closure properties of coalgebra automata. In *LICS '05: Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)*, pages 199–208, Washington, DC, USA, 2005. IEEE Computer Society.

[11] Alexander Kurz. Coalgebras and modal logic. In *Proceedings of Advances in Modal Logic*, pages 222–230, 1998. Also available at `http://citeseer.ist.psu.edu/article/kurz98coalgebras.html`.

[12] Max Michel and Olivier Carton. Unambiguous büchi automata. *Theoretical Computer Science*, 297:37–81, 2003.

[13] Lawrence Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96:277–317, 1999.

[14] Dirk Pattinson. An introduction to the theory of coalgebras, 2003. Available at `http://www.pst.ifi.lmu.de/~pattinso/Publications/nasslli.all.ps.gz`.

[15] Dominique Perrin and Jean Éric Pin. *Infinite Words*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004. ISBN 0-12-532111-2.

[16] Michael Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[17] Markus Roggenbach. Determinization of büchi-automata. In Grädel et al. [5], pages 43–60.

[18] Jan Rutten. Universal coalgebra: A theory of systems. Technical report, CWI, 1996. Available at `http://citeseer.ist.psu.edu/301731.html`.

[19] Wolfgang Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science (vol. B): formal models and semantics*, pages 133–191. MIT Press, Cambridge, MA, USA, 1990.

[20] Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Languages, vol. 3: Beyond Words*, pages 389–455. Springer-Verlag New York, Inc., New York, NY, USA, 1997. Also available at `http://citeseer.ist.psu.edu/thomas96language.html`.

[21] Johan van Benthem. *Modal Logic and Classical Logic*. Bibliopolis, Napoli, 1983.

[22] Yde Venema. Automata and fixed point logics for coalgebras. *Electronic Notes in Computer Science*, 106:335–375, 2004.

[23] Yde Venema. Algebras and coalgebras. In Johan van Benthem, Patrick Blackburn, and Frank Wolter, editors, *Handbook of Modal Logic*, pages 331–426. Elsevier, Amsterdam, 2006.

[24] Thomas Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bulletin of the Belgian Mathematical Society*, 8(2):359–391, 2001. Also available at `http://citeseer.ist.psu.edu/wilke00alternating.html`.

Index