Coalgebraic Representation of Nondeterministic Systems and Büchi Automata

Midterm Report Research Internship

Jorrit de Boer

Abstract. *We provide an explanation of existing literature for describing Büchi automata coalgebraically. To do this we also explain modal mu-calculus and a coalgebraic model of nondeterministic systems. Finally, we outline our plans for the rest of the research internship.*

1 Introduction

Büchi automata and *nondeterministic systems* are crucial in theoretical computer science for modeling and verifying systems with infinite behaviors [3, 8]. Nondeterministic systems capture uncertainty and multiple outcomes, and are used in models like concurrent processes and nondeterministic Turing machines. Büchi automata, which are often also nondeterministic, handle infinite sequences of events, crucial for verifying systems that run indefinitely, such as operating systems or network protocols.

Coalgebra provides an effective framework for modeling state-based, dynamic systems. Techniques such as *coinduction* allow for reasoning about infinite structures, while *bisimulation* offers a formal way to establish behavioral equivalence between systems [6]. By modeling Büchi automata coalgebraically, we unify these powerful tools for reasoning about infinite behaviors and nondeterminism.

The main goal of this report is to provide an understanding of the coalgebraic construction of Büchi automata described in [7]. To do so we also explain *modal mu-calculus*, a system to verify properties of transition systems, and provide a coalgebraic model of nondeterministic systems, upon which the construction for the Büchi automata builds. By outlining these concepts we advance our first goal of the research internship, which is to gain an understanding of the current research into this topic.

Next to providing an overview of the available literature, we outline our plan for the rest of the internship. Our goal is to use *game semantics* as an alternative framework to derive the coalgebraic representation of Büchi automata. Game semantics is a framework of describing a system in terms of a two-player game between a *verifier* and a *refuter* who want to verify, respectively refute, a statement [3]. By utilizing game semantics we hope to provide a more intuitive proof of the existing results.

The document is outlined as follows. In Section 2 we provide some background and relevant definitions for the rest of the report. In Section 3 we give the main results from the studied literature, which is divided into: modal mu-calculus in Section 3.1, coalgebraic model of nondeterministic systems in Section 3.2, and the coalgebraic model of Büchi automata in Section 3.3. Finally, in Section 4 we summarize the results and give our plans for the rest of the internship.

2 Background

2.1 Büchi Automata

Let us consider a very simple motivating example for a Büchi automaton, shown in Figure 1.

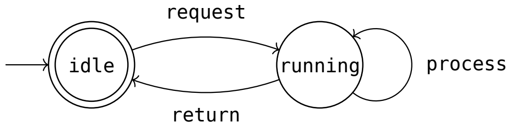

Figure 1: Example of a Büchi automaton.

This system represents some machine that takes requests, processes them, and returns some result. One might want to verify that this machine does not get stuck. In terms of the system shown, this would mean that the machine always ends up in the `idle` state again.

This behavior can be modeled using a Büchi automaton. A Büchi automaton, namely, is an automaton which models infinite behavior, and accepts those words for which there is path through the automaton where the transitions are labeled by the letters of the words, an there is an accepting state that the path moves through infinitely many times. In this example, we make the `idle` state accepting, so the automaton accepts those words that always take the `return` transition again, and thus do not process indefinitely.

We can now give a formal definition of a Büchi automaton, and its *accepted language*:

Definition 2.1: A (nondeterministic) Büchi Automaton [3] is a tuple $\chi = \langle X, \Sigma, \delta, s, F \rangle$, with X a set of states, Σ the alphabet, $s \subseteq X$ the set of initial states, $\delta : X \times \Sigma \to \mathcal{P}(X)$ the transition function, $F \subseteq X$ the set of *final* (or *accepting*) states.

A run of a Büchi Automaton χ is $(x_0, \sigma_0)(x_1, \sigma_1)... \in (X \times \Sigma)^\omega$, such that for every $n \in \omega$, $x_{n+1} \in \delta(x_n, \sigma_n)$. We denote the set of runs of the Büchi Automaton χ as Run_χ.

A run is accepted if it passes through an accepting state infinitely many times. I.e. a run $\rho = (x_0, \sigma_0)(x_1, \sigma_1)...$ is accepted if $\{x_i \mid x_i \in F\}$ is an infinite set. AccRun_χ is the set of accepted runs of χ. $\mathrm{Run}_{\chi,X'}$ is the set of runs starting in X', i.e. $(x_0, \sigma_0)(x_1, \sigma_1)... \in \mathrm{Run}_{\chi,X}$ if $x_0 \in X'$. The set $\mathrm{AccRun}_{\chi,X'}$ is defined the same way. The map $\mathrm{DelSt} : \mathrm{Run}_\chi \to \Sigma^\omega$ 'deletes' the states from a run and returns an ω-word. So $\mathrm{DelSt}((x_0, \sigma_0)(x_1, \sigma_1)...) = \sigma_0\sigma_1....$ Naturally, the set of accepted language of a Büchi Automaton can then be defined as $\mathrm{Lang}(\chi) = \mathrm{DelSt}(\mathrm{AccRun}_{\chi,s})$.

Indeed we now see that the accepted language for the example automaton is $(\texttt{request} \cdot \texttt{process}^* \cdot \texttt{return})^\omega$. I.e. the machine gets a request, processes for at most some *finite* number of transitions and then returns some result. It does not get stuck processing indefinitely.

2.2 Fixed Points

Crucial for the next section, Section 3.1 about modal mu-calculus, is reasoning about *fixed points* of *monotone* functions. We briefly recall the important definitions and theorems.

Definition 2.2: A *complete lattice* is a partially ordered set $\langle L, \leq \rangle$ such that every subset $M \subseteq L$ has a least upper bound $\bigvee M$ and greatest lower bound $\bigwedge M$. Specifically, the whole set L has a least and greatest element, which we denote $\bigwedge L = \bot$ and $\bigvee L = \top$, respectively.

In this report we usually deal with the partially ordered set of the powerset of some set and ordering given by inclusion. Indeed, for a set S, $\langle \mathcal{P}(S), \subseteq \rangle$ is a complete lattice. For $U \subseteq \mathcal{P}(S)$, $\bigvee U = \bigcup U$, and $\bigwedge U = \bigcap U$. The least and greatest elements are $\emptyset$ and S, respectively.

Theorem 2.3 (Knaster-Tarski Fixed Point Theorem [1:Theorem 1.2.8]): Let $\langle L, \leq \rangle$ a complete lattice and $f : L \to L$ monotone ($f(x) \leq f(y)$ when $x \leq y$). Then, the set of fixed points $\{x \in L | f(x) = x\}$, is a complete lattice. Particularly, the function has a *least fixed point* (lfp) and a *greatest fixed point* (gfp).

There is a useful way of constructing these least and greatest fixed points. This is done by repeated function application on $\bot$ for the least fixed point, and $\top$ for the greatest fixed point. Concretely, we define for a monotone $f : L \to L$, for α an ordinal, and β a limit ordinal:

$$\begin{aligned} f^0 &:= \bot \\ f^{\alpha+1} &:= (f^\alpha) \\ f^\beta &:= \bigvee \{f^\alpha \mid \alpha < \beta\} \end{aligned} \tag{1}$$

This constructs an increasing chain

$$\bot = f^0 \leq f^1 \leq f^2 \leq ... \tag{2}$$

which eventually stabilizes, giving the least fixed point, as stated by the following theorem:

Theorem 2.4 ([1:Theorem 1.2.11]): There exists an ordinal κ, such that $f^{\kappa} = f^{\kappa+1}$, which implies that f^{κ} is a fixed point of f. Furthermore, f^{κ} is the least fixed point of f. The dual process, beginning from $\top$ and moving downward, constructs the greatest fixed point of f.

3 Main Results

3.1 Modal Mu-Calculus

Modal mu-calculus is a powerful framework, used to verify properties of transition systems [1, 3]. We use it in Section 3.3 to select the right accepting words for our coalgebraic system. In this section we give a concrete definition of modal mu-calculus formulas and provide intuition on how to use it to verify certain properties.

In this section we will consider *labeled transition systems* (LTS) for which we will verify some properties. These systems are a little bit different than those we consider in the rest of the report, but they are useful for explaining modal mu-calculus. Concretely, an LTS is a tuple $\langle X, \Sigma, \delta, \mathit{Prop}, \rho\rangle$. Here X is the set of states, Σ the set of labels, $\delta : X \times \Sigma \to \mathcal{P}(X)$ the transition function (we write $x \overset{a}{\to} y$ for $y \in \delta(x)(a)$), *Prop* the set of atomic propositions, and $\rho : \mathit{Prop} \to \mathcal{P}(X)$ which interprets the atomic propositions. Note that you can see an LTS as a nondeterministic finite automaton without accepting states where the states are labeled by atomic propositions $\mathcal{P}(\mathit{Prop})$.

Definition 3.1: A modal mu-calculus formula is defined by the grammar:

$$\varphi \coloneqq P \mid Z \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid [a]\varphi \mid \langle a\rangle\varphi \mid \neg\varphi \mid \mu Z.\varphi \mid \nu Z.\varphi \tag{3}$$

where $P \in \mathit{Prop}$ is an atomic proposition, $a \in \Sigma$ a label, and $Z \in \mathit{Var}$ which is a set of second order variables. We require that in $\nu Z.\varphi$ and $\mu Z.\varphi$, every occurance of Z in φ is in the scope of an even number of negations, such that that φ is a monotone function.

Note that you could define the modal mu-calculus without the $\vee$, $\langle a\rangle$, and ν operators, and define these instead in terms of the other operators, but we include them in the definition for legibility.

We do not give a formal definition of the semantics of the formulas for an LTS, but rather give an informal and intuitive explanation, see [1, 3] for the formal definition. We want to say whether in an LTS T a formula φ holds in a state x, which we denote $x \vDash \varphi$. The semantics are then roughly as follows:

- $x \vDash P$, if the atomic proposition P holds in x;
- $x \vDash \varphi_1 \wedge \varphi_2$, $x \vDash \varphi_1 \vee \varphi_2$, if in state x both or either, respectively, formulas φ_1 and φ_2 hold;
- $x \vDash [a]\varphi$ if for all a transitions taken from x, φ holds in the next state;
- $x \vDash \langle a\rangle\varphi$ if there is some a transition from x where φ holds after the transition;
- $x \vDash \mu Z.\varphi$, if $x \in \mathit{lfp}(\lambda U.\ \|\varphi\|[Z \mapsto U])$, i.e. x is in the least fixed point of the monotone function that takes U and replaces very occurance of Z in φ with U. Again, this is not a formal definition. The ν operator is defined analogously for the greatest fixed point. We will see that, intuitively, the μ operator is for finite looping and ν is for infinite looping. To make this clear, let us look at the LTS given in Figure 2, and consider some examples of modal mu-calculus formulas.

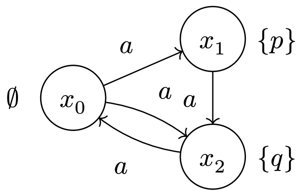

Figure 2: Example of an LTS. The sets next to the states denote the atomic propositions that hold in that state.

We have $x_0 \vDash \langle a\rangle p$, because there is an a transition that takes us to a state where p holds, namely $x_0 \overset{a}{\to} x_1$, because $x_1 \vDash p$. We, however, do not have $x_0 \vDash [a]p$, because $x_0 \overset{a}{\to} x_2$ and $x_2 \nvDash p$.

To observe that μ incites finite looping, we look at the fact that $x_0 \vDash \mu Z.q \vee [a]Z$. This roughly means that all a paths have q eventually, which we can see holds in Figure 2. To more formally show that this holds, we make use of the method of constructing least and greatest fixed points in Theorem 2.4. The function we are calculating the lfp for is $f := \lambda U.\ \|q\| \cup \|[a]U\|$. The first iteration yields $f^1 = f(\emptyset) = \{x_2\}$, because $x_2 \vDash q$. Continuing, $f^2 = \{x_1, x_2\}$ and $f^3 = \{x_0, x_1, x_2\} = f^4$. So the lfp is the entire set of states X, and thus $x_0 \vDash \mu Z.q \vee [a]Z$.

Next we look at ν, which can be used for infinite looping. We show that $x_0 \vDash \nu Z.\langle a\rangle Z$. This intuitively means that there exists an infinite path of as. Indeed, we observe there are multiple such paths, also starting at x_0. We confirm by computing the gfp: $f^1 = f(X) = \langle a\rangle X = X$. Dually, observe that the lfp of this formula is $f^1(\emptyset) = \emptyset$. So we do not have $x_0 \vDash \mu Z.\langle a\rangle Z$. This confirms the intuition given above that μ is for finite looping: there has to be some end point of the loop.

We briefly discuss here the game semantics view of deciding whether $x \vDash \varphi$. We do not go into much detail, but this is to introduce the topic and show what we want to look into applying for the coalgebraic representation of the Büchi automaton as we describe in Section 4. We can construct a two player game between a verifier and a refuter, who want to verify, respectively, refute $x \vDash \varphi$. For example, if $\varphi = \varphi_1 \vee \varphi_2$, only one of the two options has to hold, so the verifier can choose which one they will verify. If $\varphi = \varphi_1 \wedge \varphi_2$, only one of the two has to *not* hold in order to refute $x \vDash \varphi$, so the refuter can pick which one the verifier has to verify holds, which the refuter thinks they cannot. Analogously for $\langle\sigma\rangle\varphi$ and $[a]\varphi$ where the verifier can pick a σ transition to verify for $\langle\sigma\rangle\varphi$, and the refuter can pick a transition to refute for $[\sigma]\varphi$. If we work this out more, we can prove that $x \vDash \varphi$ iff there is a winning strategy for the verifier, and $x \nvDash \varphi$ iff the refuter has a winning strategy, see [3] for more details.

3.1.1 System of Equations

Next we introduce a system of equations for alternating fixed points. We only show how such a system works for two equations to save space and because that is all we use in the rest of the report. For more detail into this specific topic see [1, 7].

Definition 3.2: Let L_1, L_2 be partially ordered sets. An *equational system* is a system of two equations

$$u_1 \underset{\eta_1}{=} f_1(u_1, u_2) \qquad u_2 \underset{\eta_2}{=} f_2(u_1, u_2) \tag{4}$$

where u_1, u_2 are variables, $\eta_1, \eta_2 \in \{\mu, \nu\}$, and $f_i : L_1 \times L_2 \to L_i$ are monotone functions. The solution to the system is defined by the following set of steps:

The intermediate solution $l_1^{(1)} := \eta_1 u_1.f_1(u_1, u_2)$, where we take the lfp if $\eta_1 = \mu$ and gfp if $\eta_1 = \nu$. Note that $l_1^{(1)} : L_2 \to L_1$.

The solution to the second equation is then given by $l^{\text{sol}} := \eta_2 u_2.f_2\left(l_1^{(1)}(u_2), u_2\right)$, where again we take the lfp if $\eta_2 = \mu$, and gfp if $\eta_2 = \nu$. The solution to the first equation is then $l_1^{\text{sol}} = l_1^{(1)}\left(l_2^{\text{sol}}\right)$.

3.2 Finite Behavior Nondeterministic Systems

In this section we present a coalgebraic representation of nondeterministic systems. The next section for Büchi automata builds upon this construction.

3.2.1 Deterministic Automata

First we consider a deterministic finite automaton, $\langle X, \Sigma, \delta, o\rangle$ with X the states, Σ the alphabet, $\delta : X \times \Sigma \to X$ the transition function, and $o : X \to 2$ with $2 = \{0, 1\}$, the output function determining if a state is final. Such an automaton can be represented by a coalgebra $c : X \to 2 \times X^\Sigma$ for the functor $F(X) = 2 \times X^\Sigma$. This a very useful construction because a final coalgebra for this functor is carried by 2^{Σ^*}, and the unique coalgebra homomorphism to this final coalgebra captures exactly the language accepted by a state [6]. This is shown in the commuting diagram:

$$\begin{tikzcd}
2 \times X^{\Sigma} \arrow[r, dash, "\mathrm{id} \times \mathsf{beh}^{\Sigma}"] & 2 \times \left(2^{\Sigma^*}\right)^{\Sigma} \\
X \arrow[u, "{\langle o, \delta \rangle}"] \arrow[r, "\mathsf{beh}"'] & 2^{\Sigma^*} \arrow[u, "{\langle e, d \rangle}"']
\end{tikzcd} \tag{5}$$

Working out the paths through the diagram we obtain that
- $\mathsf{beh}(x)(\varepsilon) = o(x)$, and
- $\mathsf{beh}(x)(\sigma w) = \mathsf{beh}(\delta(x)(\sigma))(w)$,

for $x \in X$, $\sigma \in \Sigma$, $w \in \Sigma^*$. So $\mathsf{beh}(x)$ accepts the empty word if x is a final state, and accepts σw if $\delta(x)(\sigma)$ accepts w. Which is precisely the language accepted by state x in the deterministic finite automaton!

3.2.2 Finite Behavior Nondeterministic Systems

We would like to do the same thing for nondeterministic systems, but we run into a problem, which is highlighted by the following example, shown in Figure 3.

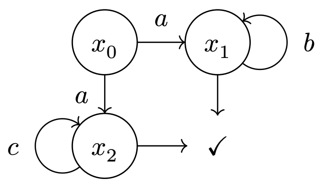

Figure 3: Example of a nondeterministic system.

The system is an LTS without atomic propositions but with termination, denoted by the transition to ✓. It is a nondeterministic system because in x_0 there are two transitions for a, and in x_1 and x_2 the system can transition back to itself or to ✓. Intuitively, the finite words accepted by the system from state x_0 should be

$$\mathsf{beh}(x_0) = \{a, ab, abb, ...\} \cup \{a, ac, acc, ...\}. \tag{6}$$

This transitions system might be modeled by a coalgebra $c : X \to \mathcal{P}(1 + \Sigma \times X)$, i.e. for every state some subset of a terminating transition or reading a letter and transitioning to another state. The problem is that this functor $FX = \mathcal{P}(1 + \Sigma \times X)$ does not have a final coalgebra. Because, by Lambek's lemma, such a final coalgebra $z : Z \to \mathcal{P}(1 + \Sigma \times Z)$ for some carrying object Z, would have to be an isomorphism [2]. But an isomorphism $Z \cong \mathcal{P}(1 + \Sigma \times Z)$ would imply a bijection between Z and $\mathcal{P}(Z)$, which cannot exist.

The solution, as given by Hasuo et al. [4], is to work in the Kleisli category for the monad $\mathcal{P}$. For this to work we have to define some details regarding construction of the coalgebra in the Kleisli Category. We give the resulting commuting diagram to show what we are working towards:

$$\begin{tikzcd}
\overline{F}X \arrow[r, dashed, "\overline{F}(\mathsf{beh}_c)"] & \overline{F}A \\
X \arrow[u, "c"] \arrow[r, dashed, "\mathsf{beh}_c"'] & A \arrow[u, "\cong", "J\alpha^{-1}"']
\end{tikzcd} \qquad \textit{in } \mathcal{K\ell}(\mathcal{P}). \tag{7}$$

Here $c : X \to \overline{F}X = \mathcal{P}(1 + \Sigma \times X)$ will be the coalgebra our nondeterministic system, and $\mathsf{beh}_c : X \to A$ in $\mathcal{K\ell}(\mathcal{P})$ is the unique map from X to A, which contains the finite accepting words in the nondeterministic system. Key here is that because we are working in the Kleisli category the map beh_c is actually a map $\mathsf{beh}_c : X \to \mathcal{P}(A)$ in the category **Sets**, which captures exactly the desired finite words, thus solving the problem when trying to obtain the final coalgebra in the category **Sets** directly.

So we need to define the Kleisli Category, and define the right functor in $\mathcal{K\ell}(\mathcal{P})$ to give us the desired words.

The powerset monad $\mathcal{P}$ is defined by the unit $\eta_X : X \to \mathcal{P}(X)$ which sends an element of X to the singleton set, $\eta_X(x) = \{x\}$ for $x \in X$, and the multiplication $\mu_X : \mathcal{P}(\mathcal{P}(X)) \to \mathcal{P}(X)$ which takes the union of the sets, i.e. $\mu_X(A) = \bigcup_{a\in A} a$. For a function $f : X \to Y$ we get $\mathcal{P}(f) : \mathcal{P}(X) \to \mathcal{P}(Y)$ by $\mathcal{P}(f)(A) = \{f(a) \mid a \in A\}$. The Kleisli category for this monad is defined as follows:

- **objects**: the same as for **Sets**, sets
- **morphisms**: a morphism f from X to Y in $\mathcal{K}\ell(\mathcal{P})$ is a maps $f : X \to \mathcal{P}(Y)$ in **Sets**. For morphisms $f : X \to Y$ and $g : Y \to Z$ in $\mathcal{K}\ell(\mathcal{P})$ (so $f : X \to \mathcal{P}(Y)$ and $g : Y \to \mathcal{Z}$ in **Sets**) we define $(g \circ f)$ in $\mathcal{K}\ell(\mathcal{P})$ as $(\mu_Z \circ \mathcal{P}(g) \circ f)$ in **Sets**. Indeed $(\mu_Z \circ \mathcal{P}(g) \circ f) : X \to \mathcal{P}(Z)$, so $(g \circ f) : X \to Z$ in $\mathcal{K}\ell(\mathcal{P})$.

Next, we construct our functor in $\mathcal{K}\ell(\mathcal{P})$, which we call the lifting of F in $\mathcal{K}\ell(\mathcal{P})$, and denote $\overline{F}$. The key here is that because we are working in the Kleisli category, if we use the functor $FX = 1 + \Sigma \times X$, the coalgebra map $c : X \to FX$, will be a map $c : X \to \mathcal{P}(1 + \Sigma \times X) = \overline{F}X$ in $\mathcal{K}\ell(\mathcal{P})$, which is what we needed to model a nondeterministic transition.

This works easily on objects, $\overline{F}X = FX$, because in the Kleisli category, the objects are the same. But for morphisms we have to do a little bit more work. Observe that because a map $f : X \to Y$ in $\mathcal{K}\ell(\mathcal{P})$ is a map $f : X \to \mathcal{P}(X)$ in **Sets**, applying the functor on the map itself would yield $Ff : FX \to F\mathcal{P}(X)$. So what we need is a natural transformation $\lambda : F\mathcal{P} \Rightarrow \mathcal{P}F$, such that $1 + \Sigma \times (\mathcal{P}(X)) \xrightarrow{\lambda} \mathcal{P}(1 + \Sigma \times X)$. We define this as $* \mapsto \{*\}$ (note that we use $1 = \{*\}$), and $(\sigma, S) = \{(\sigma, x) | x \in S\}$ for $\sigma \in \Sigma$ and $S \subseteq X$. This follows intuitively if you observe that if from state s taking transition σ takes you to $\{x, y, z\}$ ($(\sigma, \{x, y, z\}) \in c(s)$, or $\{x, y, z\} \in \delta(s)(\sigma)$), you can also see this as transitions $\{(\sigma, x), (\sigma, y), (\sigma, z)\}$.

Finally, the main theorem from [4] (Theorem 3.3), and the last ingredient to make the construction work is that the initial algebra for the functor F in sets, gives us the final coalgebra for the lifted functor $\overline{F}$ in $\mathcal{K}\ell(\mathcal{P})$. Specifically, for this functor $FX = 1 + \Sigma \times X$ and its lifting as described above, the initial F-algebra $\alpha : FA \to A$ in **Sets** yields a final $\overline{F}$-coalgebra in $\mathcal{K}\ell(\mathcal{P})$ by:

$$(J\alpha)^{-1} = J(\alpha^{-1}) = \eta_{FA}\alpha^{-1} : A \to \overline{F}A \ in\ \mathcal{K}\ell(\mathcal{P}) \tag{8}$$

where $J = \eta_{FA}$ is the canonical adjunction associated with the Kleisli category [2, 4]. This result holds more generally: for the lifting monad $\mathcal{L}$, the subdistribution monad $\mathcal{D}$, and any shapely functor F, see [4] for more details.

Let us go back to our concrete example and instantiate the commuting diagram from before. The initial F-algebra for our functor $FX = 1 + \Sigma \times X$ in **Sets** is $[\mathsf{nil}, \mathsf{cons}] : 1 + \Sigma \times \Sigma^* \to \Sigma^*$. So we get the commuting diagram

$$\begin{tikzcd}
1 + \Sigma \times X \arrow[r, dashed, "1 + \Sigma \times \mathsf{beh}_c"] & 1 + \Sigma \times \Sigma^* \\
X \arrow[u, "c"] \arrow[r, dashed, "\mathsf{beh}_c"'] & \Sigma^* \arrow[u, "\cong"', "J[\mathsf{nil},\mathsf{cons}]^{-1}"]
\end{tikzcd} \quad in\ \mathcal{K}\ell(\mathcal{P}). \tag{9}$$

Following the paths within the diagram we obtain that

$$\begin{aligned} \varepsilon \in \mathsf{beh}_c(x) &\iff x \to \checkmark \\ \sigma w \in \mathsf{beh}_c(x) &\iff \exists y.\big(x \xrightarrow{\sigma} y \wedge w \in \mathsf{beh}_c(y)\big). \end{aligned} \tag{10}$$

Explained in words, a state accepts the empty word if it can transition to $\checkmark$, and it accepts σw for $\sigma \in \Sigma$ and $w \in \Sigma^*$ if it can transition with σ to a state which accepts w. Which is exactly the desired words!

3.2.3 Possibly Infinite Behavior

As a step towards infinite words in Büchi automata let us consider infinite words in Figure 3. We can slightly alter our previous construction to additionally obtain infinite words through this system. Concretely, the infinite words for the system in Figure 3 for x_0 are ab^ω and ac^ω.

The intuition for this new construction is as follows. In the previous section we constructed the final coalgebra for the lifted functor $\overline{F}$ using the initial F-algebra in **Sets**. In the example of the LTS with termination the initial algebra was carried by Σ^*. The final coalgebra in **Sets** for F is carried by $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$, the set of finite and infinite words. So if we use this final coalgebra instead of the initial algebra, do we obtain both the finite and infinite words?

Consider again the monad $\mathcal{P}$, our functor F (this too holds more general, see [4, 5]), and its lifting in the Kleisli category $\overline{F}$. For a final coalgebra $\xi : Z \to FZ$, the coalgebra

$$J\xi : Z \to \overline{F}Z \textit{ in } \mathcal{K}\ell(\mathcal{P}) \tag{11}$$

is *weakly final*. That means, for any coalgebra $c : X \to \overline{F}X$, there is a morphism $\mathsf{beh} : X \to Z$ in $\mathcal{K}\ell(\mathcal{P})$ such that the following diagram commutes

$$\begin{tikzcd} \overline{F}X \arrow[r, squiggly, "\overline{F}(\mathsf{beh}_c)"] & \overline{F}Z & \\ X \arrow[u, "c"] \arrow[r, squiggly, "\mathsf{beh}_c"'] \arrow[ur, phantom, "\cong"] & Z \arrow[u, "J\xi"'] & \textit{in } \mathcal{K}\ell(\mathcal{P}), \end{tikzcd} \tag{12}$$

but this morphism is not necessarily unique. However, there is a canonical choice beh_c^∞ among these morphisms, namely the one which is maximal with respect to inclusion. We call this function $\mathsf{beh}_c^\infty : X \to \mathcal{P}(Z)$ the *possibly-infinite* behavior for c.

Indeed, if we consider our running example LTS with termination, $\xi : \Sigma^\infty \to 1 + \Sigma \times \Sigma^\infty$ is the final F-coalgebra. Defined by $\xi(\varepsilon) = * \in 1$ and $\xi(\sigma w) = (\sigma, w)$. Instantiating the diagram in Equation 12, we obtain

$$\begin{gathered} \varepsilon \in \mathsf{beh}_c(x) \iff x \to \checkmark \\ \sigma w \in \mathsf{beh}_c(x) \iff \exists y. \Big(x \overset{\sigma}{\to} y \wedge w \in \mathsf{beh}_c(y)\Big). \end{gathered} \tag{13}$$

Which is the same as in Equation 10. However, because the domain is Σ^∞, we obtain different words when we take the maximal function satisfying these equations. Namely the finite words, in addition to the infinite ones! For the system in Figure 3 we get the same words as before, but additionally $\{ab^\infty, ac^\infty\} \subseteq \mathsf{beh}_c^\infty(x_0)$. Interestingly, taking the minimum morphism we again obtain just the finite words [4, 5].

3.3 Coalgebraic Representation Büchi Automata

We can apply the previous framework for possibly infinite words to our initial exmample for a Büchi automaton, in Figure 1. This would yield all infinite words through automaton: $(\texttt{request} \cdot \texttt{process}^\infty \cdot \texttt{return})^\omega$. This is not quite the desired outcome yet, because $\texttt{process}^\infty$ means it takes the `process` transition zero, some finite number, or an infinite number of times. How do we eliminate those words that process indefinitely? I.e. only accept those words under the Büchi acceptance criterion of passing through an accepting state infinitely many times.

A way of solving this is given by [7]. In short, the main idea of their paper is to divide the states into accepting and non-accepting states. Then, applying the previous construction using the final F-coalgebra in **Sets** we obtain two separate commuting diagrams for these disjoint sets of states. And finally, using greatest and least fixed points we can precisely pick exactly the accepting words for the Büchi automaton.

We first give the commuting diagrams which govern the behavior mappings. We are now considering Büchi automata, so the functor we consider is $FX = \Sigma \times X$, the final coalgebra for this functor is $d : \Sigma^\omega \to \Sigma \times \Sigma^\omega$, defined by $d(\sigma \cdot w) = (\sigma, w)$, and the monad is still $\mathcal{P}$. The lifting $\overline{F}$ is effectively the same, just without a case for $* \in 1$. We now consider the state space as a disjoint union $X = X_1 \cup X_2$ of non-accepting and accepting states, respectively. This gives rise to two separate coalgebras $c_i : X_i \to \overline{F}X$, defined by the restriction $c \circ \kappa_i : X_i \to \overline{F}X$ along the coprojection $\kappa_i : X_i \hookrightarrow X$ for $i \in \{1, 2\}$. We then get the two commuting diagrams:

$$\begin{tikzcd}
\Sigma \times X \arrow[r, squiggly, "\Sigma \times {[\mathsf{beh}_1, \mathsf{beh}_2]}"] & \Sigma \times \Sigma^\omega \\
X_1 \arrow[u, "c_1"] \arrow[r, squiggly, "\mathsf{beh}_1"'] \arrow[ur, phantom, "\underset{\mu}{=}"] & \Sigma^\omega \arrow[u, "\cong", "Jd"']
\end{tikzcd}
\qquad
\begin{tikzcd}
\Sigma \times X \arrow[r, squiggly, "\Sigma \times {[\mathsf{beh}_1, \mathsf{beh}_2]}"] & \Sigma \times \Sigma^\omega \\
X_2 \arrow[u, "c_2"] \arrow[r, squiggly, "\mathsf{beh}_2"'] \arrow[ur, phantom, "\underset{\nu}{=}"] & \Sigma^\omega \arrow[u, "\cong", "Jd"']
\end{tikzcd}
\quad \textit{in } \mathcal{K}\ell(\mathcal{P}). \tag{14}$$

Where $\underset{\mu}{=}$ and $\underset{\nu}{=}$ mean that we take the lfp behavior mapping in the left diagram to obtain beh_1, and the gfp in the right diagram to obtain beh_2. More concretely, $\mathsf{beh}_1 : X_1 \to \mathcal{P}(\Sigma^\omega)$ and $\mathsf{beh}_2 : X_2 \to \mathcal{P}(\Sigma^\omega)$, are the solutions to the following system of equations:

$$\begin{aligned} u_1 &\underset{\mu}{=} (Jd)^{-1} \odot \overline{F}[u_1, u_2] \odot c_1 \\ u_2 &\underset{\nu}{=} (Jd)^{-1} \odot \overline{F}[u_1, u_2] \odot c_2 \end{aligned} \tag{15}$$

By taking exactly those behavior mappings which are the solution to this system of equation, we take exactly those words that the Büchi automaton accepts. The proof that this works is mostly done in the following two lemmas.

Lemma 3.3 ([7:Lemma 4.4]): Let $\chi = ((X_1, X_2), \Sigma, \delta, s)$ be a Büchi automaton. Let $l_1^{\text{sol}}, l_2^{\text{sol}}$ be the solutions to the following equational system, where the variables u_1, u_2 range over $\mathcal{P}(\text{Run}_\chi)$

$$u_1 \overset{\mu}{=} \Diamond_\chi^1(u_1 \cup u_2) \qquad u_2 \overset{\nu}{=} \Diamond_\chi^2(u_1 \cup u_2) \tag{16}$$

Here: $\Diamond_\chi^i : \mathcal{P}(\text{Run}_\chi) \to \mathcal{P}(\text{Run}_{\chi, X_i})$ is given by $\Diamond_\chi^i R := \{(\sigma, x) \cdot \rho \mid \sigma \in \Sigma, x \in X_i, \rho = (\sigma_1, x_1)(\sigma_2, x_2)... \in R, x_1 \in \delta(x, \sigma)\}$. Then $l_1^{\text{sol}} = \text{AccRun}_{\chi, X_1}, l_2^{\text{sol}} = \text{AccRun}_{\chi, X_2}$.

A formal proof is found in Appendix A, but we can use the intuition from Section 3.1 to gain an understanding as to why this lemma holds. Namely the fact that the μ operator is for finite looping, it has to have some endpoint or exit out of the loop, and that ν is for infinite looping. So the second equation makes sure the run passes through X_2 infinitely many times. Note that it can still move through X_1, but it has to move through X_2 infinitely many times. The first equation, with the μ operator, makes sure that any run passing through X_1 passes to the second equation in some finite number of steps, where it passes through X_2 infinitely many times.

Note that this is where we think a game semantic view could be applied. We think we might be able to define an intuitive correspondence between taking transitions in the Büchi automaton and the runs 'passing' through the equations in Lemma 3.3, and use that to characterize a game to decide which words belong to $\text{Lang}(\chi)$.

Of course Lemma 3.3 worked with Runs, instead of actual words. The next lemma is closer to our desired result:

Lemma 3.4 ([7:Lemma 4.5]): Let $\chi = ((X_1, X_2), \Sigma, \delta, s)$ be a Büchi automaton. Let $l_1^{\text{sol}}, l_2^{\text{sol}}$ be the solutions to the following equational system, where the variables u_1, u_2 range over $(\mathcal{P}(\Sigma^\omega))^{X_i}$

$$u_1 \overset{\mu}{=} \Diamond_\delta^1(u_1 \cup u_2) \qquad u_2 \overset{\nu}{=} \Diamond_\delta^2(u_1 \cup u_2) \tag{17}$$

Where $\Diamond_\delta^i : (\mathcal{P}(\Sigma^\omega))^X \to (\mathcal{P}(\Sigma^\omega))^{X_i}$ is given by

$$\Diamond_\delta^i(\mathsf{beh})(x) = \{\sigma \cdot w \mid \sigma \in \Sigma, x' \in \delta(x)(\sigma), w \in \mathsf{beh}(x')\}. \tag{18}$$

Then the solutions $l_i^{\text{sol}} = \text{DelSt}\left(\text{AccRun}_{\chi, X_i}\right)$, i.e. the words accepted starting from X_i.

After observing that Equation 15 in fact coincides with the definition of $\Diamond_\delta$, we obtain the final theorem:

Theorem 3.5 ([7:Theorem 4.6]): Let $\chi = ((X_1, X_2), \Sigma, \delta, s)$ be a Büchi automaton. Then the behavior mappings $\mathsf{beh}_1, \mathsf{beh}_2$, which are the solution to the system of equations in Equation 15 coincide with the accepted language of χ: $\mathsf{beh}(\chi) = [\mathsf{beh}_1, \mathsf{beh}_2] \odot s(*) = \text{Lang}(\chi)$. Where we interpret $s \subseteq X$ as $s : 1 \to \mathcal{P}(X)$.

4 Conclusion and Future Work

In this report we have shown a coalgebraic representation of Büchi automata. The construction relies upon two key ideas: working in the Kleisli category for the monad $\mathcal{P}$ and deriving two separate commuting diagrams for the accepting and non-accepting states and obtaining the right words by utilizing fixed point equations from these two mappings.

We explained the model in the Kleisli category in Section 3.2 by showing how to construct a final coalgebra for finite words for a nondeterministic system. Subsequently we constructed a weakly final coalgebra to additionally obtain the infinite words within such a system. Building upon these ideas we derived the coalgebraic construction for Büchi automata in Section 3.3, making use of the modal mu-calculus explained in Section 3.1.

We provided a proof for Lemma 3.3, but not for Lemma 3.4, which is crucial for coincidence result in Theorem 3.5, and thus understanding why the construction indeed provides the words accepted by the Büchi automaton. Therefore, the first next step in the internship will be understanding the proof provided by [7].

After understanding the full proof of the coincidence result, we can start to think about replacing it using a different framework. Our goal is to replace it using a game semantics framework, which we briefly explained in Section 3.1 in relation to the modal mu-calculus. There, we showed how one can see the check whether a formula holds in a state as a two player game between a verifier and a refuter, who want to verify, respectively refute, that the formula holds. Our vision is that this view can be applied to whether a word is accepted by the coalgebraic model of a Büchi automaton, and that this could simplify the result.

Bibliography

[1] André Arnold and Damian Niwinski. 2001. *Rudiments of mu-calculus.* Elsevier.

[2] Steve Awodey. 2010. *Category theory.* OUP Oxford.

[3] Erich Grädel, Wolfgang Thomas, and Thomas Wilke. 2003. *Automata, logics, and infinite games: a guide to current research.* Springer.

[4] Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. 2007. Generic trace semantics via coinduction. *Logical Methods in Computer Science* 3, (2007).

[5] Bart Jacobs. 2004. Trace semantics for coalgebras. *Electronic Notes in Theoretical Computer Science* 106, (2004), 167–184.

[6] Jan JMM Rutten. 2000. Universal coalgebra: a theory of systems. *Theoretical computer science* 249, 1 (2000), 3–80.

[7] Natsuki Urabe, Shunsuke Shimizu, and Ichiro Hasuo. 2016. Coalgebraic Trace Semantics for Buechi and Parity Automata. In *27th International Conference on Concurrency Theory (CONCUR 2016)*, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 1–15.

[8] Moshe Y. Vardi. 1996. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*, Faron Moller and Graham Birtwistle (eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 238–266. https://doi.org/10.1007/3-540-60915-6_6

A Proofs

Proof of Lemma 3.3 : We prove this in three steps: we show some properties for the first intermediate solution $l_1^{(1)}$, subsequently we use that to show $l_2^{\mathrm{sol}} = \mathrm{AccRun}_{\mathcal{X},X_2}$, and finally we use both results to show $l_1^{\mathrm{sol}} = \mathrm{AccRun}_{\mathcal{X},X_1}$.

Let $|\rho|$ denote the length of the run ρ.

Let $k \in \omega$, $u_2 \in \mathcal{P}\left(\mathrm{Run}_{\mathcal{X},X_2}\right)$, and any (possibly infinite) run $\rho = (\sigma_1, x_1)(\sigma_2, x_2)... \in \mathrm{Run}_{\mathcal{X}}$,

$$\rho \in \left[\lambda u_1.\Diamond_{\mathcal{X}}^1(u_1 \cup u_2)\right]^k(\emptyset) \tag{19}$$

if and only if there exists an $i \leq k$, such that for all $j \leq i$ we have $x_j \in X_1$ and $(\sigma_{i+1}, x_{i+1})(\sigma_{i+2}, x_{i+2})... \in u_2$. Meaning, for i steps the run passes through X_1 and after that it moves into u_2.

This is the case because when applying the function on k times there occur at most k steps in X_1 due to the $\Diamond$ operator. After this the run has to move into u_2. The other direction is obvious.

Now we observe that

$$l_1^{(1)}(u_2) = \mu u_1.\Diamond_\chi^1(u_1 \cup u_2) = \bigcup_{k\in\omega} \left[\lambda u_1.\Diamond_\chi^1(u_1 \cup u_2)\right]^k(\emptyset) \tag{20}$$

So $\rho \in \mu u_1.\Diamond_\chi^1(u_1 \cup u_2)$ if and only if the run ρ moves after some finite number of steps into u_2.

Now for $k \in \omega$, and a (possibly infinite) run $\rho = (\sigma_1, x_1)(\sigma_2, x_2)... \in \mathrm{Run}_\chi$,

$$\rho \in \left[\lambda u_2.\Diamond_\chi^2\left(l_1^{(1)}(u_2) \cup u_2\right)\right]^k\left(\mathrm{Run}_{\chi,X_2}\right). \tag{21}$$

if and only $|\{i \mid x_i \in X_2\}| \geq k$, i.e. we have passed through X_2 at least k times.

This is because, again, we take some number of steps in X_2 due to the $\Diamond$ operator, and between these steps we can 'pass through' $l_1^{(1)}$ but then, as shown above, the run has to move back to X_2.

Analogous to the least fixed point, we now observe that

$$l_2^{\mathrm{sol}} = \nu u_2.\Diamond_\chi^2\left(l_1^{(1)}(u_2) \cup u_2\right) = \bigcap_{k\in\omega} \left[\lambda u_2.\Diamond_\chi^2\left(l_1^{(1)}(u_2) \cup u_2\right)\right]^k\left(\mathrm{Run}_{\chi,X_2}\right). \tag{22}$$

So $\rho \in \nu u_2.\Diamond_\chi^2\left(l_1^{(1)}(u_2) \cup u_2\right)$ if and only if for all k ρ has moved through X_2 at least k times. Meaning, ρ has passed through X_2 infinitely many times. So $l_2^{\mathrm{sol}} = \mathrm{AccRun}_{\chi,X_2}$.

Finally,

$$l_1^{\mathrm{sol}} = l_1^{(1)}\left(l_2^{\mathrm{sol}}\right). \tag{23}$$

So for $\rho \in l_1^{\mathrm{sol}}$, in finitely many steps the run moves to l_2^{sol}, for which it passes infinitely many times through X_2. So $l_1^{\mathrm{sol}} = \mathrm{AccRun}_{\chi,X_1}$. □