Coalgebraic Automata and Canonical Models of Moore Machines

Brendan Cordy

Department of Mathematics and Statistics,
McGill University, Montréal
Québec, Canada

August, 2008

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements of the degree of
Master of Science

Abstract

We give a concise introduction to the coalgebraic theory of Moore machines, and building on [6], develop a method for constructing a final Moore machine based on a simple modal logic. Completeness for the logic follows easily from the finality construction, and we furthermore show how this logical framework can be used for machine learning.

Résumé

Nous présentons une brève introduction à la théorie coalgèbrique des machines Moore et, en ajoutant à [6], nous développons une méthode pour la construction d'une machine Moore finale. Cette méthode est fondée sur une simple logique modale dont la complétude résulte facilement de la construction de finalité. De plus, nous démontrons comment notre cadre logique peut être utilisé pour l'apprentissage automatique.

Acknowledgments

I would like to thank my friend and supervisor, Prakash Panangaden, for his depth of enthusiasm and patience. His guidance and encouragement transformed what began as an aside into the core of this work.

I am forever indebted to my parents for their unconditional encouragement. It is to them that this work is dedicated.

Table of Contents

Introduction

The coalgebraic theory of automata has seen a burst of activity in recent years, spearheaded by the work of Rutten [15], Kurz [9] [11], and Jacobs [7]. The description of many varieties of automata in terms of coalgebras for an endofunctor is extremely natural, and bisimulation, which is one of the key ideas in automata theory, is a general coalgebraic construction.

Many important results in automata theory have turned out to be instances of more abstract coalgebraic theorems. In particular, the results in Chapter 2 concerning bisimulation should be familiar to those well versed in automata theory or machine learning. Moreover, the existence of a final coalgebra in a coalgebraic category of automata will immediately imply fundamental results about minimization.

In [6], the authors give a construction of a final Mealy machine using a very simple modal logic, and identify that machine as a canonical model for the logic. This identification gives rise to a simple completeness proof. We will do the same here, but for Moore machines, and in the special case of Kripke machines we will show that the logic can in fact be made significantly simpler.

In Chapter 5, we use this construction of the final Moore machine as the basis of an algorithm for Moore machine reconstruction. Suppose one is given some information about the behaviour of a Moore machine whose internal structure is unknown, in the form of a set of formulae in the aforementioned logic. The algorithm then uses this information to reconstruct a machine whose behaviour is consistent with this known information, which can be used as an approximation to the original machine.

Chapter 1

Categories and Coalgebras

Definition 1.1 *Given a functor $F : \mathbf{C} \to \mathbf{C}$, a coalgebra for F is a pair (A, α) where A is an object of $\mathbf{C}$ and $\alpha : A \to FA$ is a morphism, sometimes called a structure map.*

A homomorphism of coalgebras $f : (A, \alpha) \to (B, \beta)$ is a map $f : A \to B$ in $\mathbf{C}$ such that $Ff \circ \alpha = \beta \circ f$.

$$\begin{tikzcd}
A \arrow[r, "\alpha"] \arrow[d, "f"'] & FA \arrow[d, "Ff"] \\
B \arrow[r, "\beta"] & FB
\end{tikzcd}$$

The F-coalgebras and coalgebra homomorphisms form a category denoted by $\mathbf{CoAlg}(F)$.

If we regard a category as a generalized poset, and a functor F as a generalized monotone function, an F-coalgebra corresponds to a postfixed point of F. Furthermore, a final object (Z, ζ) of $\mathbf{CoAlg}(F)$ corresponds to the greatest fixed point of F, but in order for this generalization to make sense, the object Z must in some sense be fixed by F, which is the content of the following lemma.

Lemma 1.1 *(Co-Lambek Lemma) Let $F : \mathbf{C} \to \mathbf{C}$ be a functor. If (Z, ζ) is final in the category* $\mathbf{CoAlg}(F)$*, then ζ is an isomorphism.*

Proof. Certainly $(FZ, F\zeta)$ is an object of $\mathbf{CoAlg}(F)$. Note that by finality, there is a unique coalgebra homomorphism $f : (FZ, F\zeta) \to (Z, \zeta)$, and hence the bottom square in the diagram below commutes, while the top square trivially does.

$$\begin{tikzcd}
Z \arrow[r, "\zeta"] \arrow[d, "\zeta"'] & FZ \arrow[d, "F\zeta"] \\
FZ \arrow[r, "F\zeta"] \arrow[d, "f"'] & F^2Z \arrow[d, "Ff"] \\
Z \arrow[r, "\zeta"] & FZ
\end{tikzcd}$$

Hence both squares in the diagram commute, so the whole rectangle does, and therefore $f \circ \zeta : (Z, \zeta) \to (Z, \zeta)$ is a coalgebra homomorphism. However, since (Z, ζ) is final, the only such coalgebra homomorphism is the identity, so $f \circ \zeta = id_Z$. As well, the commutativity of the bottom square gives

$$\zeta \circ f = Ff \circ F\zeta = F(f \circ \zeta) = F(id_Z) = id_{FZ}$$

which implies that $f = \zeta^{-1}$ and hence ζ is an isomorphism.

□

Finality plays a central role in coalgebraic automata theory. In the following sections we will see that if we can describe a particular category of automata as a category of coalgebras for a functor, then a final coalgebra is in fact a sort of universal machine of that type.

Lemma 1.2 *Let $F : \mathbf{Set} \to \mathbf{Set}$ be a functor. Given F-coalgebras (S, α) and (T, β), the coproduct $(S + T, \xi)$ exists in $\mathbf{CoAlg}(F)$.*

Proof. Let $i_S : S \to S + T$ and $i_T : T \to S + T$ be the injections of S and T into their disjoint union. It is easy to see how $\xi : S + T \to F(S + T)$ should be defined by examining the diagram below.

$$\begin{tikzcd}
S \arrow[r, "i_S"] \arrow[d, "\alpha"'] & S+T \arrow[d, "\xi"'] & T \arrow[l, "i_T"'] \arrow[d, "\beta"'] \\
FS \arrow[r, "F(i_S)"'] & F(S+T) & FT \arrow[l, "F(i_T)"]
\end{tikzcd}$$

For $s \in S$, define $\xi(s) = F(i_S) \circ \alpha(s)$, and for $t \in T$, define $\xi(t) = F(i_T) \circ \beta$. Given any (U, γ) and coalgebra homomorphisms $f : (S, \alpha) \to (U, \gamma)$, $g : (T, \beta) \to (U, \gamma)$, there is a unique $f + g : (S + T, \xi) \to (U, \gamma)$ such that

$$\begin{tikzcd}
(S,\alpha) \arrow[r, "i_S"] \arrow[dr, "f"'] & (S+T,\xi) \arrow[d, dashed, "f+g"] & (T,\beta) \arrow[l, "i_T"'] \arrow[dl, "g"] \\
& (U,\gamma) &
\end{tikzcd}$$

where $(f + g)(s) = f(s)$ for $s \in S$ and $(f + g)(t) = g(t)$ for $t \in T$, exactly as in **Set**. It is a coalgebra homomorphism since, given $s \in S$,

$$\begin{aligned}
F(f+g) \circ \xi(s) &= F(f+g) \circ F(i_S) \circ \alpha(s) \\
&= F((f+g) \circ i_S) \circ \alpha(s) \\
&= F(f) \circ \alpha(s) \\
&= \gamma \circ f(s) \\
&= \gamma \circ f + g(s)
\end{aligned}$$

and similarly for $t \in T$. Uniqueness is immediate since $S + T$ is the coproduct in **Set**, so $f + g$ is the only function with $(f + g) \circ i_S = f$ and $(f + g) \circ i_T = g$.

□

Note that the same construction works for general (possibly infinite) coproducts. In the contexts we will examine, F-coalgebras will be sets with transition structure, and the coproduct of two such transition systems will turn out to be the new transition system obtained by viewing the two systems as one system with two independent components.

Lemma 1.3 *Let* $F : \mathbf{Set} \to \mathbf{Set}$ *be a functor. Given two* F*-coalgebra homomorphisms* $f, g : (S, \alpha) \to (T, \beta)$*, the coequalizer* $h : (T, \beta) \to (U, \gamma)$ *exists in* $\mathbf{CoAlg}(F)$*.*

Proof. Since f and g are functions in **Set** satisfying a particular commutativity condition, we can form their coequalizer $h : T \to U$ in **Set**. Consider the map $F(h) \circ \beta : T \to F(U)$.

$$\begin{aligned}
F(h) \circ \beta \circ f &= F(h) \circ F(f) \circ \alpha \\
&= F(h \circ f) \circ \alpha \\
&= F(h \circ g) \circ \alpha \\
&= F(h) \circ F(g) \circ \alpha \\
&= F(h) \circ \beta \circ g
\end{aligned}$$

Thus, by the universal property of h, there exists a unique $\gamma : U \to F(U)$ making the right square of the diagram below commute.

$$\begin{tikzcd}
S \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] \arrow[d, "\alpha"'] & T \arrow[r, "h"] \arrow[d, "\beta"] & U \arrow[d, "\gamma"] \\
FS \arrow[r, shift left, "Ff"] \arrow[r, shift right, "Fg"'] & FT \arrow[r, "Fh"] & FU
\end{tikzcd}$$

Clearly, given any other coalgebra (V, ξ), and a homomorphism $h' : (T, \beta) \to (V, \xi)$ with $h' \circ f = h' \circ g$, there is a unique $k : U \to V$ in **Set** with $h' = k \circ h$ by the universal property of h. To verify that k (which was constructed in **Set**) is a coalgebra homomorphism, note that by the assumption that h' is a homomorphism,

$$\begin{aligned}
Fh' \circ \beta &= \xi \circ h' \\
F(k \circ h) \circ \beta &= \xi \circ k \circ h \\
F(k) \circ F(h) \circ \beta &= \xi \circ k \circ h \\
F(k) \circ \gamma \circ h &= \xi \circ k \circ h \\
F(k) \circ \gamma &= \xi \circ k
\end{aligned}$$

where the last line uses the fact that coequalizers in any category are epi. Therefore, the coequalizer of f and g in **CoAlg**(F) is $h : (T, \beta) \to (U, \gamma)$.

□

Although the two lemmas above are restricted to the case where F is an endofunctor on the category of sets, there are no other assumptions made on F (in particular F need not preserve any limits or colimits). This result is summarized in the theorem below.

Theorem 1.1 *For any functor $F : \mathbf{Set} \to \mathbf{Set}$, all coproducts and coequalizers exist in $\mathbf{CoAlg}(F)$ and are constructed in $\mathbf{Set}$.*

We now define a slight generalization of a pullback, which will become quite prominent in the following chapter. Fix a category **C** and consider the diagram below.

$$\begin{tikzcd}
S \arrow[dr, "f"'] & & T \arrow[dl, "g"] \\
& U &
\end{tikzcd}$$

Definition 1.2 *An object V with maps $p_1 : V \to S$ and $p_2 : V \to T$ is said to be a* <u>*weak pullback*</u> *of f and g, if for any object X with maps $q_1 : X \to S$ and $q_2 : X \to T$*

such that $f \circ q_1 = g \circ q_2$, there exists a map $u : X \to V$ such that $p_1 \circ u = q_1$ and $p_2 \circ u = q_2$.

$$\begin{tikzcd}
 & X \arrow[d, "u"] \arrow[ldd, bend right, "q_1"'] \arrow[rdd, bend left, "q_2"] & \\
 & V \arrow[ld, "p_1"] \arrow[rd, "p_2"'] & \\
S \arrow[rd, "f"'] & & T \arrow[ld, "g"] \\
 & U &
\end{tikzcd}$$

Note that a weak pullback is simply a pullback without the uniqueness requirement on the map $u : X \to V$. As one would expect, a functor $F : \mathbf{C} \to \mathbf{D}$ is said to preserve weak pullbacks when it sends a weak pullback diagram in $\mathbf{C}$ to a weak pullback diagram in $\mathbf{D}$.

Definition 1.3 *A functor $F : \mathbf{C} \to \mathbf{D}$ preserves weak pullbacks if, given a weak pullback (V, p_1, p_2) of f and g in $\mathbf{C}$, for any object $Y \in \mathbf{D}$ with maps $r_1 : Y \to FS$ and $r_2 : Y \to FT$ such that $Ff \circ r_1 = Fg \circ r_2$, there exists a map $w : Y \to FV$ such that $Fp_1 \circ w = r_1$ and $Fp_2 \circ w = r_2$.*

$$\begin{tikzcd}
 & Y \arrow[d, "w"] \arrow[ldd, bend right, "r_1"'] \arrow[rdd, bend left, "r_2"] & \\
 & FV \arrow[ld, "Fp_1"] \arrow[rd, "Fp_2"'] & \\
FS \arrow[rd, "Ff"'] & & FT \arrow[ld, "Fg"] \\
 & FU &
\end{tikzcd}$$

Chapter 2

Bisimulation

Bisimulation is the coalgebraic analogue of a congruence relation in universal algebra. It is already widely studied and well understood in theoretical computer science.

Historically, bisimulation was invented by Park [13] as a formalization of ideas of Milner [12] in his studies of concurrent systems. Bisimulation has turned out to be a key notion in concurrency theory and process algebra. The same concept also arose in modal logic in the work of van Benthem [3] [4]. A fundamental result is that there is a simple modal logic characterization of bisimulation. Later it was realized that bisimulation is naturally a coalgebraic concept [1].

Throughout this section, we will assume that the functor $F : \mathbf{Set} \to \mathbf{Set}$ preserves weak pullbacks. However, the assumption is not used everywhere, so we will distinguish those results that require it by marking them with an asterisk. This section is a quick review of some of the most important results on bisimulation for coalgebras of endofunctors on **Set**, all of which are present in [15].

Definition 2.1 *A bisimulation between coalgebras (S, α) and (T, β) is a subset $R \subseteq S \times T$ with a coalgebra structure $\gamma : R \to FR$, such that both of the projections $p_1 : R \to S$ and $p_2 : R \to T$ are coalgebra homomorphisms.*

$$\begin{tikzcd}
& R \arrow[dl, "p_1"'] \arrow[d, dashed, "\gamma"] \arrow[dr, "p_2"] & \\
S \arrow[d, "\alpha"'] & FR \arrow[dl, "Fp_1"] \arrow[dr, "Fp_2"'] & T \arrow[d, "\beta"] \\
FS & & FT
\end{tikzcd}$$

If $(S, \alpha) = (T, \beta)$, then (R, γ) is called a bisimulation on (S, α), and if a bisimulation relation is also an equivalence relation, it is called a bisimulation equivalence. We will often refer to a bisimulation (R, γ) simply as R.

Theorem 2.1 *Let (S, α) and (T, β) be F-coalgebras. A function $f : S \to T$ is a coalgebra homomorphism iff its graph relation $Grph(f) = \{\langle s, f(s)\rangle\}$ is a bisimulation between (S, α) and (T, β).*

Proof. Let $\gamma : Grph(f) \to F(Grph(f))$ be such that $(Grph(f), \gamma)$ is a bisimulation between (S, α) and (T, β). Note that $p_1 : Grph(f) \to S$ is bijective, and ${p_1}^{-1} : S \to Grph(f)$ is a coalgebra homomorphism since

$$\begin{aligned}
\alpha \circ {p_1}^{-1} &= F({p_1}^{-1}) \circ Fp_1 \circ \alpha \circ {p_1}^{-1} \\
&= F({p_1}^{-1}) \circ \beta \circ p_1 \circ {p_1}^{-1} \\
&= F({p_1}^{-1}) \circ \beta
\end{aligned}$$

Thus, f must be a homomorphism because we can write $f = p_2 \circ {p_1}^{-1}$,

Conversely, suppose that $f : S \to T$ is a coalgebra homomorphism, and consider the coalgebra $(Grph(f), \gamma)$, where $\gamma = F{p_1}^{-1} \circ \alpha \circ p_1$.

$$\begin{aligned}
Fp_1 \circ \gamma &= Fp_1 \circ (Fp_1{}^{-1} \circ \alpha \circ p_1) \\
&= (Fp_1 \circ Fp_1{}^{-1}) \circ \alpha \circ p_1 \\
&= F(p_1 \circ p_1^{-1}) \circ \alpha \circ p_1 \\
&= F(id_S) \circ \alpha \circ p_1 \\
&= \alpha \circ p_1
\end{aligned}$$

$$\begin{aligned}
Fp_2 \circ \gamma &= Fp_2 \circ (Fp_1{}^{-1} \circ \alpha \circ p_1) \\
&= F(p_2 \circ p_1^{-1}) \circ \alpha \circ p_1 \\
&= Ff \circ \alpha \circ p_1 \\
&= \beta \circ f \circ p_1 \\
&= \beta \circ (p_2 \circ p_1^{-1}) \circ p_1 \\
&= \beta \circ p_2
\end{aligned}$$

Therefore, the projections $p_1 : Grph(f) \rightarrow S$ and $p_2 : Grph(f) \rightarrow T$ are both coalgebra homomorphisms, and hence $(Grph(f), \gamma)$ is a bisimulation between (S, α) and (T, β).

□

Lemma 2.1* *Let $f : (S, \alpha) \rightarrow (U, \gamma)$ and $g : (T, \beta) \rightarrow (U, \gamma)$ be coalgebra homomorphisms. Then the pullback (P, p_1, p_2) of f and g in* **Set** *gives a bisimulation between (S, α) and (T, β).*

Proof. First note the pullback in set is $P = \{\langle s, t \rangle \in S \times T \mid f(s) = g(t)\}$, which is clearly a subset of $S \times T$. We can use the assumption that F preserves weak pullbacks to construct a coalgebra structure on P.

$$\begin{tikzcd}
& & P \arrow[lld, "p_1"'] \arrow[d, dashed, "\xi"] \arrow[rrd, "p_2"] & & \\
S \arrow[d, "\alpha"] \arrow[rrd, "f"] & & FP \arrow[lld, "Fp_1"'] \arrow[rrd, "Fp_2"] & & T \arrow[d, "\beta"] \arrow[lld, "g"'] \\
FS \arrow[rrd, "Ff"'] & & U \arrow[d, "\gamma"] & & FT \arrow[lld, "Fg"] \\
& & FU & &
\end{tikzcd}$$

The top face in the cubic diagram above is the pullback of f and g, while the bottom face is a weak pullback by our assumption on F, and the two side faces determined by f and g commute by definition. Thus, we have maps $\alpha \circ p_1 : P \to FS$ and $\beta \circ p_2 : P \to FT$ such that $Ff \circ \alpha \circ p_1 = Fg \circ \beta \circ p_2$, so there exists a (not necessarily unique) $\xi : P \to FP$ making the whole diagram commute, and hence (P, ξ) is a bisimulation.

□

Lemma 2.2 *Given two homomorphisms $f : (U, \gamma) \to (S, \alpha)$ and $g : (U, \gamma) \to (T, \beta)$, the image $Im(f, g) = \{\langle f(u), g(u)\rangle \mid u \in U\}$ is a bisimulation between (S, α) and (T, β).*

Proof. Define $j : U \to Im(f, g)$ by $j(t) = \langle f(t), g(t)\rangle$, let $i : Im(f, g) \to U$ be any right inverse to j, and consider the diagram below.

$$\begin{tikzcd}
& Im(f,g) \arrow[ld, "p_1"'] \arrow[rd, "p_2"] \arrow[d, shift left, "i"] & \\
S & U \arrow[l, "f"] \arrow[r, "g"'] \arrow[u, shift left, "j"] & T
\end{tikzcd}$$

Note that each of the right angle triangles in the diagram above commutes. Now define a map $\xi : Im(f, g) \to F(Im(f, g))$ by $\xi = F(j) \circ \gamma \circ i$. It is easy to verify that $p_1 : Im(f, g) \to S$ and $p_2 : Im(f, g) \to T$ are both homomorphisms.

$$\begin{aligned}Fp_1 \circ \xi &= Fp_1 \circ (Fj \circ \gamma \circ i)\\ &= F(p_1 \circ j) \circ \gamma \circ i\\ &= Ff \circ \gamma \circ i\\ &= \alpha \circ f \circ i\\ &= \alpha \circ p_1\end{aligned}$$

$$\begin{aligned}Fp_2 \circ \xi &= Fp_2 \circ (Fj \circ \gamma \circ i)\\ &= F(p_2 \circ j) \circ \gamma \circ i\\ &= Fg \circ \gamma \circ i\\ &= \beta \circ g \circ i\\ &= \beta \circ p_2\end{aligned}$$

Therefore, the coalgebra $(Im(f,g),\xi)$ is a bisimulation between (S,α) and (T,β).

□

Lemma 2.3* *The relational composition $Q \circ R = \{\langle r,u\rangle \mid \exists s \in S \text{ with } \langle r,s\rangle \in R$ and $\langle s,u\rangle \in Q\}$ of two bisimulations $R \subseteq S \times T$ and $Q \subseteq T \times U$ is a bisimulation between (S,α) and (U,γ).*

Proof. We can define $Q \circ R = Im(r_1 \circ p_1, q_2 \circ p_2) = \{\langle r_1 \circ p_1(x), q_2 \circ p_2(x)\rangle \mid x \in X\}$, where X is the pullback of r_2 and q_1 in **Set**.

$$\begin{tikzcd}
 & & X \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & & \\
 & R \arrow[dl, "r_1"'] \arrow[dr, "r_2"] & & Q \arrow[dl, "q_1"'] \arrow[dr, "q_2"] & \\
S & & T & & U
\end{tikzcd}$$

By Lemma 2.1, the set X can be given a coalgebra structure, and by Lemma 2.2, $Q \circ R$ is then a bisimulation between (S, α) and (U, γ).

□

Theorem 2.2 *Given a family of bisimulations* (R_i, γ_i) *between two colagebras* (S, α) *and* (T, β)*, the union* $\bigcup_i R_i$ *is a bisimulation between* (S, α) *and* (T, β)*.*

Proof. Let $p_1 : \Sigma_i R_i \rightarrow S$ and $p_2 : \Sigma_i R_i \rightarrow T$ be the projections from the disjoint union of the R_i's in **Set**. We know that $\Sigma_i R_i$ can be given a coalgebra structure by Lemma 1.2, and since $\bigcup_i R_i = Im(p_1, p_2)$, $\bigcup_i R_i$ is a bisimulation by Lemma 2.2.

□

Corollary 2.1 *The set of all bisimulations between two coalgebras* (S, α) *and* (T, β) *is a complete lattice with suprema and infima given by*

$$\begin{aligned}\bigvee_i R_i &= \bigcup_i R_i \\ \bigwedge_i R_i &= \bigcup \{R \subseteq S \times T \mid R \textit{ is a bisimulation with } R \subseteq R_i, \forall i\}\end{aligned}$$

This means that there is a greatest bisimulation between (S, α) *and* (T, β)*, which we will denote by* $\sim_{\langle S,T \rangle}$*, or simply* $\sim_S$ *when* $(S, \alpha) = (T, \beta)$*.*

$$\sim_{\langle S,T \rangle} \quad = \quad \bigcup \{R \subseteq S \times T \mid R \textit{ is a bisimulation}\}$$

Proposition 2.1 *The greatest bisimulation,* $\sim_S$*, on a single coalgebra* (S, α) *is always a bisimulation equivalence.*

Proof. Clearly the diagonal relation Δ_S is a bisimulation on (S, α), and hence $\sim_S$ is reflexive. Given a bisimulation (R, γ) on (S, α), the relational inverse R^{op} is again a bisimulation. Let $i : R \to R^{op}$ be the isomorphism sending $\langle s, t\rangle \in R$ to $\langle t, s\rangle \in R^{op}$. We claim that $(R^{op}, F(i) \circ \gamma \circ i^{-1})$ is a bisimulation on (S, α).

$$\begin{aligned} Fp_1 \circ (Fi \circ \gamma \circ i^{-1}) &= F(p_1 \circ i) \circ \gamma \circ i^{-1} \\ &= F(p_2) \circ \gamma \circ i^{-1} \\ &= \alpha \circ p_2 \circ i^{-1} \\ &= \alpha \circ p_1 \end{aligned}$$

The argument is similar for p_2. Thus, both projections p_1 and p_2 are homomorphisms, so R^{op} is a bisimulation, and hence $\sim_S$ is symmetric. The transitivity of $\sim_S$ is immediate from Lemma 2.3. Therefore, $\sim_S$ is an equivalence relation.

□

Bisimulation equivalences on a coalgebra (S, α) and coalgebra homomorphisms with domain (S, α) are related by the following two propositions.

Proposition 2.2 * *A homomorphism $f : (S, \alpha) \to (T, \beta)$ defines a bisimulation equivalence on (S, α) given by $K(f) = \{\langle s, s'\rangle \mid f(s) = f(s')\}$.*

Proof. Clearly, $K(f)$ is an equivalence relation. Note that by definition, $K(f)$ is the pullback in **Set** of f along itself, and therefore by Lemma 2.1 it can be given the structure of a bisimulation.

□

Proposition 2.3 *Let R be a bisimulation equivalence on a coalgebra (S, α), and let $\epsilon_R : S \to S/R$ be the quotient map induced by R. Then there is a unique coalgebra structure $\alpha_R : S/R \to F(S/R)$ on S/R such that ϵ_R is a homomorphism.*

$$\begin{tikzcd} S \arrow[r, "\alpha"] \arrow[d, "\epsilon_R"'] & FS \arrow[d, "F(\epsilon_R)"] \\ S/R \arrow[r, "\alpha_R"] & F(S/R) \end{tikzcd}$$

Proof. By definition, ϵ_R is the coequalizer in **Set** of the two projections from $R \subseteq S \times S$ to S, so the result is immediate by Lemma 1.3. If we denote an element of the R-equivalence class of s by $[s]$, we can concretely define $\alpha_R([s]) = F(\epsilon_R) \circ \alpha(s)$, where $s \in [s]$.

□

If we are a given a coalgebra (S, α), we say that $s, s' \in S$ are bisimilar if there exists a bisimulation $R \subseteq S \times S$ with $\langle s, s' \rangle \in R$. Together, the two propositions above imply that $s, s' \in S$ are bisimilar if and only if there exists a homomorphism $f : (S, \alpha) \to (T, \beta)$ with $f(s) = f(s')$.

Recall that we can apply a function $f : S \to T$ to a relation $R \subseteq S \times S$ componentwise, to obtain $f(R) = \{\langle f(s), f(s') \rangle \mid \langle s, s' \rangle \in R\}$. Similarly, for a relation $R \subseteq T \times T$, $f^{-1}(R) = \{\langle s, s' \rangle \mid \langle f(s), f(s') \rangle \in R\}$.

Proposition 2.4* *Let $f : (S, \alpha) \to (T, \beta)$ be a homomorphism, then*

1. If $R \subseteq S \times S$ is a bisimulation on (S, α), $f(R)$ is a bisimulation on (T, β).

2. If $Q \subseteq T \times T$ is a bisimulation on (T, β), $f^{-1}(Q)$ is a bisimulation on (S, α).

Proof. Immediate from Lemma 2.3 and the observation that $f(R) = Grph(f)^{op} \circ R \circ Grph(f)$, and $f^{-1}(Q) = Grph(f) \circ Q \circ Grph(f)^{op}$.

□

Chapter 3

Machines as Coalgebras

In order for the general results on coalgebras and bisimulation developed in the two previous chapters to be applicable in the context of automata theory, we have to explain how we can describe automata as coalgebras for some functor.

We will begin by defining a few very simple classes of machines. Their coalgebraic descriptions will be used to construct definitions of more complex types of automata.

Example 3.1 *Output: Consider a structure which consists of a set of states, each of which has a successor state and an associated output in some output set $\mathcal{O}$. We call such a structure an output machine.*

We can model this structure as a coalgebra for the functor $F : \mathbf{Set} \to \mathbf{Set}$ defined by $FS = S \times \mathcal{O}$, and $Ff = f \times id_{\mathcal{O}}$. Such a coalgebra has the form

$$S \xrightarrow{\alpha} S \times \mathcal{O}$$

We will denote the product projections on $S \times \mathcal{O}$ by π_1 and π_2. Given a state $s \in S$, consider $\alpha(s)$. The first component, $\pi_1 \circ \alpha(s)$, represents the successor state of s, so we define $next_S = \pi_1 \circ \alpha$. Similarly, the second component, $\pi_2 \circ \alpha(s)$, represents *the output in state s, so let* $out_S = \pi_2 \circ \alpha$. *Hence a coalgebra for this functor is a set with a simple transition and output structure.*

A homomorphism $f : (S, \alpha) \to (T, \beta)$ *is a function* $f : S \to T$ *such that the diagram*

$$\begin{tikzcd}
S \arrow[r, "\alpha"] \arrow[d, "f"'] & S \times \mathcal{O} \arrow[d, "f \times id_{\mathcal{O}}"] \\
T \arrow[r, "\beta"] & T \times \mathcal{O}
\end{tikzcd}$$

commutes, which means that for each $s \in S$, $f(\pi_1(\alpha(s)) = \pi_1(\beta(f(s)))$, *and* $\pi_2(\alpha(s)) = \pi_2(\beta(f(s)))$, *or using our interpretation of these coalgebras as state transition machines above,* $f(next_S(s)) = next_T(f(s))$, *and* $out_S(s) = out_T(f(s))$.

A bisimulation between output machines (S, α) *and* (T, β) *is a relation* $R \subseteq S \times T$ *with a coalgebra structure* $\gamma : R \to R \times \mathcal{O}$ *such that*

$$\begin{tikzcd}
& R \arrow[dl, "p_1"'] \arrow[d, dashed, "\gamma"] \arrow[dr, "p_2"] & \\
S \arrow[d, "\alpha"'] & R \times \mathcal{O} \arrow[dl, "p_1 \times id_{\mathcal{O}}"] \arrow[dr, "p_2 \times id_{\mathcal{O}}"'] & T \arrow[d, "\beta"] \\
S \times \mathcal{O} & & T \times \mathcal{O}
\end{tikzcd}$$

commutes. It is easy to see that the commutativity of the diagram means exactly that if $\langle s, t \rangle \in R$, *then* $next_R(\langle s, t \rangle) = \langle next_S(s), next_T(t) \rangle$, *and* $out_R(\langle s, t \rangle) = out_S(s) = out_T(t)$.

Example 3.2 *Observation: If we let* $\mathcal{O}$ *be the powerset of some set of observables* $\{p_1, ...p_n\}$, *then a coalgebra for the functor described above associates a set of observations with each state in* S, *and we can interpret* $out_S(s)$ *as the subset of observables seen in the state* s. *We can model stochastic observations similarly, with* $\mathcal{O} = \mathcal{D}(\{p_1, ..., p_n\})$, *the set of sub-probability distributions on* $\{p_1, ..., p_n\}$.

Example 3.3 Input: Fix a set of inputs Σ, and consider a structure we will call an input machine, which consists of a set of states with a transition function $\delta : S \times \Sigma \to S$ that produces a successor state for each state and input pair.

We can model such a structure as a coalgebra for the functor $F : \mathbf{Set} \to \mathbf{Set}$ defined by $FS = S^\Sigma$, the set of functions from Σ to S, and $Ff = f^\Sigma$, where for $d : \Sigma \to S$, $f^\Sigma(d) = f \circ d$. In this case, a coalgebra has the form

$$\begin{tikzcd} S \arrow[r, "\alpha"] & S^\Sigma \end{tikzcd}$$

We can define a function $\delta_S : S \times \Sigma \to S$ by uncurrying α, i.e. $\delta_S(s, a) = \alpha(s)(a)$. In the following, we will often write $a \cdot s$ for $\delta_S(s, a)$. As usual we will extend $\delta_S : \Sigma \times S \to S$ to ${\delta_S}^* : \Sigma^* \times S \to S$ by induction. For each $a \in \Sigma$ and any $w \in \Sigma^*$, let ${\delta_S}^*(s, aw) = {\delta_S}^*(a \cdot s, w)$, or in the notation defined above, $aw \cdot s = w \cdot (a \cdot s)$.

A homomorphism $f : (S, \alpha) \to (T, \beta)$ is a function $f : S \to T$ such that the diagram

$$\begin{tikzcd} S \arrow[r, "\alpha"] \arrow[d, "f"'] & S^\Sigma \arrow[d, "f^\Sigma"] \\ T \arrow[r, "\beta"] & T^\Sigma \end{tikzcd}$$

commutes, which means that for each $s \in S$, $f^\Sigma(\alpha(s)) = \beta(f(s))$. Using our definition of δ_S above, $f(\delta_S(s, a)) = \delta_T(f(s), a)$ for all $a \in \Sigma$, or equivalently, $f(a \cdot s) = a \cdot f(s)$.

A bisimulation between input machines (S, α) and (T, β) is a relation $R \subseteq S \times T$ with a coalgebra structure $\gamma : R \to R^\Sigma$ such that

$$\begin{tikzcd}
 & R \arrow[dl, "p_1"'] \arrow[d, dashed, "\gamma"] \arrow[dr, "p_2"] & \\
S \arrow[d, "\alpha"'] & R^{\Sigma} \arrow[dl, "p_1^{\Sigma}"] \arrow[dr, "p_2^{\Sigma}"'] & T \arrow[d, "\beta"] \\
S^{\Sigma} & & T^{\Sigma}
\end{tikzcd}$$

commutes. Chasing an element $\langle s,t\rangle \in R$ through the diagram, we obtain $p_1^{\Sigma} \circ \gamma(\langle s,t\rangle) = \alpha \circ p_1(\langle s,t\rangle)$, i.e. $p_1 \circ \gamma(\langle s,t\rangle) = \alpha(s)$. We can write this equation as $p_1(a \cdot \langle s,t\rangle) = a \cdot s$ for all $a \in \Sigma$. Similarly, we can show $p_2(a \cdot \langle s,t\rangle) = a \cdot t$ for all $a \in \Sigma$, and the two together imply $a \cdot \langle s,t\rangle = \langle a \cdot s, a \cdot t\rangle$ for all $a \in \Sigma$.

We can use the functors defined above to give coalgebraic descriptions of various types of automata. The class of automata we are most concerned with is described below, and we will later identify two additional types of abstract machines which are special cases of this definition.

Definition 3.1 *A Moore machine with input alphabet Σ and output lattice $\mathcal{O}$ is a coalgebra for the functor $F : \mathbf{Set} \to \mathbf{Set}$ defined by $FS = S^{\Sigma} \times \mathcal{O}$, and $Ff = f^{\Sigma} \times id$, where $\mathcal{O}$ is a complete lattice.*

$$S \xrightarrow{\alpha} S^{\Sigma} \times \mathcal{O}$$

Such a coalgebra consists of two maps, a transition function $\pi_1 \circ \alpha : S \to S^{\Sigma}$, which we will uncurry as $\delta_S : S \times \Sigma \to S$ defined by $\delta_S(s,a) = (\pi_1 \circ \alpha(s))(a)$, and an output map $o_S = \pi_2 \circ \alpha : S \to \mathcal{O}$. Again we can extend $\delta_S : S \times \Sigma \to S$ to ${\delta_S}^* : S \times \Sigma^* \to S$, and we will write $a \cdot s$ for $\delta_S(s,a)$.

A homomorphism of Moore machines $f : (S,\alpha) \to (T,\beta)$ is a function $f : S \to T$ such that the diagram below commutes.

$$\begin{tikzcd}
S \arrow[r, "\alpha"] \arrow[d, "f"'] & S^\Sigma \times \mathcal{O} \arrow[d, "f^\Sigma \times id"] \\
T \arrow[r, "\beta"] & T^\Sigma \times \mathcal{O}
\end{tikzcd}$$

$$\begin{aligned}
(f^\Sigma \times id) \circ \alpha(s) &= \beta \circ f(s) \\
(f^\Sigma \times id)\langle \delta_S(s,-), o(s)\rangle &= \langle \delta_T(f(s),-), o(f(s))\rangle \\
\langle f \circ \delta_S(s,-), o(s)\rangle &= \langle \delta_T(f(s),-), o(f(s))\rangle
\end{aligned}$$

Observe that $f \circ \delta_S(s,-) = \delta_T(f(s),-)$ if and only if $f(a \cdot s) = a \cdot f(s)$ for all $a \in \Sigma$, and hence a homomorphism of coalgebras for this functor can be described as a function $f : S \to T$ such that $a \cdot f(s) = f(a \cdot s)$ for all $a \in \Sigma$ and $o_T(f(s)) = o_S(s)$, which is the standard automata theoretic notion of a homomorphism of Moore machines.

Definition 3.2 *A bisimulation between Moore machines (S, α) and (T, β) is a relation $R \subseteq S \times T$, with a coalgebra structure $\gamma : R \to R^\Sigma \times \mathcal{O}$ such that the diagram below commutes.*

$$\begin{tikzcd}
 & & R \arrow[dll, "p_1"'] \arrow[d, dashed, "\gamma"] \arrow[drr, "p_2"] & & \\
S \arrow[d, "\alpha"'] & & R^\Sigma \times \mathcal{O} \arrow[dll, "p_1^\Sigma \times id"] \arrow[drr, "p_2^\Sigma \times id"'] & & T \arrow[d, "\beta"] \\
S^\Sigma \times \mathcal{O} & & & & T^\Sigma \times \mathcal{O}
\end{tikzcd}$$

$$\begin{aligned}
\alpha \circ p_1(\langle s,t\rangle) &= (p_1^\Sigma \times id) \circ \gamma(\langle s,t\rangle) \\
\alpha(s) &= (p_1^\Sigma \times id) \circ \langle \delta_R(\langle s,t\rangle), o_R(\langle s,t\rangle)\rangle \\
\langle \delta_S(s,-), o_S(s)\rangle &= \langle p_1 \circ \delta_R(\langle s,t\rangle,-), o_R(\langle s,t\rangle)\rangle
\end{aligned}$$

$$\begin{aligned} \beta \circ p_2(\langle s, t\rangle) &= (p_2^\Sigma \times id) \circ \gamma(\langle s, t\rangle) \\ \beta(t) &= (p_2^\Sigma \times id) \circ \langle \delta_R(\langle s, t\rangle), o_R(\langle s, t\rangle)\rangle \\ \langle \delta_T(t, -), o_T(t)\rangle &= \langle p_2 \circ \delta_R(\langle s, t\rangle, -), o_R(\langle s, t\rangle)\rangle \end{aligned}$$

In the first component of each the two commutativity conditions, $p_1 \circ \delta_R(\langle s, t\rangle, -) = \delta_S(s, -)$ and $p_2 \circ \delta_R(\langle s, t\rangle, -) = \delta_T(t, -)$, or equivalently, $p_1(a \cdot \langle s, t\rangle) = a \cdot s$, and $p_2(a \cdot \langle s, t\rangle) = a \cdot t$, for all $a \in \Sigma$. These two conditions can be written together in a more compact form, $a \cdot \langle s, t\rangle = \langle a \cdot s, a \cdot t\rangle$ for all $a \in \Sigma$.

The second components simply state that $o_R(\langle s, t\rangle) = o_S(s) = o_T(t)$. Therefore, we recover the usual definition of a bisimulation for Moore machines, that is, a relation $R \subseteq S \times T$ such that if $\langle s, t\rangle \in R$, then $\langle a \cdot s, a \cdot t\rangle \in R$, and $o_S(s) = o_T(t)$.

Now that we have verified that the coalgebraic description of a Moore machine is equivalent to specifying a set of states S, transition function $\delta_S : S \times \Sigma \to S$, and output function $o_S : S \to \mathcal{O}$, we will frequently refer to Moore machine as triples (S, δ_S, o_S) as well as coalgebras (S, α) for F.

Classically, a Moore machine has a finite set of outputs $\mathcal{O}$ with no additional structure. To describe such a machine, we can give $\mathcal{O}$ a trivial complete lattice structure by adding two outputs, $\top$ and $\bot$, which are never produced.

Now we need to establish that the Moore machine functor F preserves weak pullbacks. Once this is done, we know the functor F satisfies the assumptions at the beginning of Chapter 2, and thus all of the results on bisimulation developed there apply to Moore machines.

Theorem 3.1 *The Moore machine functor $F : \mathbf{Set} \to \mathbf{Set}$ defined by $FS = S^\Sigma \times \mathcal{O}$, and $Ff = f^\Sigma \times id$ preserves weak pullbacks.*

Proof. Suppose that the diagram below is a weak pullback in **Set**, that is, given any set X with functions $h : X \to S$ and $k : X \to T$ such that $f \circ h = g \circ k$, there exists a function $u : X \to P$ with $p_1 \circ u = h$ and $p_2 \circ u = k$.

$$\begin{tikzcd}
 & P \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & \\
S \arrow[dr, "f"'] & & T \arrow[dl, "g"] \\
 & U &
\end{tikzcd}$$

If we apply F to the diagram above, and suppose that there is a set X with maps $h : X \to S^\Sigma \times \mathcal{O}$ and $k : X \to T^\Sigma \times \mathcal{O}$ such that $(f \times id) \circ h = (g \times id) \circ k$ we obtain

$$\begin{tikzcd}
 & X \arrow[ddl, bend right, "h"'] \arrow[ddr, bend left, "k"] & \\
 & P^\Sigma \times \mathcal{O} \arrow[dl, "p_1 \times id"] \arrow[dr, "p_2 \times id"'] & \\
S^\Sigma \times \mathcal{O} \arrow[dr, "f \times id"'] & & T^\Sigma \times \mathcal{O} \arrow[dl, "g \times id"] \\
 & U^\Sigma \times \mathcal{O} &
\end{tikzcd}$$

Let $h_1 = \pi_1 \circ h$, $h_2 = \pi_2 \circ h$ and similarly for k_1, k_2, so $h = h_1 \times h_2$ and $k = k_1 \times k_2$.

Given $h_1 : X \to S^\Sigma$, any $a \in \Sigma$ determines a function $h_a = (h_1(-))(a) : X \to S$, and similarly $k_a = (k_1(-))(a) : X \to T$. Moreover, since $f \circ h_1 = g \circ k_1$, it is immediate that $f \circ h_a = g \circ k_a$, and therefore there is a $u_a : X \to P$ such that $p_1 \circ u_a = h_a$ and $p_2 \circ u_a = k_a$ because P is a weak pullback of f along g.

Let $v_1 : X \to P^\Sigma$ be defined by $(v_1(x))(a) = u_a(x)$. Then for all $a \in \Sigma$ and $x \in X$, $p_1 \circ (v_1(x))(a) = p_1 \circ u_a(x) = h_a(x) = h_1(x)(a)$, and hence $p_1 \circ v_1 = h_1$. Similarly

$p_2 \circ v_1 = k_1$. Let $v_2 = h_2 = k_2 : X \to \mathcal{O}$, and define $v : X \to P^\Sigma \times \mathcal{O}$ by $v = v_1 \times v_2$.

$$\begin{aligned}
(p_1 \circ v_1) \times v_2 &= h_1 \times h_2 \\
(p_1 \times id) \circ (v_1 \times v_2) &= h \\
(p_1 \times id) \circ v &= h
\end{aligned}$$

$$\begin{aligned}
(p_2 \circ v_1) \times v_2 &= k_1 \times k_2 \\
(p_2 \times id) \circ (v_1 \times v_2) &= k \\
(p_2 \times id) \circ v &= k
\end{aligned}$$

Therefore, $P^\Sigma \times \mathcal{O}$ with $p_1 \times id$ and $p_2 \times id$ is a weak pullback of $f \times id$ along $g \times id$, and therefore the Moore machine functor F preserves weak pullbacks.

□

Definition 3.3 *A subautomaton $(S', \delta_{S'}, o_{S'})$ of a Moore machine (S, δ_S, o_S) is a Moore machine with $S' \subseteq S$, and for which the inclusion function $i : S' \to S$ is a homomorphism.*

Given any Moore machine (S, δ_S, o_S) and any state $s \in S$, let $Rch(s)$ be the set of states reachable from s in a finite number of transitions. Formally, $Rch(s) = \{w \cdot s \mid w \in \Sigma^*\}$.

Definition 3.4 *The subautomaton of a Moore machine (S, δ_S, o_S) generated by a state $s \in S$, which will be denoted $\langle s \rangle_{(S,\delta_S,o_S)}$ is given by $(Rch(s), \delta_S, o_S)$.*

Observe that $\langle s \rangle_{(S,\delta_S,o_S)}$ is a well defined Moore machine with a nontrivial set of states. In particular, $s \in Rch(s)$ and certainly the restriction of δ_S and o_S to $Rch(S)$ makes $(Rch(s), \delta_S, o_S)$ a Moore machine.

Also, note that $\langle s \rangle_{(S,\delta_S,o_S)}$ is in fact a subautomaton of (S, δ_S, o_S), because $i(a \cdot s) = a \cdot s = a \cdot i(s)$ for all $s \in Rch(s)$, and clearly $o(s) = o(i(s))$ as well.

Lemma 3.1 *Any homomorphism of Moore machines $f : (S, \delta_S, o_S) \to (T, \delta_T, o_T)$ has the property that $f(w \cdot s) = w \cdot f(s)$ for all $w \in \Sigma^*$.*

Proof. If $w \in \Sigma$, then $f(a \cdot s) = a \cdot f(s)$ by definition. Assume $f(w' \cdot s) = w' \cdot f(s)$ for all w' of length n, then if w is of length $n+1$, we can write $w = w'a$ with $a \in \Sigma$ and w' of length n, and hence

$$\begin{aligned} f(w \cdot s) &= f(w'a \cdot s) \\ &= f(a \cdot (w' \cdot s)) \\ &= a \cdot f(w' \cdot s) \\ &= a \cdot (w' \cdot f(s)) \\ &= w'a \cdot f(s) \\ &= w \cdot f(s) \end{aligned}$$

Therefore, by induction on the length of w, $f(w \cdot s) = w \cdot f(s)$ for all $w \in \Sigma^*$.

□

Theorem 3.2 *Any homomorphism of Moore machines $f : (S, \delta_S, o_S) \to (T, \delta_T, o_T)$ has the property that $f(\langle s \rangle_{(S, \delta_S, o_S)}) = \langle f(s) \rangle_{(T, \delta_T, o_T)}$, i.e. the image of the subautomaton generated by s is the subautomaton generated by the image of s.*

Proof. Note that the transition and output functions of $f(\langle s \rangle_{(S, \delta_S, o_S)})$ are the restrictions of δ_T and o_T to $f(Rch(s))$, while the transition and output functions of $\langle f(s) \rangle_{(T, \delta_T, o_T)}$ are the restrictions of δ_T and o_T to Rch(f(s)). So all we need to show is that $Rch(f(s)) = f(Rch(s))$. But this is clear since by the previous lemma,

$$\begin{aligned} Rch(f(s)) &= \{w \cdot f(s) \mid w \in \Sigma^*\} \\ &= \{f(w \cdot s) \mid w \in \Sigma^*\} \\ &= f(\{w \cdot s \mid w \in \Sigma^*\}) \\ &= f(Rch(s)) \end{aligned}$$

□

An important special case of a Moore machine is a Kripke machine. In this case, each state is associated with one or more observations that may be witnessed when the machine is in that particular state, so an output is a set of observations. Alternatively, a Kripke machine is just a Moore machine where the output lattice carries a complete atomic boolean algebra structure.

Definition 3.5 *A Kripke machine with input alphabet Σ and observation set $\mathcal{O}$ is a coalgebra for the functor F :* **Set** $\rightarrow$ **Set** *defined by $FS = S^\Sigma \times \mathcal{PO}$, and $Ff = f^\Sigma \times id$.*

$$S \xrightarrow{\alpha} S^\Sigma \times \mathcal{PO}$$

Such a coalgebra also consists of two maps, a transition function $\pi_1 \circ \alpha : S \rightarrow S^\Sigma$, which we will uncurry as $\delta_S : S \times \Sigma \rightarrow S$ defined by $\delta_S(s, a) = (\pi_1 \circ \alpha(s))(a)$, and an observation map $o_S = \pi_2 \circ \alpha : S \rightarrow \mathcal{PO}$.

In the context of Kripke machines, a coalgebra homomorphism $f : (S, \alpha) \rightarrow (T, \beta)$ is again a function $f : S \rightarrow T$ such that $a \cdot f(s) = f(a \cdot s)$ for all $a \in \Sigma$ and $o_T(f(s)) = o_S(s)$. The definition of bisimulation remains unchanged as well.

Definition 3.6 *Let $\mathcal{D}(X)$ denote the set of sub-probability distributions on the set X, with an added constant $\perp$. A DASO (short for Deterministic Actions, Stochastic Observations) with input alphabet Σ and observation set $\mathcal{O}$ is a coalgebra for the functor $F : \mathbf{Set} \rightarrow \mathbf{Set}$ defined by $FS = S^{\Sigma} \times \mathcal{DO}$, and $Ff = f^{\Sigma} \times id$.*

We can obtain a transition function $\delta_S : S \times \Sigma \rightarrow S$ and an output function $o_S : S \rightarrow \mathcal{DO}$ in the same manner as before. The definition of homomorphism and bisimulation in terms of these functions remains unchanged.

Once again, a DASO is a special case of a Moore machine. We can define a complete lattice structure on $\mathcal{DO}$ by letting $P \leq_{\mathcal{D}} Q$ iff P dominates Q, i.e. $P(x) \leq Q(x)$ for all $x \in \mathcal{O}$. The meet of a set of distributions, $\bigwedge_{i \in I} P_i$ is given by the pointwise maximum

$$\bigwedge_{i \in I} P_i(x) \quad = \quad \sup_{i \in I} P_i(x)$$

If this fails to define a sub-probability distribution because $\sum_{x \in \mathcal{O}} \bigwedge_{i \in I} P_i(x) > 1$, then the meet is defined as $\perp$.

Chapter 4

Logic and Finality

Kripke Machines

Suppose we are given a Kripke machine with inputs in Σ and observations in $\mathcal{O}$, as in Definition 3.5, in the form of a black box which represents a certain function $f : \Sigma^* \to \mathcal{P}\mathcal{O}$, and our goal is to learn the behaviour of the machine, i.e. the function it represents. The only way to extract information about the behaviour of the machine is to watch it working. One must wait for it to take an input from its environment and observe which outputs, if any, the machine produces.

Note that in this situation, we are only given partial information about the behaviour of the machine. In particular, we cannot deny that the machine will ever display a certain behaviour simply because we have not observed it. This leads us to consider a positive logic where formulae represent observable behaviours.

With this motivation in mind, we define the simple modal logic below. Each formula in the logic will represent a finite observation about the behaviour of a Kripke machine.

Definition 4.1 *The syntax of $\mathcal{L}$ is defined by induction as follows, where $a \in \Sigma$, and $p \in \mathcal{O}$.*

$$\mathcal{L} \quad ::= \quad true \ \mid \ p \ \mid \ a(\phi) \ \mid \ \phi \wedge \psi$$

We will refer to the elements of $\mathcal{O}$ as atomic formulae, and given a set Φ of formulae in $\mathcal{L}$, define $At(\Phi)$ as the set of all atomic formulae in Φ. Note that the atomic formulae are also the atoms of the boolean algebra $\mathcal{PO}$.

Definition 4.2 *(Semantics) Given a Kripke machine (S, α), and a formula $\varphi \in \mathcal{L}$, the subset $[\![\varphi]\!] \subseteq S$ of states satisfying φ is defined by induction as*

$$s \in [\![true]\!] \textit{ for all } s \in S$$

$$s \in [\![p]\!] \textit{ if } p \in o(s)$$

$$s \in [\![a(\phi)]\!] \textit{ if } a \cdot s \in [\![\phi]\!]$$

$$s \in [\![\phi \wedge \psi]\!] \textit{ if } s \in [\![\phi]\!] \textit{ and } s \in [\![\psi]\!]$$

This simple logic represents a modified fragment of the Mealy logic presented in [6], which has been stripped down here and adapted to Kripke machines. Following [6], we define a logical consequence relation $\leq$ between formulae by the set of proof rules below. An inequality $\phi_1 \leq \phi_2$ should be read as "ϕ_1 implies ϕ_2". If the inequality $\phi_1 \leq \phi_2$ can be derived from a finite number of applications of the rules below, we write $\vdash \phi_1 \leq \phi_2$.

$$(ref) \ \ \phi \leq \phi \qquad\qquad (top) \ \ \phi \leq true$$

$$(\wedge_1) \ \ \phi_1 \wedge \phi_2 \leq \phi_1 \qquad\qquad (\wedge_2) \ \ \phi_1 \wedge \phi_2 \leq \phi_2$$

$$(trs) \ \ \frac{\phi_1 \leq \phi_2 \quad \phi_2 \leq \phi_3}{\phi_1 \leq \phi_3} \qquad (\wedge_i) \ \ \frac{\phi \leq \phi_1 \quad \phi \leq \phi_2}{\phi \leq \phi_1 \wedge \phi_2}$$

$$(\top)\ \ \mathit{true} \leq \top \qquad (\mathit{top}_a)\ \ \mathit{true} \leq a(\mathit{true})$$

$$(\wedge_a)\ \ a(\phi_1) \wedge a(\phi_2) \leq a(\phi_1 \wedge \phi_2)$$

$$(\leq_a)\ \ \frac{\phi_1 \leq \phi_2}{a(\phi_1) \leq a(\phi_2)}$$

Theorem 4.1 *(Soundness) If $\vdash \phi_1 \leq \phi_2$ then any state s of any Kripke machine (S, α) satisfying ϕ_1 also satisfies ϕ_2.*

Proof. By induction on the length of the proof of $\phi_1 \leq \phi_2$. Let (S, α) and $s \in S$ be given. Suppose $\vdash \psi_1 \leq \psi_2$ implies $s \in [\![\psi_1]\!] \Rightarrow s \in [\![\psi_2]\!]$ so long as $\psi_1 \leq \psi_2$ has a proof of length less than n, and let $\phi_1 \leq \phi_2$ have a proof of length n.

For the base case, we must confirm that for each inference rule without any hypotheses, if $s \in S$ satisfies the formula on the right hand side of the inequality, then it satisfies the formula on the left. The verification is routine and we will simply omit it here.

In the inductive step, we need to verify that any $s \in S$ satisfying the hypothesis of one of (trs), $(\wedge_i)$, or $(\leq_a)$ satisfies its conclusion. For (trs), $\phi_1 \leq \phi_2$ and $\phi_2 \leq \phi_3$ have proofs of length n or less by assumption, hence $s \in [\![\phi_1]\!] \Rightarrow s \in [\![\phi_2]\!]$ and $s \in [\![\phi_2]\!] \Rightarrow s \in [\![\phi_3]\!]$ by the induction hypothesis. Therefore $s \in [\![\phi_1]\!] \Rightarrow s \in [\![\phi_3]\!]$. Similarly, for $(\wedge_i)$, $s \in [\![\phi]\!] \Rightarrow s \in [\![\phi_1]\!]$ and $s \in [\![\phi]\!] \Rightarrow s \in [\![\phi_2]\!]$ by assumption, and hence $s \in [\![\phi]\!] \Rightarrow s \in [\![\phi_1 \wedge \phi_2]\!]$ by definition of the semantics of $\mathcal{L}$. Finally, for $(\leq_a)$, we know $s \in [\![\phi_1]\!] \Rightarrow s \in [\![\phi_2]\!]$ by assumption. If $s \in [\![a(\phi_1)]\!]$, then $a \cdot s \in [\![\phi_1]\!]$ by definition of the semantics, hence $a \cdot s \in [\![\phi_2]\!]$ as we just observed, and therefore $s \in [\![a(\phi_2)]\!]$ again by our semantics for $\mathcal{L}$.

□

Canonical Models

The following section gives a construction of the final Kripke machine with inputs in Σ and observations in $\mathcal{O}$, which is adapted from [6]. This construction uses the system of inference defined above and makes no mention of the final sequence typically used to construct final coalgebras [14]. Moreover, we identify the final Kripke machine as an analogue of the well known canonical model for a modal logic [5] in this setting.

Two formulae ϕ_1 and ϕ_2 are logically equivalent (written $\phi_1 \simeq \phi_2$) if $\phi_1 \leq \phi_2$ and $\phi_2 \leq \phi_1$. Logical equivalence is clearly an equivalence relation (see (ref) and (trs)), so we can construct the Lindenbaum algebra of $\mathcal{L}$, call it $\mathcal{L}/\simeq$. Let Θ be the set of filters of $\mathcal{L}/\simeq$.

We can now turn Θ into a Kripke machine, by defining $\zeta : \Theta \to \Theta^\Sigma \times \mathcal{PO}$ as follows. For a filter $F \in \Theta$, let $a \cdot F = \{\phi | a(\phi) \in F\}$, and $o(F) = At(F)$.

It is easy to see that $a \cdot F$ is again a filter since if $\phi_1, \phi_2 \in a \cdot F$, then $a(\phi_1), a(\phi_2) \in F$, so $a(\phi_1 \wedge \phi_2) \in F$ by $(\wedge_a)$, and hence $\phi_1 \wedge \phi_2 \in a \cdot F$. Furthermore, if $\phi_1 \in a \cdot F$ and $\phi_1 \leq \phi_2$, then $a(\phi_1) \leq a(\phi_2)$ by $(\leq_a)$, and hence $a(\phi_2) \in F$ since F is up-closed, so $\phi_2 \in a \cdot F$.

Theorem 4.2 *The Kripke machine* (Θ, ζ) *described above is final, i.e. for any Kripke machine* (S, α)*, there exists a unique homomorphism* $f_{sat} : (S, \alpha) \to (\Theta, \zeta)$ *sending a state of* (S, α) *to the set of (equivalence classes of) formulae it satisfies.*

Proof. Define $f_{sat} : S \to \Theta$ by $f_{sat}(s) = \{\phi | s \in [\![\phi]\!]\}$. The fact that $f_{sat}(s)$ is a filter follows easily from soundness and the definition of the semantics of $\mathcal{L}$. In particular, soundness implies that $f_{sat}(s)$ is an up-closed set of formulae, while the semantics ensure that $\phi_1, \phi_2 \in f_{sat}(s)$ implies $\phi_1 \wedge \phi_2 \in f_{sat}(s)$.

To verify $f_{sat} : (S, \alpha) \to (\Theta, \zeta)$ is a homomorphism, we must check that for each $a \in \Sigma$, $a \cdot f_{sat}(s) = f_{sat}(a \cdot s)$ and $o(f_{sat}(s)) = o(s)$. However, these conditions are immediate from the definition of f_{sat} and the semantics of $\mathcal{L}$.

$$\begin{aligned}
a \cdot f_{sat}(s) &= \{\phi | a(\phi) \in f_{sat}(s)\} \\
&= \{\phi | s \in [\![a(\phi)]\!]\} \\
&= \{\phi | a \cdot s \in [\![\phi]\!]\} \\
&= f_{sat}(a \cdot s)
\end{aligned}$$

$$\begin{aligned}
o(f_{sat}(s)) &= At(f_{sat}(s)) \\
&= \{p \in \mathcal{O} | p \in f_{sat}(s)\} \\
&= \{p \in \mathcal{O} | s \in [\![p]\!]\} \\
&= \{p \in \mathcal{O} | p \in o(s)\} \\
&= o(s)
\end{aligned}$$

To complete the proof, we also need to show uniqueness. Suppose $g : (S, \alpha) \to (\Theta, \zeta)$ is a homomorphism, and let $s \in S$. We prove $\phi \in f_{sat}(s)$ iff $\phi \in g(s)$ by structural induction on the formula ϕ.

$$\begin{aligned}
p \in g(s) &\iff p \in o(g(s)) \\
&\iff p \in o(s) \\
&\iff s \in [\![p]\!] \\
&\iff p \in f_{sat}(s)
\end{aligned}$$

$$\begin{aligned} a(\phi) \in g(s) &\iff \phi \in a \cdot g(s) \\ &\iff \phi \in g(a \cdot s) \\ &\iff \phi \in f_{sat}(a \cdot s) \\ &\iff \phi \in a \cdot f_{sat}(s) \\ &\iff a(\phi) \in f_{sat}(s) \end{aligned}$$

$$\begin{aligned} \phi_1 \wedge \phi_2 \in g(s) &\iff \phi_1 \in g(s) \;\; \textit{and} \;\; \phi_2 \in g(s) \\ &\iff \phi_1 \in f_{sat}(s) \;\; \textit{and} \;\; \phi_2 \in f_{sat}(s) \\ &\iff \phi_1 \wedge \phi_2 \in f_{sat}(s) \end{aligned}$$

Thus, for any Kripke machine (S, α), $f_{sat} : (S, \alpha) \to (\Theta, \zeta)$ is a homomorphism, and moreover, it is the unique homomorphism from (S, α) to (Θ, ζ). Therefore, the machine (Θ, ζ) is a final object in the category of Kripke machines.

□

The following lemma is commonly known as the truth lemma in modal logic [5]. Its content is the observation that the formulae contained in a filter $F \in \Theta$ are exactly the formulae which F satisfies as a state of (Θ, ζ). This lemma is key, as it shows that (Θ, ζ) plays the role of a canonical model for the modal logic $\mathcal{L}$, and moreover, it will allow for an extremely simple proof of completeness.

Lemma 4.1 *For all $\phi \in \mathcal{L}$ and all $F \in \Theta$, $F \in [\![\phi]\!] \iff \phi \in F$.*

Proof. By structural induction on ϕ.

$$\begin{aligned} F \in [\![p]\!] &\iff p \in o(F) \\ &\iff p \in At(F) \\ &\iff p \in F \end{aligned}$$

$$\begin{aligned} F \in [\![a(\phi)]\!] &\iff a \cdot F \in [\![\phi]\!] \\ &\iff \phi \in a \cdot F \\ &\iff \phi \in \{\psi | a(\psi) \in F\} \\ &\iff a(\phi) \in F \end{aligned}$$

$$\begin{aligned} F \in [\![\phi_1 \wedge \phi_2]\!] &\iff F \in [\![\phi_1]\!] \ \ \textit{and} \ \ F \in [\![\phi_2]\!] \\ &\iff \phi_1 \in F \ \ \textit{and} \ \ \phi_2 \in F \\ &\iff \phi_1 \wedge \phi_2 \in F \end{aligned}$$

□

Theorem 4.3 *(Completeness) If* $s \in [\![\phi_1]\!] \Rightarrow s \in [\![\phi_2]\!]$ *for all Kripke machines* (S, α)*, then* $\vdash \phi_1 \leq \phi_2$.

Proof. Assume $\not\vdash \phi_1 \leq \phi_2$. It suffices to find $s \in (S, \alpha)$ with $s \in [\![\phi_1]\!]$ but $s \notin [\![\phi_2]\!]$.

Let $F_{\phi_1} = \{\psi | \phi_1 \leq \psi\}$. This is clearly a filter, and hence also a state of (Θ, ζ). Note that $\phi_1 \in F_{\phi_1}$, but by assumption $\phi_2 \notin F_{\phi_1}$, and hence by Lemma 4.1, $F_{\phi_1} \in [\![\phi_1]\!]$ but $F_{\phi_1} \notin [\![\phi_2]\!]$.

□

Moore Machines

This construction of a final Kripke machine through a deductive system in fact generalizes easily to a much larger class of machines. To deal with general Moore machines, as described in Definition 3.1, we need to impose two new inference rules in our logic, which were already present in [6], but were not necessary for Kripke machines. These rules describe how the complete lattice structure of the outputs of a Moore machine interacts with the logic $\mathcal{L}$.

Note that a Kripke machine with observations in $\mathcal{O}$ is also a Moore machine, where the output lattice is $\mathcal{PO}$, ordered by reverse containment. The meet operation corresponds to union in this context.

Definition 4.3 *The syntax of $\mathcal{L}$ is once again defined by induction as follows, where $a \in \Sigma$, and $p \in \mathcal{O}$.*

$$\mathcal{L} \quad ::= \quad true \ \mid \ p \ \mid \ a(\phi) \ \mid \ \phi \wedge \psi$$

We will refer to the elements of $\mathcal{O}$ as atomic formulae, and given a set Φ of formulae in $\mathcal{L}$, define $At(\Phi)$ as the set of all atomic formulae in Φ. Note that the atomic formulae are not atoms of $\mathcal{O}$, but simply arbitrary elements.

Definition 4.4 *(Semantics) Given a Moore machine (S, α), and a formula $\varphi \in \mathcal{L}$, the subset $[\![\varphi]\!] \subseteq S$ of states satisfying φ is defined by induction as*

$$s \in [\![true]\!] \textit{ for all } s \in S$$

$$s \in [\![p]\!] \textit{ if } o(s) \leq_{\mathcal{O}} p$$

$$s \in [\![a(\phi)]\!] \textit{ if } a \cdot s \in [\![\phi]\!]$$

$$s \in [\![\phi_1 \wedge \phi_2]\!] \textit{ if } s \in [\![\phi_1]\!] \textit{ and } s \in [\![\phi_2]\!]$$

The system of inference for general Moore machines is similar to the Kripke machine case, but the atomic structure of $\mathcal{PO}$ allows for two fewer inference rules in the case of Kripke machines. In the general case, the rules $(\wedge_p)$, and $(\leq_p)$ are required to describe how the lattice structure on $\mathcal{O}$ interacts with the logic $\mathcal{L}$

$$(ref)\ \ \phi \leq \phi \qquad\qquad (top)\ \ \phi \leq true$$

$$(\wedge_1)\ \ \phi_1 \wedge \phi_2 \leq \phi_1 \qquad\qquad (\wedge_2)\ \ \phi_1 \wedge \phi_2 \leq \phi_2$$

$$(trs)\ \ \frac{\phi_1 \leq \phi_2 \quad \phi_2 \leq \phi_3}{\phi_1 \leq \phi_3} \qquad (\wedge_i)\ \ \frac{\phi \leq \phi_1 \quad \phi \leq \phi_2}{\phi \leq \phi_1 \wedge \phi_2}$$

$$(\top_p)\ \ true \leq \top \qquad\qquad (top_a)\ \ true \leq a(true)$$

$$(\wedge_p)\ \ p \wedge q \leq p \wedge_{\mathcal{O}} q \qquad\qquad (\wedge_a)\ \ a(\phi_1) \wedge a(\phi_2) \leq a(\phi_1 \wedge \phi_2)$$

$$(\leq_p)\ \ \frac{p \leq_{\mathcal{O}} q}{p \leq q} \qquad\qquad (\leq_a)\ \ \frac{\phi_1 \leq \phi_2}{a(\phi_1) \leq a(\phi_2)}$$

Theorem 4.4 *(Soundness) If $\vdash \phi_1 \leq \phi_2$ then any state s of any Moore machine (S, α) satisfying ϕ_1 also satisfies ϕ_2.*

Proof. The same as the Kripke machine case, but with the two additional rules $(\wedge_p)$ and $(\leq_p)$. Again we proceed by induction on the length of the proof. Let (S, α) and $s \in S$ be given. Suppose $\vdash \phi_1 \leq \phi_2$ implies $s \in \llbracket \phi_1 \rrbracket \Rightarrow s \in \llbracket \phi_2 \rrbracket$ so long as $\phi_1 \leq \phi_2$ has a proof of length less than n, and let $\psi_1 \leq \psi_2$ have a proof of length n.

For $(\wedge_p)$, suppose that $s \in \llbracket p \wedge q \rrbracket$. Then $s \in \llbracket p \rrbracket$ and $s \in \llbracket q \rrbracket$, so $o(s) \leq_{\mathcal{O}} p$ and $o(s) \leq_{\mathcal{O}} q$, and hence $o(s) \leq_{\mathcal{O}} p \wedge_{\mathcal{O}} q$. Therefore, $s \in \llbracket p \wedge_{\mathcal{O}} q \rrbracket$. Note that the induction hypothesis is not used here since $(\wedge_p)$ has no hypotheses. In the case of $(\leq_p)$, if $p \leq_{\mathcal{O}} q$ and $s \in [\![p]\!]$, then $o(s) \leq_{\mathcal{O}} p$, and therefore $o(s) \leq_{\mathcal{O}} q$, so $s \in [\![q]\!]$.

□

Canonical Models

Once again, let Θ be the set of all filters of $\mathcal{L}/\simeq$, and turn it into a Moore machine by defining $\zeta : \Theta \to \Theta^{\Sigma} \times O$ as follows. For a filter $F \in \Theta$, let $a \cdot F = \{\phi | a(\phi) \in F\}$, and $o(F) = \bigwedge_{\mathcal{O}} At(F)$.

Theorem 4.5 *The Moore machine (Θ, ζ) described above is final, i.e. for any Moore machine (S, α), there exists a unique homomorphism $f_{sat} : (S, \alpha) \to (\Theta, \zeta)$ sending a state of (S, α) to the set of (equivalence classes of) formulae it satisfies.*

Proof. Define $f_{sat} : (S, \alpha) \to (\Theta, \zeta)$ by $f_{sat}(s) = \{\phi | s \in [\![\phi]\!]\}$. The fact that $f_{sat}(s)$ is a filter follows easily from soundness and the semantics of $\mathcal{L}$, exactly as in the Kripke machine case. To verify f_{sat} is a homomorphism, we check that for each $a \in \Sigma$ we have

$$\begin{aligned} a \cdot f_{sat}(s) &= \{\phi | a(\phi) \in f_{sat}(s)\} \\ &= \{\phi | s \in [\![a(\phi)]\!]\} \\ &= \{\phi | a \cdot s \in [\![\phi]\!]\} \\ &= f_{sat}(a \cdot s) \end{aligned}$$

$$\begin{aligned} o(f_{sat}(s)) &= \bigwedge At(f_{sat}(s)) \\ &= \bigwedge \{p \in \mathcal{O} | p \in f_{sat}(s)\} \\ &= \bigwedge \{p \in \mathcal{O} | s \in [\![p]\!]\} \\ &= \bigwedge \{p \in \mathcal{O} | o(s) \leq_{\mathcal{O}} p\} \\ &= o(s) \end{aligned}$$

To complete the proof, we also need to show uniqueness. Suppose $g : (S, \alpha) \to (\Theta, \zeta)$ is a homomorphism, and let $s \in S$. We prove $\phi \in f_{sat}(s)$ iff $\phi \in g(s)$ by structural induction on the formula ϕ.

$$\begin{aligned}
p \in g(s) &\iff o(g(s)) \leq_{\mathcal{O}} p \\
&\iff o(s) \leq_{\mathcal{O}} p \\
&\iff s \in [\![p]\!] \\
&\iff p \in f_{sat}(s)
\end{aligned}$$

$$\begin{aligned}
a(\phi) \in g(s) &\iff \phi \in a \cdot g(s) \\
&\iff \phi \in g(a \cdot s) \\
&\iff \phi \in f_{sat}(a \cdot s) \\
&\iff \phi \in a \cdot f_{sat}(s) \\
&\iff a(\phi) \in f_{sat}(s)
\end{aligned}$$

$$\begin{aligned}
\phi \wedge \psi \in g(s) &\iff \phi \in g(s) \;\; \textit{and} \;\; \psi \in g(s) \\
&\iff \phi \in f_{sat}(s) \;\; \textit{and} \;\; \psi \in f_{sat}(s) \\
&\iff \phi \wedge \psi \in f_{sat}(s)
\end{aligned}$$

Therefore, for any Moore machine (S, α), there is a homomorphism $f_{sat} : (S, \alpha) \to (\Theta, \zeta)$, and it is the unique homomorphism from (S, α) to (Θ, ζ). I.e. the machine (Θ, ζ) is the final Moore machine.

□

Once again, an analogue of the truth lemma in modal logic [5] holds for the machine (Θ, ζ), i.e. the formulae of $\mathcal{L}$ that a filter $F \in \Theta$ satisfies as a state of (Θ, ζ) are exactly those formulae in F.

Lemma 4.2 *For all $\phi \in \mathcal{L}$ and all $F \in \Theta$, $F \in [\![\phi]\!] \iff \phi \in F$.*

Proof. By structural induction on the formula ϕ.

$$\begin{aligned} F \in [\![p]\!] &\iff o(F) \leq_{\mathcal{O}} p \\ &\iff \bigwedge At(F) \leq_{\mathcal{O}} p \\ &\iff p \in F \end{aligned}$$

$$\begin{aligned} F \in [\![a(\phi)]\!] &\iff a \cdot F \in [\![\phi]\!] \\ &\iff \phi \in a \cdot F \\ &\iff \phi \in \{\psi | a(\psi) \in F\} \\ &\iff a(\phi) \in F \end{aligned}$$

$$\begin{aligned} F \in [\![\phi \wedge \psi]\!] &\iff F \in [\![\phi]\!] \;\; \textit{and} \;\; F \in [\![\psi]\!] \\ &\iff \phi \in F \;\; \textit{and} \;\; \psi \in F \\ &\iff \phi \wedge \psi \in F \end{aligned}$$

Note that $\bigwedge At(F) \leq_{\mathcal{O}} p$ is equivalent to $p \in F$ because F is a filter, hence up-closed, and $q \leq_{\mathcal{O}} p$ implies $q \leq p$ in $\mathcal{L}/\leq$ by $(\leq_p)$.

□

Theorem 4.6 *(Completeness) If $s \in [\![\phi_1]\!] \Rightarrow s \in [\![\phi_2]\!]$ for all Moore machines (S, α), then $\vdash \phi_1 \leq \phi_2$.*

Proof. Exactly as in the Kripke machine case. Assume $\not\vdash \phi_1 \leq \phi_2$. It suffices to find $s \in (S, \alpha)$ with $s \in [\![\phi_1]\!]$ but $s \notin [\![\phi_2]\!]$.

Let $F_{\phi_1} = \{\psi | \phi_1 \leq \psi\}$. This is clearly a filter, hence a state of (Θ, ζ). Note that $\phi_1 \in F_{\phi_1}$, but $\phi_2 \notin F_{\phi_1}$, so by Lemma 4.2, $F_{\phi_1} \in [\![\phi_1]\!]$ but $F_{\phi_1} \notin [\![\phi_2]\!]$.

□

Consequences of Finality

The final Moore machine has a number of remarkable properties. The first is often called the coinductive proof principle [15], which states that two states of the final automaton are bisimilar iff they are identical (for more on coinduction, see [8] and [16]).

Theorem 4.7 *Let $\Delta_\Theta = \{\langle F, F\rangle \mid F \in \Theta\}$ be the diagonal relation on Θ. Then any for bisimulation $R \subseteq \Theta \times \Theta$, $R \subseteq \Delta_\Theta$. Equivalently, $\sim_\Theta = \Delta_\Theta$.*

Proof. Let $R \subseteq \Theta \times \Theta$ be a bisimulation on (Θ, ζ), then $p_1 : R \to \Theta$ and $p_2 : R \to \Theta$ are parallel homomorphisms, and hence $p_1 = p_2$ by finality of (Θ, ζ), so $R \subseteq \Delta_\Theta$.

□

Moreover, for any Moore machine (S, α), the unique map to $f_{sat} : (S, \alpha) \to (\Theta, \zeta)$ identifies all bisimilar states in S.

Theorem 4.8 *Let (S, α) be a Moore machine, and $f_{sat} : (S, \alpha) \to (\Theta, \zeta)$ the unique homomorphism to (Θ, ζ) described earlier. For all $s, s' \in S$,*

$$s \sim_S s' \quad \textit{iff} \quad f_{sat}(s) = f_{sat}(s')$$

Thus $f_{sat}(s)$ represents the $\sim_S$-equivalence class of the state s.

Proof. Let $s \sim_S s'$, then there exists a bisimulation $R \subseteq S \times S$ with $\langle s, s'\rangle \in R$. By Proposition 2.4, $f_{sat}(R)$ is a bisimulation on (Θ, ζ), and $\langle f_{sat}(s), f_{sat}(s')\rangle \in f_{sat}(R)$ by definition. However, $f_{sat}(R) \subseteq \Delta_\Theta$ as we have just shown, and hence $f_{sat}(s) = f_{sat}(s')$.

Conversely, $f_{sat}^{-1}(\Delta_\Theta)$ is a bisimulation on (S, α) by Proposition 2.4. But then by definition, if $f_{sat}(s) = f_{sat}(s')$, then $\langle s, s'\rangle \in f_{sat}^{-1}(\Delta_\Theta)$, and hence $s \sim_S s'$.

□

Corollary 4.1 *The logic $\mathcal{L}$ is adequate and expressive. That is, given any Moore machine (S, α) and $s, s' \in S$, the states s and s' are bisimilar iff they satisfy the same $\mathcal{L}$ formulae, i.e.* $\{\phi \mid s \in [\![\phi]\!]\} = \{\phi \mid s' \in [\![\phi]\!]\}$ *iff* $s \sim_S s'$.

Given any filter $F \in \Theta$, the subautomaton of (Θ, ζ) generated by F, which we denoted $\langle F \rangle_{(\Theta,\zeta)}$, is in fact the minimal Moore machine containing a state whose $\mathcal{L}$-theory is F.

Theorem 4.9 *Suppose (S, α) is a Moore machine with some state $s \in S$ such that $f_{sat}(s) = F$. Then $\langle F \rangle_{(\Theta,\zeta)}$ has at most $\mid S \mid$ states.*

Proof. Consider the subautomaton of (S, α) given by $\langle s \rangle_{(S,\alpha)}$, which certainly has fewer than $\mid S \mid$ states. By Theorem 3.2, $f(\langle s \rangle_{(S,\alpha)}) = \langle f_{sat}(s) \rangle_{(\Theta,\zeta)} = \langle F \rangle_{(\Theta,\zeta)}$. Hence $\langle F \rangle_{(\Theta,\zeta)}$ can have no more states than $\langle s \rangle_{(S,\alpha)}$.

□

So the final Moore machine (Θ, ζ) is a universal machine, in a very precise sense. Given any state of any Moore machine, there is a state of (Θ, ζ) which realizes the same behaviour, and moreover, the submachine generated by that state is a minimal automaton with a state displaying that behaviour.

Chapter 5

Machine Reconstruction

Traditionally in automata theory, one has a given machine and wants to understand its behaviour. In machine learning, one observes aspects of a machine's behaviour, and attempts to reconstruct the machine, or at least an approximation of it.

Suppose that for some state s of some Moore machine (S, α), we are given information about its behaviour in the form of its $\mathcal{L}$-theory, Φ. Without knowledge of any of the structure of (S, α), we can reconstruct a machine with a state bisimilar to s by generating the subautomaton $\langle \Phi \rangle_{(\Theta,\zeta)}$ of (Θ, ζ).

This subautomaton can be generated by defining transitions on Φ via $a \cdot \Phi = \{\varphi \mid a(\varphi) \in \Phi\}$, and recursively repeating this procedure on $\Psi = \delta(\Phi, a)$ for each $a \in \Sigma$ until no new states are generated. As one would expect, $o(\Psi) = \bigwedge_O At(\Psi)$.

Algorithm 5.1

```
Reconstruct($\Phi$)
    if $o(\Phi)$ is already defined
        end
    else
        define $o(\Phi) = \bigwedge At(\Phi)$
```

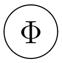
```
foreach $a \in \Sigma$
    define $\delta(\Phi, a) = a \cdot \Phi$
foreach $a \in \Sigma$
    Reconstruct$(a \cdot \Phi)$
```

Note that so long as Φ is the $\mathcal{L}$-theory of a state of some finite state Moore machine, this procedure is guaranteed to terminate by Theorem 4.13.

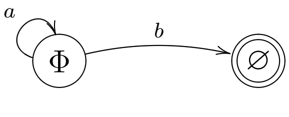
Example 5.1 *Fix* $\Sigma = \{a, b\}$*, and* $\mathcal{O} = \mathcal{P}(\{p, q\})$*, so that we are dealing with Kripke machines with a binary input alphabet and two observations. If we define a filter* $\Phi \in \Theta$ *by the grammar*

$$\Phi = \varnothing \mid p \mid a(\varphi) \mid \varphi_1 \wedge \varphi_2$$

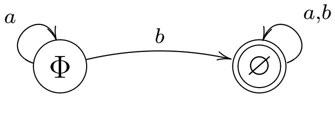
and apply Algorithm 5.1, we generate a Kripke machine from Φ *as follows.*



The algorithm proceeds by first defining $o(\Phi) = \bigwedge_{\mathcal{O}} = \bigcup\{\varnothing, p\} = \{p\}$*, then setting* $\delta(\Phi, a) = \{\varphi \mid a(\varphi) \in \Phi\} = \Phi$*, and* $\delta(\Phi, b) = \{\varphi \mid b(\varphi) \in \Phi\} = \varnothing$*.*



Now the algorithm runs again on $a \cdot \Phi = \Phi$*, but* $o(\Phi) = \{p\}$*, so this brach terminates. However, on* $b \cdot \Phi = \varnothing$*, it sets* $o(\varnothing) = \varnothing$ *and then defines* $\delta(\varnothing, a) = \{\varphi \mid a(\varphi) \in \varnothing\} = \varnothing$ *and* $\delta(\varnothing, b) = \{\varphi \mid b(\varphi) \in \varnothing\} = \varnothing$*.*



When the algorithm runs again on $\varnothing$, $o(\varnothing)$ is already defined and the algorithm then terminates, having constructed the Kripke machine above. It is easy to check that the proposition below holds for both states of this machine.

Proposition 5.1 *The $\mathcal{L}$-theory of the state corresponding to Φ is exactly Φ.*

Proof. Immediate by Lemma 4.2. □

Approximation

Our aim is to try and use this procedure in a situation where the $\mathcal{L}$-theory of a state is only partially known. In particular, we are interested in adapting this reconstruction procedure to a scenario where Φ is only a finite subset of the $\mathcal{L}$-theory of s.

However, in this situation Φ only contains partial information, so in general we want to reconstruct a machine from Φ with a state whose $\mathcal{L}$-theory is much richer than Φ. By Proposition 5.1, we know that Algorithm 5.1 will not suit this purpose.

Algorithm 5.2 below generalizes Algorithm 5.1 by incorporating a decision function $m : \mathcal{P}_{fin}(\mathcal{L}) \times \mathcal{P}_{fin}(\mathcal{L}) \to \{0, 1\}$ which decides whether two different sets of formulae represent partial information about the same state. The question of how best to define this function is still quite open, and will be discussed in more detail at the end of this section.

The new algorithm proceeds in the same manner as Algorithm 5.1, but before generating a new state, the function m is called to determine whether the partial $\mathcal{L}$-theory corresponding to any previously generated state is consistent with the partial $\mathcal{L}$-theory of the new state. If this is indeed the case, then the two states are merged. Note that we can recover Algorithm 5.1 by defining $m(\Phi, \Psi) = 1$ iff $\Phi = \Psi$.

Algorithm 5.2

```
**Reconstruct**$(\Phi)$
    **if** $o(\Phi)$ is already defined
        **end**
    **else**
        define $o(\Phi) = \bigwedge At(\Phi)$

    **foreach** $a \in \Sigma$
        **foreach** $\Psi$ such that $o(\Psi)$ is defined (*)
            **if** $m(a \cdot \Phi, \Psi) = 1$
                define $\delta(\Phi, a) = \Psi$
                **break**
        **if** $\delta(\Phi, a)$ is undefined
                define $\delta(\Phi, a) = a \cdot \Phi$
    **foreach** $a \in \Sigma$
        **Reconstruct**$(a \cdot \Phi)$
```

In this algorithm we need to put an ordering on the generated states. Suppose that pairs of generated states and their outputs are stored in a list ordered from most to least recently defined, and that the loop (*) in Algorithm 5.2 traverses this list in that order. Furthermore, each loop through the elements of Σ must use some fixed order as well. In our examples we will use alphabetical order.

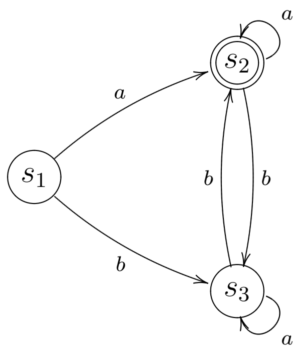
One obvious choice for m is to define $m(\Phi, \Psi) = 1$ iff $\Phi \subseteq \Psi$. This means that when defining $\delta(\Phi, a)$, if there is a state Ψ already generated whose $\mathcal{L}$-theory includes $a \cdot \Phi = \{\varphi \mid a(\varphi) \in \Phi\}$, then we define $\delta(\Phi, a) = \Psi$.

To make the algorithm most effective, we can $flatten$ the set Φ before running it. This procedure removes all conjunctions in each formula and replaces it by an equivalent list of conjunction-free formulae. Let $a(\{\phi_1, \phi_2, \cdots \phi_k\}) = \{a(\phi_1), a(\phi_2), \cdots a(\phi_k)\}$, for each $a \in \Sigma$. We define the 'deconjunction' of a single formula by induction.

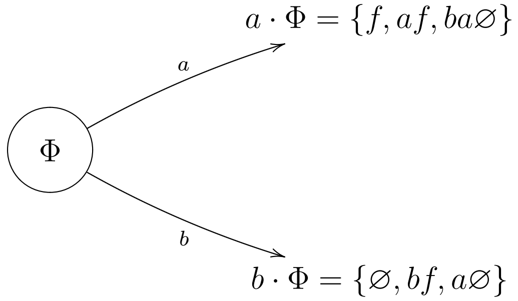
$$\begin{aligned} deconj(true) &= \{true\} \\ deconj(p) &= \{p\} \\ deconj(a(\phi)) &= a(deconj(\phi)) \\ deconj(\phi_1 \wedge \phi_2) &= deconj(\phi_1) \cup deconj(\phi_2) \end{aligned}$$

Using our inference rules and soundness, it is easy to see that $s \in [\![\phi_1 \wedge \phi_2]\!]$ iff $s \in [\![\phi_1]\!]$ and $s \in [\![\phi_2]\!]$, and also that $s \in [\![a(\phi_1 \wedge \phi_2)]\!]$ iff $s \in [\![a(\phi_1)]\!]$ and $s \in [\![a(\phi_2)]\!]$ for each $a \in \Sigma$. Hence this procedure is sound, and we define $flatten(\Phi) = \bigcup_{\varphi \in \Phi} deconj(\varphi)$.

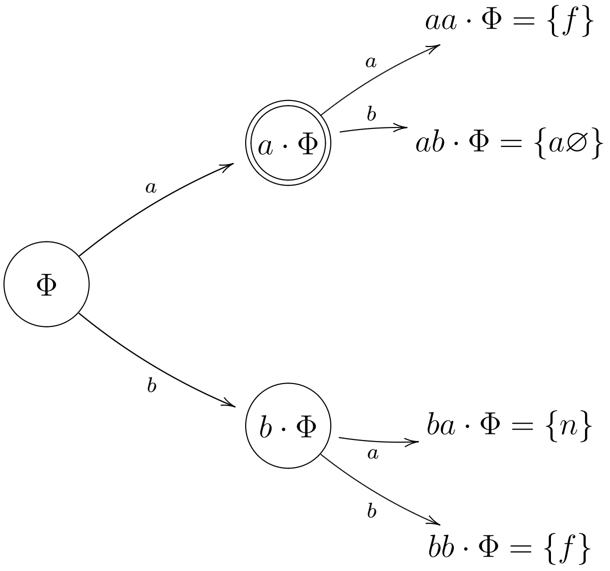
Example 5.2 *Fix $\Sigma = \{a, b\}$, $\mathcal{O} = \mathcal{P}(\{f\})^{op}$, and $m(\Phi, \Psi) = 1$ iff $\Phi \subseteq \Psi$. Consider the machine below, where we indicate that $o(s) = \{f\}$ by writing s inside a doubled circle, and $o(s) = \varnothing$ by writing s inside a single circle.*



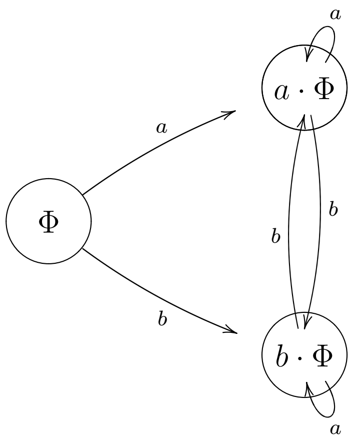
Now we identify some finite subset of the $\mathcal{L}$-theory of s_1 to run Algorithm 5.2 on. Take, for example, $\Phi = \{\varnothing, af, aaf, bbf, b\varnothing, ba\varnothing, aba\varnothing\}$.


$a \cdot \Phi = \{f, af, ba\varnothing\}$
a
Φ
b
$b \cdot \Phi = \{\varnothing, bf, a\varnothing\}$

The algorithm proceeds by generating the first two states exactly as in Algorithm 5.1. Note that since neither $a \cdot \Phi$ nor $b \cdot \Phi$ are subsets of Φ, two new states are generated.



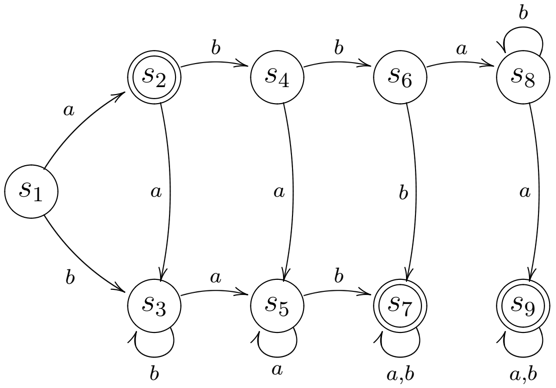
However, at this stage, since $aa \cdot \Phi$ and $bb \cdot \Phi$ are both subsets of $a \cdot \Phi$, $\delta(a \cdot \Phi, a) = a \cdot \Phi$ and $\delta(b \cdot \Phi, b) = a \cdot \Phi$. Similarly, $ab \cdot \Phi \subseteq b \cdot \Phi$, and hence $\delta(a \cdot \Phi, b) = b \cdot \Phi$. However $ba \cdot \Phi$ is a subset of both Φ and $b \cdot \Phi$, but $b \cdot \Phi$ was generated more recently, and hence $\delta(b \cdot \Phi, a) = b \cdot \Phi$.



The algorithm then terminates since the outputs of $a \cdot \Phi$ and $b \cdot \Phi$ are already defined. The final product is the machine above, which is an exact reconstruction of the original machine that Φ was extracted from.

Although in this case the algorithm provided a perfect reconstruction of the original machine, it can also fail miserably. The example below illustrates this well.

Example 5.3 *Fix $\Sigma = \{a, b\}$, $\mathcal{O} = \mathcal{P}(\{f\})^{op}$, and $m(\Phi, \Psi) = 1$ iff $\Phi \subseteq \Psi$, as in Example 5.1. However, let us consider a different subset of the $\mathcal{L}$-theory of s_1 as input to Algorithm 5.2. Let $\Phi = \{\varnothing, af, babf, abbaaf, babb\varnothing, baa\varnothing\}$, then one can check that Algorithm 5.2 will reconstruct the machine below.*



It is worth noting that the algorithm used in both examples above (Algorithm

5.2 with m defined by $m(\Phi, \Psi) = 1$ iff $\Phi \subseteq \Psi$) seems to work best when its input, Φ, contains most of the short, simple, formulae which a particular state s satisfies, and few of the longer formulae.

There are many other choices for the function m that have yet to be explored. For example, one could define m by

$$m(\Phi, \Psi) = 1 \iff \frac{\mid \Phi \triangle \Psi \mid}{min(\mid \Phi \mid, \mid \Psi \mid)} < 0.2$$

so that the symmetric difference of Φ and Ψ must be small, relative to the smaller of the two sets.

If we imagine that Φ is built up from observations about the behaviour of some machine whose internal structure is unknown, then we could let Φ be a multiset, and use redundancy to estimate how much information about the behaviour of a particular state has been observed. Such an estimate would be extremely useful, as our criterion for merging states could be made less lenient when more information is available, and more lenient when less information is available.

Conclusion

After giving a brief, but detailed review of the foundations of coalgebraic automata theory, we introduced a modified version of the logic and finality construction in [6] for Moore machines. Using the finality construction, we showed that the logic is sound, complete, adequate, and expressive. We also provided a simplified logic for the special case of Kripke machines which yields exactly the same results. In addition, we have used the structure of the final Moore machine to devise a (family of) Moore machine reconstruction algorithm(s), which have potential applications in machine learning.

It would be interesting to see what kind of analogous logical results hold for families of automata with probabilistic transitions, since typically the coalgebraic description of such machines involves an endofunctor which, for set theoretic size reasons, cannot have a final coalgebra.

The other major direction for future work is how best to refine Algorithm 5.2. In particular the choice of the function m is crucial to the performance of the algorithm, and although some functions seem to perform well on examples, it is not clear how to evaluate which choices give desirable results most consistently.

Bibliography

[1] Aczel, P. and Mendler, N., *A Final-coalgebra Theorem.* Category Theory and Computer Science, Lecture Notes In Computer Science **389**, Springer-Verlag (1989), pp. 357-365.

[2] Barr M. and Wells C., *Category Theory for Computing Science, Third Edition.* Les Publications CRM (1999).

[3] van Benthem, J., *A Note on Modal Formulae and Relational Properties.* Journal of Symbolic Logic **40** (1975), pp. 55-58.

[4] van Benthem, J., *Modal Reduction Principles.* Journal of Symbolic Logic **41** (1976), pp. 301-312.

[5] Blackburn, P., De Rijke, M. and Venema, Yde., *Modal Logic.* Cambridge Tracts in Theoretical Computer Science, Cambridge University Press (2001).

[6] Bonsangue, M., Rutten, J. and Silva, A., *Coalgebraic Logic and Synthesis of Mealy Machines.* Proceeding of FoSSaCS 2008, Lecture Notes in Computer Science **4962**, Springer-Verlag (2008), pp. 231-245.

[7] Hasuo, I., Jacobs, B. and Sokolova, A., *Generic Trace Semantics Via Coinduction.* Logical Methods In Computer Science **3**, Issue 4 (2007), pp. 1-36.

[8] Jacobs, B. and Rutten, J., *A Tutorial on (Co)Algebras and (Co)Induction.* Bulletin of the EATCS **62** (1997), pp. 222-259.

[9] Kupke, C., Kurz, A. and Pattinson, D., *Algebraic Semantics for Coalgebraic Modal Logic.* Electronic Notes in Theoretical Computer Science **106**, Elsevier (2004).

[10] Kurz, A., *Logics Admitting Final Semantics.* Lecture Notes in Computer Science **2303**, Springer-Verlag (2002), pp. 238-249.

[11] Kurz, A. and Pattinson D., *Definability, Canonical Models, Compactness for Finitary Coalgebraic Modal Logic.* Electronic Notes in Theoretical Computer Science **65**, Elsevier (2002).

[12] Milner, R., *A Calculus of Communicating Systems.* Springer-Verlag (1980).

[13] Park, D., *Title Unknown.* Slides for Bad Honnef Workshop on Semantics of Concurrency (1981).

[14] Pattinson, D., *An Introduction to the Theory of Coalgebras.* Course Notes from NASSLI 2003, Indiana University (2003).

[15] Rutten, J., *Universal Coalgebra: A Theory of Systems.* Theoretical Computing Science **249**, Elsevier (2000).

[16] Rutten, J., *Automata and Coinduction (and exercise in coalgebra).* Report SEN-R9803, CWI (1998).