STATE BASED SYSTEMS ARE COALGEBRAS

H. PETER GUMM

Abstract. Universal coalgebra is a mathematical theory of state based systems, which in many respects is dual to universal algebra. Equality must be replaced by *indistinguishability. Coinduction* replaces induction as a proof principle and maps are defined by *co-recursion.* In this (entirely self-contained) paper we give a first glimpse at the general theory and focus on some applications in Computer Science.

1. State based systems

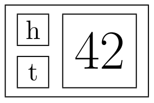
State based systems can be found everywhere in our environment – from simple appliances like alarm clocks and answering machines to sophisticated computing devices. Typically, such systems receive some input and, as a result, produce some output. In contrast to purely algebraic systems, however, the output is not only determined by the input received, but also by some modifiable "internal state". Internal states are usually not directly observable, so there may as well be different states that cannot be distinguished from the input-output behavior of the system.

A simple example of a state based system is a digital watch with several buttons and a display. Clearly, the buttons that are pressed do not by themselves determine the output - it also depends on the internal state, which might include the current time, the mode (time/alarm/stopwatch), and perhaps the information which buttons have been pressed previously.

The user of a system is normally not interested in knowing precisely, what the internal states of the system are, nor how they are represented. Of course, he might try to infer all possible states by testing various input-output combinations and attribute different behaviors to different states.

Some states might not be distinguishable by their outside behavior. It is therefore natural to define an appropriate indistinguishability relation "$\sim$" on states. One expects this relation to be an equivalence relation, and that factoring the state set by $\sim$ would yield a representation of a system with the same input-output behavior but with a minimal state set. While this is true for most of the systems that we shall consider, our definitions will be broad enough to allow for systems where indistinguishability is not transitive. Indeed, one may imagine situations where a collection of objects is observed, and it is easy to distinguish two objects that are far apart, but where objects close to each other remain indistinguishable.

1.1. **Black boxes.** We begin with the simplest possible example, a "black box", having at its front side two buttons, labeled "h" and "t", and a little display. Let us assume that the display is normally dark. Only when the button "h" is pressed, the display will show a natural number. Pressing "h" several times in a row, will not change the number displayed. However, after pressing "t" one or more times, when "h" is pressed again, we might see a new number.



Mathematically, a black box can be modeled by some set S of states together with two functions

$$h : S \to \mathbb{N}$$
$$t : S \to S.$$

1.1.1. *An Example.* Consider a black box with an eight-element state set, where the state transition function t is indicated in the following figure by arrows, and the observation function h is indicated by the labels on states.

$$\begin{tikzcd}
33 \arrow[r] & 17 \arrow[r, bend left] & 42 \arrow[l, bend left] & 17 \arrow[l] & 42 \arrow[l] & 33 \arrow[l] & 42 \arrow[r, bend left] & 17 \arrow[l, bend left]
\end{tikzcd}$$

Clearly, states with different output value are immediately distinguishable - we only have to press h and shall see different results in the display. Similarly, the two states labeled 33 are distinguishable: After pressing t, followed by h, we see a 17 in one case, and a 42 in the other case.

In contrast, all states labeled 42 are mutually indistinguishable, as are all states labeled 17. No sequence of experiments can ever lead to different outputs.

1.1.2. *Indistinguishability.* Apparently, an *indistinguishability relation* ϑ for black boxes must satisfy the following rule, which we indicate by placing the premise above, and the conclusions, separated by commas, below a horizontal line:

$$\frac{x \ \vartheta\ y}{h(x) = h(y),\ t(x)\ \vartheta\ t(y).}$$

It is clear, that there may be several indistinguishability relations; one of them is always the equality relation "=" on states. Obviously, such indistinguishability relations are closed under set-union, so there is always a largest one, which we shall denote by $\sim$. Hence we call two states s and s' *indistinguishable* iff $s \sim s'$, which is the same as saying that the pair (s, s') belongs to *some* indistinguishability relation.

1.1.3. *Streams.* As a second example consider the set $\mathbb{N}^\omega$ of all *streams* over $\mathbb{N}$. We define $h : \mathbb{N}^\omega \to \mathbb{N}$ as the *head* and $t : \mathbb{N}^\omega \to \mathbb{N}^\omega$ as the *tail* operations, i.e.

$$\begin{aligned} h(n_0, n_1, n_2, \dots) &:= n_0 \\ t(n_0, n_1, n_2, \dots) &:= (n_1, n_2, \dots)\ . \end{aligned}$$

This system is special amongst all black boxes, for we can easily verify the following proof rule, which states that two streams are equal iff they are indistinguishable:

$$\frac{x \sim y}{x = y.}$$

1.2. **Object oriented programs.** In object oriented programming, a *class* is a collection of data elements, called *objects.* All objects of one class share a common interface, consisting of a list of attributes and methods. The user can modify objects only by using their public functions (aka *methods*) and he can observe their properties only via public data fields (aka *attributes*). We shall give a simple class definition, written in the language Java, implementing a bank account:

```
class Account{
    private int amount;
    Account(){ amount=0; }
    public trans(int n){ amount += n; }
    public show(){ return amount; }
}
```

When an account is created, its integer variable `amount` is initialized to 0. This *private* variable is not directly accessible to the user, he rather has to invoke the *public* method `show` to find the account's balance. Using `trans`, he may perform a transaction, adding or subtracting money from the account.

As far as the specification of such an account is concerned, the user should insist that the following equation be satisfied for any account x:

```
x.trans(n1).trans(n2).show() == x.trans(n1 + n2).show()
```

that is, after making two transactions, one adding an amount of `n1` and a second, adding `n2`, the user should observe the same balance as if the amount `n1 + n2` had been added at once. Note that `==` is the equality relation in Java and that the dot-notation `s.m` indicates application of method `m` to state `s`.

In contrast to the first specification, the user can *not* insist on:

```
x.transact(n1).transact(n2) == x.transact(n1 + n2)
```

Both sides yield different *internal* objects, and these are not distinguishable by observations using `show`. However, the bank might later decide to augment an account object with an additional variable `accesses`, in order to keep track of how often a given account has been accessed. In that case the last equation will definitely be violated – the two sides of the equation yield different states – but for the customer, doing transactions and observing his balance, they remain *indistinguishable.* Therefore, an indistinguishability relation ϑ for bank accounts should satisfy:

$$\frac{\mathtt{x}\ \vartheta\ \mathtt{y}}{\mathtt{x.show}() == \mathtt{y.show}(),\ \ \mathtt{x.trans(n)}\ \vartheta\ \mathtt{y.trans}()}$$

1.3. **Automata.** Automata can be considered as black boxes with an additional input device, say a keyboard, where letters from an alphabet Σ are entered. The output will only tell, whether the *word*, consisting of the sequence of letters typed, has been *accepted* or not. Automata are important computing and specification devices in various branches of Computer Science.

Mathematically, a Σ-automaton is defined as a triple $\mathcal{A} = (S, \delta, E)$ where S is a set of states, $\delta : S \times \Sigma \to S$ a *transition* map and $E \subseteq S$ a set of *accepting states.* We write $s\downarrow$, if $s \in E$.

A state x is said to *accept* the *empty word* ε, just in case $x\downarrow$. It accepts a word $e \cdot w$ with first letter $e \in \Sigma$ and rest w, when $\delta(x, e)$ accepts w. Hence, an indistinguishability relation ϑ for automata should satisfy:

$$\frac{x\ \vartheta\ y}{(x\downarrow \iff y\downarrow),\ \ \forall e \in \Sigma.(\,\delta(x,e)\ \vartheta\ \delta(y,e)\,)}$$

Again, there is always a largest such indistinguishability relation $\sim$. It is known as the "Nerode congruence". Starting with a finite automaton, one obtains a minimal automaton with identical "behavior" by factoring through this Nerode congruence.

1.4. **Nondeterministic systems.** In computer applications, non-determinism can arise when various processes run at the same time under the supervision of a scheduler, as is common in all modern operating systems. The scheduler interrupts processes at previously undetermined time points and yields the computing resources to other waiting processes. Additional dimensions of non-determinism originate in communications between different processes. It can not be foreseen, when and whether messages or signals will actually arrive.

Nondeterministic systems can be modeled by *Kripke systems*. These consist of a set S of states and a binary relation $R \subseteq S \times S$ describing all possible state transitions. If R is clear from the context, one writes $s \to t$, for $(s,t) \in R$. Moreover, one has a set Φ of *atomic propositions* together with a *labeling* $v : S \to \mathbb{P}(\Phi)$, assigning to each state s the set $v(s)$ of all atomic propositions valid in this state.

We represent Kripke Structures as graphs, where the atomic propositions are attached to the nodes at which they are valid. The following picture shows two Kripke structures over the set $\Phi = \{p,q,r,s\}$ of propositions:

$$\begin{tikzcd}
 & \circ\, p \arrow[dl] \arrow[dr] & & & & \circ\, p \arrow[d] & \\
q\, \circ \arrow[d] & & \circ\, q \arrow[d] & & & \circ\, q \arrow[dl] \arrow[dr] & \\
r\, \circ & & \circ\, s & & r\, \circ & & \circ\, s
\end{tikzcd}$$

We shall be concerned with the question whether the top nodes of these two structures are distinguishable. Both are labeled with p, and they are not distinguishable by using a single series of observations, since the sequences of atomic properties encountered on paths from the top points are (p,q,r) and (p,q,s) in both cases.

In spite of this, the top nodes can be distinguished, for we need only make a single transition to get to points which are mutually distinguishable. This is so, since in the right system, we are still given a choice of transitions - one to a state labeled r, and one to a state labeled s. In the left system, there is no choice remaining after the first transaction.

Thus, in order for two states to be indistinguishable, they must have the same labels, and each choice of transition of one state must be matched by a transition of the other state, so that the new states are still indistinguishable:

$$\frac{x \;\vartheta\; y}{\begin{array}{c} v(x) = v(y), \\ \forall x'.x \to x' \implies \exists y'.(y \to y' \wedge x'\vartheta y'), \\ \forall y'.y \to y' \implies \exists x'.(x \to x' \wedge x'\vartheta y'). \end{array}}$$

The latter two symmetric conditions are most easily visualized pictorially:

$$\begin{tikzcd}
x \arrow[r, dash, bend left, "\vartheta"] \arrow[d] & y \arrow[d, dotted] & & x \arrow[r, dash, bend left, "\vartheta"] \arrow[d, dotted] & y \arrow[d] \\
x' \arrow[r, dotted, dash, bend left, "\vartheta"] & \exists y' & \text{and} & \exists x' \arrow[r, dotted, dash, bend left, "\vartheta"] & y'
\end{tikzcd}$$

2. Coalgebras

All of the above examples of state based systems, as well as many conceivable variations and generalizations turn out to be *coalgebras.* In each case, we have some set S of states and transitions which may result in one or a group of new states, or in some combination of states and outputs. In all cases, we can code the information into a single map

$$\alpha : S \to F(S),$$

where $F(S)$ is some "set theoretic construction", depending on S. The following list shows how to encode the examples discussed so far. We have added topological spaces as a further, purely mathematical example, in order to show that the concept of *coalgebra* which we are going to introduce, extends far beyond variants and generalizations of automata.

Black Boxes:	$\alpha : S \to \mathbb{N} \times S,$	$s \mapsto (h(s), t(s))$
Bank Accounts :	$\alpha : S \to \mathbb{N} \times S^{\mathbb{N}},$	$s \mapsto (show(s), n \mapsto s.trans(n))$
Automata :	$\alpha : S \to \{t, f\} \times S^{\Sigma},$	$s \mapsto (s \in E, e \mapsto \delta(s, e))$
Φ − Kripke structures:	$\alpha : S \to \mathbb{P}(\Phi) \times \mathbb{P}(S),$	$s \mapsto (v(s), \{t \mid s\,R\,t\})$
Topological Spaces :	$\alpha : S \to \mathbb{P}(\mathbb{P}(S)),$	$s \mapsto \{U \subseteq S \mid \exists \mathcal{O} \in \tau . s \in \mathcal{O} \subseteq U\}$

In the last example, a topological space (S, τ) is encoded by mapping each point to its filter of neighborhoods.

2.1. **Type functors.** The "set theoretic construction", mentioned above, determines the type of structure under consideration. To make this notion precise, by a *type*, we shall understand a *functor* $F : \mathcal{S}et \to \mathcal{S}et$ on the category of Sets. That is, F associates to every set X a new set $F(X)$, and to each map $f : X \to Y$ between sets X and Y a new map $F(f) : F(X) \to F(Y)$, so that identity maps and function composition are preserved, i.e. for arbitrary maps $f : X \to Y$ and $g : Y \to Z$ one has:

$$F(g \circ f) = F(g) \circ F(f), \quad \text{and} \quad F(id_X) = id_{F(X)}.$$

2.1.1. *Simple properties of set functors.* For all nontrivial functors $F : \mathcal{S}et \to \mathcal{S}et$, we may assume that $F(X) \neq \emptyset$ whenever $X \neq \emptyset$, for it is easy to check, that the only functor F with $F(X) = \emptyset$ for *some* $X \neq \emptyset$ is the constant functor, mapping *each* set Y to $\emptyset$.

If $X \neq \emptyset$, then a map $f : X \to Y$ is injective if and only if it has a left inverse, i.e. some map $f^- : Y \to X$ with $f^- \circ f = id_X$. Consequently, $F(f^-) \circ F(f) = id_{F(X)}$, so $F(f)$ is injective, too.

By the axiom of choice, a map $g : X \to Y$ is surjective, iff it has a right inverse. By the same argument as above, $F(g)$ is surjective, whenever g is.

2.1.2. *Examples of functors.* Kripke structures make use of the *power set functor* $\mathbb{P}(-)$, which associates to any set X the set $\mathbb{P}(X)$ of all subsets of X and to any map $f : X \to Y$ the image map $\mathbb{P}(f) : \mathbb{P}(X) \to \mathbb{P}(Y)$, mapping each $U \subseteq X$ to its image $f[U] := \{f(u) \mid u \in U\}$.

For Φ-Kripke Structures, we combine the powerset functor with the constant functor, setting $F(X) := \mathbb{P}(\Phi) \times \mathbb{P}(X)$. A map $f : X \to Y$ is sent to

$$id_{\mathbb{P}(\Phi)} \times \mathbb{P}(f) : \mathbb{P}(\Phi) \times \mathbb{P}(X) \to \mathbb{P}(\Phi) \times \mathbb{P}(Y), \text{ with } (P, U) \mapsto (P, f[U]).$$

2.2. **Coalgebras and homomorphisms.** Given a type functor F, we define a *coalgebra of type* F as a pair $\mathcal{A} = (A, \alpha_A)$, consisting of a set A and a map

$$\alpha_A : A \to F(A).$$

We refer to A as the *base set* and to α_A as the *structure map* of $\mathcal{A}$.

A *homomorphism* between coalgebras $\mathcal{A} = (A, \alpha_A)$ and $\mathcal{B} = (B, \alpha_B)$ is a map $\varphi : A \to B$ with $\alpha_B \circ \varphi = F(\varphi) \circ \alpha_A$, i.e. with the following diagram commuting:

$$\begin{tikzcd} A \arrow[r, "\varphi"] \arrow[d, "\alpha_A"'] & B \arrow[d, "\alpha_B"] \\ F(A) \arrow[r, "F(\varphi)"] & F(B) \end{tikzcd}$$

By the defining properties of a functor, the identity id_A is a homomorphism on $\mathcal{A}$, and homomorphisms are closed under function composition. Consequently, F-coalgebras with their homomorphisms form a category, which is denoted by $\mathcal{S}et_F$.

2.2.1. *Example: Σ-Automata.* In the case of Σ-automata, the homomorphism condition says that a map $\varphi : A \to B$ is a coalgebra homomorphism between automata (A, δ_A, E_A) and (B, δ_B, E_B), coded as coalgebras, iff for all $a \in A$ and for all $e \in \Sigma$:

$$a \in E_A \iff \varphi(a) \in E_B, \tag{1}$$

$$\varphi(\delta_A(a, e)) = \delta_B(\varphi(a), e). \tag{2}$$

2.2.2. *Example: Kripke Structures.* A map $\varphi : A \to B$ between Φ-Kripke Structures (A, R_A, v_A) and (B, R_B, v_B) is easily checked to be a coalgebra homomorphism, iff for all $a, a' \in A$, and all $b' \in B$:

$$v_A(a) = v_B(\varphi(a)), \tag{3}$$

$$a\, R_A\, a' \implies \varphi(a)\, R_B\, \varphi(a'), \tag{4}$$

$$\varphi(a)\, R_B\, b' \implies \exists a' \in A.(a\, R_A\, a' \wedge \varphi(a') = b'). \tag{5}$$

2.3. **Isomorphisms, homomorphic images, subcoalgebras, sums.** Whenever $\varphi : \mathcal{A} \to \mathcal{B}$ is a bijective homomorphism, then its inverse map φ^{-1} is a homomorphism too, in other words, φ is an isomorphism. This can be easily checked by calculating, using the homomorphism condition for φ:

$$\alpha_A \circ \varphi^{-1} = F(\varphi^{-1}) \circ F(\varphi) \circ \alpha_A \circ \varphi^{-1} = F(\varphi^{-1}) \circ \alpha_B \circ \varphi \circ \varphi^{-1} = F(\varphi^{-1}) \circ \alpha_B.$$

If $\varphi : \mathcal{A} \twoheadrightarrow \mathcal{B}$ is a surjective homomorphism, then the structure map α_B on B is uniquely determined by φ and the structure α_A on A. Hence $\mathcal{B}$ is called the *homomorphic image* of $\mathcal{A}$ under φ.

2.3.1. *Subcoalgebras.* If for a subset $S \subseteq A$ there exists a structure map α_S so that the canonical embedding $\subseteq^A_S$: $S \to A$ is a homomorphism between $\mathcal{S} = (S, \alpha_S)$ and $\mathcal{A}$, then such an α_S is uniquely determined. In this case, $\mathcal{S} = (S, \alpha_S)$ is called a *subcoalgebra* of $\mathcal{A}$, and we write $\mathcal{S} \leq \mathcal{A}$. By abuse of notation, the term "subcoalgebra" is also used for the carrier set S itself. In our diagrams we use hooked arrows for canonical embeddings and their F-images.

$$\begin{tikzcd} S \arrow[r, hook, "\subseteq"] \arrow[d, dashed, "\alpha_S"'] & A \arrow[d, "\alpha_A"] \\ F(S) \arrow[r, hook, "F(\subseteq)"] & F(A) \end{tikzcd}$$

2.3.2. *Sums.* The disjoint union $S := \Sigma_{i\in I}A_i$ of a family of sets $(A_i)_{i\in I}$, together with the canonical inclusions $\iota_{A_i} : A_i \to \Sigma_{i\in I}A_i$ is the *sum* of the A_i in the category $\mathcal{S}et$. This means, that for every "competitor", i.e. for every set Q with its own maps $q_i : A_i \to Q$, there is exactly one map $q : \Sigma_{i\in I}A_i \to Q$ with $q_i = q \circ \iota_{A_i}$ for all $i \in I$. (q is obtained as the "disjoint union" of the q_i).

$$\begin{tikzcd}
\Sigma_{i\in I}A_i & A_i \arrow[l, "\iota_{A_i}"] \arrow[r, dashed, "q_i"'] & Q \arrow[from=1-1, to=1-3, dotted, bend left, "q"]
\end{tikzcd}$$

Given a family $(\mathcal{A}_i)_{i\in I}$ of coalgebras, the maps $q_i := F(\iota_{A_i})\circ\alpha_{A_i} : A_i \to F(\Sigma_{i\in I}A_i)$, make the latter set a competitor for the sum. Consequently, there is a unique map $\alpha : \Sigma_{i\in I}A_i \to F(\Sigma_{i\in I}A_i)$ with $\alpha \circ \iota_{A_i} = F(\iota_{A_i}) \circ \alpha_{A_i}$. This means that there is a unique coalgebra structure α on $\Sigma_{i\in I}A_i$, for which the canonical embeddings $\iota_{A_i} : A_i \to \Sigma_{i\in I}A_i$ are homomorphisms. It is easy to verify that $\Sigma_{i\in I}\mathcal{A}_i = (\Sigma_{i\in I}A_i, \alpha)$ is in fact the sum in the category $\mathcal{S}et_F$, i.e. for every competitor coalgebra $\mathcal{Q}$ with homomorphisms $\psi_i : \mathcal{A}_i \to \mathcal{Q}$ there is precisely one homomorphism $\psi : \Sigma_{i\in I}\mathcal{A}_i \to \mathcal{Q}$ with $\psi_i = \psi \circ \iota_{A_i}$.

$$\begin{tikzcd}
\Sigma_{i\in I}A_i \arrow[d, dashed, "\alpha"'] \arrow[rr, dotted, bend left, "\psi"] & A_i \arrow[l, "\iota_{A_i}"] \arrow[r, dashed, "\psi_i"'] \arrow[d, "\alpha_{A_i}"] & Q \arrow[d, dashed, "\alpha_Q"] \\
F(\Sigma_{i\in I}A_i) \arrow[rr, dotted, bend right, "F(\psi)"'] & F(A_i) \arrow[l, "F(\iota_{A_i})"'] \arrow[r, dashed, "F(\psi_i)"] & F(Q)
\end{tikzcd}$$

2.3.3. *Pushouts.* Given coalgebras $\mathcal{A}$, $(\mathcal{A}_i)_{i\in I}$ and homomorphisms $\varphi_i : \mathcal{A} \to \mathcal{A}_i$, the *pushout* of the $(\varphi_i)_{i\in I}$ is a coalgebra $\mathcal{P}$ with homomorphisms $\psi_i : \mathcal{A}_i \to \mathcal{P}$, so that $\psi_i \circ \varphi_i = \psi_j \circ \varphi_j$ for all $i, j \in I$, and for every "competitor" coalgebra $\mathcal{Q}$ with homomorphisms $\phi_i : \mathcal{A}_i \to \mathcal{Q}$, also satisfying $\phi_i \circ \varphi_i = \phi_j \circ \varphi_j$ for all $i, j \in I$, there is exactly one homomorphism $\phi : \mathcal{P} \to \mathcal{Q}$ with $\phi \circ \psi_i = \phi_i$ for all $i \in I$.

$$\begin{tikzcd}
 & \mathcal{A}_i \arrow[dr, "\psi_i"'] \arrow[drr, dashed, "\phi_i"] & & \\
\mathcal{A} \arrow[ur, "\varphi_i"] \arrow[dr, "\varphi_j"'] & & \mathcal{P} \arrow[r, dotted, "\phi"] & \mathcal{Q} \\
 & \mathcal{A}_j \arrow[ur, "\psi_j"] \arrow[urr, dashed, "\phi_j"'] & &
\end{tikzcd}$$

Just as with sums, one checks that pushouts exist in $\mathcal{S}et_F$, and that they are formed just as in $\mathcal{S}et$. More generally, this can be said for *all* colimits in the category $\mathcal{S}et_F$. (In category theoretical language, the forgetful functor $U : \mathcal{S}et_F \to \mathcal{S}et$ *creates colimits.*)

2.4. **Homomorphisms.** The first lemma is quite technical, but we can draw from it quite a number of useful consequences. It indicates how to carry over the coalgebra structure from a given coalgebra along a surjective map, and how to restrict the structure map of a coalgebra to any of its subsets:

Lemma 2.1 (Image Construction, Restriction). (i) *Let $\mathcal{A} = (A, \alpha_A)$ be a coalgebra and $f : A \twoheadrightarrow S$ a surjective map. We can define an F-coalgebra structure α_S on S, so that for any coalgebra $\mathcal{C} = (C, \alpha_C)$ and any map $g : S \to C$ we have: If $g \circ f$ is a homomorphism, then so is g.*

(ii) *Let $\mathcal{C} = (C, \alpha_C)$ be a coalgebra and $g : S \rightarrowtail C$ an injective map. We can define an F-coalgebra structure on S, so that for any coalgebra $\mathcal{A} = (A, \alpha_A)$ and any map $f : A \to S$ we have: If $g \circ f$ is a homomorphism then so is f.*

Proof. We just indicate the proof of (i), similarly one proves (ii). Let f^- be a right inverse map of f and set $\alpha_S := F(f) \circ \alpha_A \circ f^-$. Given a coalgebra $\mathcal{C}$ and a map $g : S \to C$ where $g \circ f$ is a homomorphism, we check:

$$\begin{tikzcd}
A \arrow[r, two heads, shift left, "f"] \arrow[rr, bend left, "g\circ f"] \arrow[d, "\alpha_A"'] & S \arrow[l, dashed, shift left, "f^-"] \arrow[r, dotted, "g"'] \arrow[d, dashed] & C \arrow[d, "\alpha_C"] \\
F(A) \arrow[r, two heads, "F(f)"] \arrow[rr, bend right, "F(g\circ f)"'] & F(S) \arrow[r, dotted, "F(g)"] & C
\end{tikzcd}$$

$$\begin{aligned}
F(g) \circ \alpha_S &= F(g) \circ F(f) \circ \alpha_A \circ f^- \\
&= F(g \circ f) \circ \alpha_A \circ f^- \\
&= \alpha_C \circ g \circ f \circ f^- \\
&= \alpha_C \circ g.
\end{aligned}$$

□

2.4.1. *Surjective-injective-factorization.* Every map $f : X \to Y$ can be decomposed into a surjective map $f : X \twoheadrightarrow f[X]$, followed by the canonical embedding $\subseteq^Y_{f[X]}$. The next proposition states that the same decomposition is valid for homomorphisms, i.e. in the category $\mathcal{S}et_F$:

Proposition 2.2 (Factorization). *Every homomorphism $\varphi : \mathcal{A} \to \mathcal{B}$ can be decomposed as $\mathcal{A} \twoheadrightarrow \varphi[\mathcal{A}] \hookrightarrow \mathcal{B}$, so $\varphi[\mathcal{A}]$ is a homomorphic image of $\mathcal{A}$ and a subcoalgebra of $\mathcal{B}$.*

Proof. If φ is a homomorphism between coalgebras $\mathcal{A}$ and $\mathcal{B}$, then (i) and (ii) of the previous lemma yield two structure maps, $\alpha_{(i)}$ and $\alpha_{(ii)}$ on $\varphi[A] \subseteq B$:

$$\begin{tikzcd}
A \arrow[r, two heads, "\varphi'"'] \arrow[rr, bend left, "\varphi"] \arrow[d, "\alpha_A"'] & \varphi[A] \arrow[r, hook, "\subseteq"'] \arrow[d, dashed, shift right, "\alpha_{(ii)}"'] \arrow[d, dashed, shift left, "\alpha_{(i)}"] & B \arrow[d, "\alpha_B"] \\
F(A) \arrow[r, two heads, "F(\varphi')"] \arrow[rr, bend right, "F(\varphi)"'] & F(\varphi[A]) \arrow[r, hook, "F(\subseteq)"] & F(B)
\end{tikzcd}$$

$$\begin{aligned}
F(\subseteq^B_{\varphi[A]}) \circ \alpha_{(i)} \circ \varphi' &= \alpha_B \circ \subseteq^B_{\varphi[A]} \circ \varphi' \\
&= \alpha_B \circ \varphi \\
&= F(\varphi) \circ \alpha_A \\
&= F(\subseteq^B_{\varphi[A]}) \circ F(\varphi') \circ \alpha_A \\
&= F(\subseteq^B_{\varphi[A]}) \circ \alpha_{(ii)} \circ \varphi'.
\end{aligned}$$

We can cancel the surjective map φ' on the right and, after discarding the case $\varphi[A] = \emptyset$, also the injective map $F(\subseteq^B_{\varphi[A]})$ to the left, to obtain $\alpha_{(i)} = \alpha_{(ii)}$. □

2.4.2. *Unions of subcoalgebras.* Consider a coalgebra $\mathcal{A}$ and a family of subcoalgebras $\mathcal{S}_i \leq \mathcal{A}$. From their sum $\Sigma_{i\in I}\mathcal{S}_i$ there is a unique homomorphism φ to $\mathcal{A}$ with $\varphi \circ \iota_{S_i} = \subseteq^A_{S_i}$ for all $i \in I$. The image of φ is just $\bigcup_{i\in I} S_i$, hence we get with the help of proposition 2.2:

Lemma 2.3. *If $(S_i)_{i\in I}$ are subcoalgebras of $\mathcal{A}$, then so is $\bigcup_{i\in I} S_i$.*

Using a result of Trnkovà [Trn69], one can also prove that subcoalgebras are closed under finite intersections, hence the (carrier sets of) all subcoalgebras of $\mathcal{A}$ are the open sets of a topology on A, see [GS00b]. Conversely, by [Gum01b], every topology on a set A can be obtained this way.

2.5. **Bisimulations.** Bisimulations are the *compatible relations* between coalgebras. Their importance for computer science applications had been realized long before coalgebras were introduced in this field. Intuitively, two states of a system are *bisimilar*, if they show the same behavior. The coalgebraic definition was introduced by Aczel and Mendler[AM89]:

Definition 2.1. *A* bisimulation *between coalgebras $\mathcal{A}$ and $\mathcal{B}$ is a binary relation $R \subseteq A \times B$, on which a coalgebra structure $\rho : R \to F(R)$ can be defined, making the projections $\pi_A : R \to A$ and $\pi_B : R \to B$ into homomorphisms.*

$$\begin{tikzcd}
A \arrow[d, "\alpha_A"'] & R \arrow[l, "\pi_A"'] \arrow[r, "\pi_B"] \arrow[d, dashed, "\rho"] & B \arrow[d, "\alpha_B"] \\
F(A) & F(R) \arrow[l, "F(\pi_A)"'] \arrow[r, "F(\pi_B)"] & F(B)
\end{tikzcd}$$

Working out this definition for our earlier examples of black boxes, Σ-automata, and Φ-Kripke structures, the reader may convince himself in each case, that bisimulations are just the indistinguishability relations ϑ which we have defined earlier.

2.5.1. *Bisimulations and homomorphisms.* Every bisimulation R provides a *2-span*, i.e. a pair of homomorphisms $\mathcal{R} \to \mathcal{A}$ and $\mathcal{R} \to \mathcal{B}$ with a common domain. The converse is also true, yielding a very useful characterization of bisimulations:

Proposition 2.4. *Let $\varphi : \mathcal{P} \to \mathcal{A}$ and $\psi : \mathcal{P} \to \mathcal{B}$ be homomorphisms, then*

$$(\varphi, \psi)[P] := \{(\varphi(p), \psi(p)) \mid p \in P\}$$

is a bisimulation between $\mathcal{A}$ and $\mathcal{B}$. Each bisimulation is of this form.

Proof. $(\varphi, \psi) : P \to (\varphi, \psi)[P]$ is a surjective map, $\pi_A \circ (\varphi, \psi) = \varphi$ and $\pi_B \circ (\varphi, \psi) = \psi$ are homomorphisms. By Lemma 2.1(i), we can find a coalgebra structure on $(\varphi, \psi)[P] \subseteq A \times B$, so that both π_A and π_B become homomorphisms, hence $(\varphi, \psi)[P]$ is a bisimulation.

Obviously, for every bisimulation R between $\mathcal{A}$ and $\mathcal{B}$ has the required shape, since $R = (\pi_A, \pi_b)[R]$. □

Corollary 2.5. *A map $\psi : A \to B$ is a homomorphism between coalgebras $\mathcal{A}$ and $\mathcal{B}$ if and only if its graph $Gr(\psi) := \{(a, \psi(a)) \mid a \in A\}$ is a bisimulation.*

Proof. Setting $\varphi = id_A$ in the previous proposition yields one direction. The key to the reverse direction is the observation that the first projection π_A, restricted to the graph $Gr(\psi)$ of any function $\psi : A \to B$, is always a bijection, and that a bijective homomorphism is an isomorphism (section 2.3). Consequently, π_A^{-1} is a homomorphism, hence also $\psi = \pi_B \circ \pi_A^{-1}$. $\Box$

Given a family $(\mathcal{R}_i)_{i\in I}$ of bisimulations between $\mathcal{A}$ and $\mathcal{B}$, then we have homomorphisms $\pi_A^i : \mathcal{R}_i \to \mathcal{A}$ and $\pi_B^i : \mathcal{R}_i \to \mathcal{B}$ for each $i \in I$. Consequently, both $\mathcal{A}$ and $\mathcal{B}$ are competitors of the sum of the coalgebras $\mathcal{R}_i$, $i \in I$. Thus we get homomorphisms π_A, resp. π_B from $\Sigma_{i\in I}\mathcal{R}_i$ to $\mathcal{A}$, resp. $\mathcal{B}$. It is easy to check that the image $(\pi_A, \pi_B)[\Sigma_{i\in I}\mathcal{R}_i]$ is just the set theoretical union $\bigcup_{i\in I} R_i$, so according to proposition 2.4 we obtain:

Lemma 2.6. *The union of bisimulations is a bisimulation. Consequently, there is always a largest bisimulation between coalgebras $\mathcal{A}$ and $\mathcal{B}$.*

In many respects, it seems that bisimulations behave like 2-dimensional versions of coalgebras. However, bisimulations are not necessarily closed under finite intersections.

Definition 2.2. *The largest bisimulation between coalgebras $\mathcal{A}$ and $\mathcal{B}$ is called $\sim_{\mathcal{A},\mathcal{B}}$, or just $\sim_A$, when $\mathcal{A} = \mathcal{B}$. Elements a and b are called* bisimilar*, if $(a, b) \in \sim_{\mathcal{A},\mathcal{B}}$.*

$\sim_{\mathcal{A}}$ is always reflexive and symmetric. For most functors F, the largest bisimulation $\sim_{\mathcal{A}}$ on an F-coalgebra is also transitive. An exception can be found with the functor $(-)^3_2$, sending a set X to

$$(X)^3_2 := \{(x_0, x_1, x_2) \mid \ x_i \in X, (x_0 = x_1 \vee x_1 = x_2 \vee x_0 = x_2)\}$$

and a map $f : X \to Y$ to $(f)^3_2$ with $(f)^3_2(x_0, x_1, x_2) = (f(x_0), f(x_1), f(x_2))$.

In case that $\sim_{\mathcal{A}}$ is transitive, we may call it *observational equivalence*, in all other cases, we think that the term *indistinguishability relation* is more appropriate.

3. Terminal coalgebra semantics

For most types of coalgebras there is a prototypical model which somehow embodies all possible behaviors found somewhere in some coalgebra of this type. Its definition is as follows:

Definition 3.1. *A coalgebra $\mathcal{T}$ of type F is called* terminal*, if for every F-coalgebra $\mathcal{A}$ there is precisely one homomorphism $\tau : \mathcal{A} \to \mathcal{T}$.*

The following proposition makes precise that the terminal coalgebra, if it exists, consists exactly of all possible behaviors occurring in F-coalgebras.

Lemma 3.1. *If the terminal F-coalgebra $\mathcal{T}$ exists, then for every F-coalgebra $\mathcal{A}$ and for every $a \in A$ there exists precisely one $t \in T$ so that $a \sim_{\mathcal{A},\mathcal{T}} t$.*

Proof. Given $a \in A$, then $a \sim_{\mathcal{A},\mathcal{T}} \tau(a)$ by corollary 2.5. Suppose, there is another $t \in T$ with $a \sim_{\mathcal{A},\mathcal{T}} t$. By proposition 2.4, there is a coalgebra $\mathcal{P}$, homomorphisms $\varphi : \mathcal{P} \to \mathcal{A}$ and $\psi : \mathcal{P} \to \mathcal{T}$, and an element $p \in P$ with $\varphi(p) = a$ and $\psi(p) = t$. If $t \neq \tau(a)$ then ψ and $\tau \circ \varphi$ would be different homomorphisms from $\mathcal{P}$ to $\mathcal{T}$. $\Box$

Corollary 3.2. *The terminal F-coalgebra satisfies the following "co-inductive" proof rule:*

$$\frac{x \sim y}{x = y.}$$

The reason for this rule to be called *coinductive* is that it allows the following method for proving equality of two elements a and $b \in T$:

- Find some bisimulation R with $a\,R\,b$,
- infer $a \sim b$,
- conclude $a = b$ by terminality.

3.1. **Programming with terminal coalgebras.** Modern functional programming languages permit infinite streams as data objects. The primitives to access streams are the functions `hd` (*head*) for obtaining the first element of a stream and `tl` (*tail*) returning the rest of the stream when the first element is removed. Given a stream `r` and an element `n`, with `(n : r)`, we denote the stream `s` with `hd(s)` = `n` and `tl(s)` = `r`. The following shows an interaction with an interpreter for a modern functional programming language, such as e.g. Haskell [PH97]. The user enters his input on the line beginning with the prompt "?". The other lines contain system output.

```
?  ones = (1 : ones)
       ( 1, 1, 1, 1, ... )
?  from n  = ( n : from n+1)
?  nats = from 1
       ( 1, 2, 3, 4, ... )
?  add (n : s) (m : t)  = ( n+m : add s t )
?  add ones (from 1)  == from 2
       ( true, true, true, true, ... )
```

The programmer has defined streams `ones` and `nats`, and functions returning streams `from`, and `add`. The function `add`, for instance, accepts two streams as inputs and returns a stream whose k-th element is the sum of the corresponding elements of the argument streams.

3.1.1. *Co-Recursion.* Several questions arise, for instance: *Is there always a solution for (co)-recursive definitions of the above shape, and is it unique?* The answer is contained in the following result:

Proposition 3.3. *The coalgebra of streams is the terminal black box.*

Proof. Let $\mathcal{A}$ be a black box, that is we have maps $h : A \to \mathbb{N}$ and $t : A \to A$. We need to show that there exists precisely one coalgebra homomorphism $\varphi : \mathcal{A} \to \mathbb{N}^\omega$ where $\mathbb{N}^\omega$ is the black box of all $\mathbb{N}$-streams with structure $\texttt{hd} : \mathbb{N}^\omega \to \mathbb{N}$ and $\texttt{tl} : \mathbb{N}^\omega \to \mathbb{N}^\omega$. The homomorphism conditions require of $\varphi(a)$ for an arbitrary $a \in A$:

$$\texttt{hd}(\varphi(a)) = h(a), \tag{6}$$

$$\texttt{tl}(\varphi(a)) = \varphi(t(a)). \tag{7}$$

By induction one gets $\texttt{tl}^k(\varphi(a)) = \varphi(t^k(a))$, so the k-th element of $\varphi(a)$ is just $\texttt{hd}(\varphi(t^k(a))) = h(t^k(a))$, which proves both existence and uniqueness of φ. □

Now it is easy to see that all the streams and stream maps defined in the above program are nothing but homomorphisms from certain black boxes to the terminal black box of all $\mathbb{N}$-streams. Each one is uniquely specified by the presentation of one particular black box. Such function definitions are called *co-recursive*.

In particular, the stream `ones` is defined by the (unique homomorphism from the) one-element black box with output 1, the function `from` is defined by the black box $(\mathbb{N}, id_{\mathbb{N}}, succ)$ to $\mathbb{N}^\omega$ as the following figure demonstrates. We leave it to the reader to find the black box co-recursively defining `add`.

$$\begin{tikzcd}
\mathbb{N} \arrow[r, "id_{\mathbb{N}}"] & \mathbb{N} \\
\mathbb{N} \arrow[u, "id_{\mathbb{N}}"] \arrow[r, "\texttt{from}"] \arrow[d, "succ"'] & \mathbb{N}^\omega \arrow[u, "\texttt{hd}"'] \arrow[d, "\texttt{tl}"] \\
\mathbb{N} \arrow[r, "\texttt{from}"] & \mathbb{N}^\omega
\end{tikzcd}$$

3.2. **Proofs by Coinduction.** How can we prove a statement about streams such as e.g. `add ones (from 1) == from 2`? In our programming exercise, this was checked only for the first 4 positions. We shall show how to prove such program properties by *coinduction.* As an example, we consider the mentioned equality:

$$\texttt{add ones (from 1)} = \texttt{from 2}.$$

The first step is to find some bisimulation R containing the two elements. Choose

$$R := \{(\texttt{add ones (from n) , from (n+1)}) \mid \texttt{n} \in \mathbb{N}\}.$$

R is a bisimulation, i.e. the heads are equal and the tails are again in R:

$$\begin{aligned}
\texttt{hd(add ones (from n))} &= \texttt{hd ones} + \texttt{hd (from n)} = \texttt{1} + \texttt{n} = \texttt{hd (from (n} + \texttt{1))} \\
\texttt{tl(add ones (from n))} &= \texttt{add (tl ones)(tl (from n))} = \texttt{add ones (from(n} + \texttt{1))} \\
\texttt{tl(from(n} + \texttt{1))} &= \texttt{from((n} + \texttt{1)} + \texttt{1)}
\end{aligned}$$

In particular, $(\texttt{add ones (from 1)}, \texttt{from 2}) \in R \subseteq \sim$. Since we are in the terminal black box, we may conclude: $\texttt{add ones (from 1)} = \texttt{from 2}$.

Observe, that in this co-inductive proof we actually had to show a more general result. Such a phenomenon is, of course, also familiar from inductive proofs.

3.3. **Further terminal coalgebras.** The reader may be curious as to what terminal Σ-automata or terminal Φ-Kripke Structures might look like.

3.3.1. *The terminal* Σ*-automaton.* Given an alphabet Σ, let Σ^* denote the set of all finite words with letters from Σ. Any subset $L \subseteq \Sigma^*$ is called a *language over* Σ. Given $e \in \Sigma$ and L any language, we define its "e-derivative" as

$$L_e := \{w \in \Sigma^* \mid e \cdot w \in L\}.$$

Now we can define an automaton $\mathcal{T} = (\mathbb{P}(\Sigma^*), \delta, E)$, having as base set the set of all languages over Σ, and as transition operation the derivative, i.e. $\delta(L, e) := L_e$. A language L is defined to be an *accepting state*, if it contains the empty word, i.e.

$$L \in E \ :\Longleftrightarrow \ \varepsilon \in L.$$

We leave it to the reader to verify that this indeed defines the terminal Σ-automaton.

A bisimulation of automata is exactly an indistinguishability relation as introduced earlier. For the terminal automata this can be restated as:

$$\frac{L \ \vartheta \ M}{(\varepsilon \in L \iff \varepsilon \in M), \quad \forall e \in \Sigma.\, (L_e \ \vartheta \ M_e)}$$

Hence in order to show that two languages L_1 and L_2 are equal, we need to find a relation R, containing (L_1, L_2), and satisfying the above condition.

J. Rutten [Rut98] demonstrates how to prove regular language equations by coinduction. For instance, in order to show that for each language L,

$$(1 + L \cdot L^*) = L^*,$$

it suffices to show that $\vartheta := \{(1 + L \cdot L^*, L^*) \mid L \subseteq \Sigma^*\} \cup \{(L, L) \mid L \subseteq \Sigma^*\}$ is a bisimulation. Here, $+$, $\cdot$, and * stand for union, concatenation and "Kleene-Star" operations on languages; 0 denotes the empty language and 1 denotes the language $\{\varepsilon\}$ containing only the empty word. Checking that the above relation is in fact a bisimulation is made easy with the following rules of derivative:

$$\begin{aligned}(L+M)_e &= L_e + M_e,\\ (L\cdot M)_e &= \begin{cases} L_e \cdot M, & \text{if } \varepsilon \notin L\\ L_e\cdot M + M_e, & \text{if } \varepsilon \in L,\end{cases}\\ (L^*)_e &= L_e \cdot L^*,\\ 1_e \;=\; 0_e &= 0.\end{aligned}$$

The relevant calculation in checking that ϑ is a bisimulation consists of:

$$(1 + L\cdot L^*)_e = 0 + (L \cdot L^*)_e = \begin{cases} L_e \cdot L^*, & \text{if } \varepsilon \notin L\\ L_e \cdot L^* + L_e \cdot L^*, & \text{if } \varepsilon \in L,\end{cases} = L_e \cdot L^* = (L^*)_e.$$

3.4. **Existence of terminal coalgebras.** The terminal Φ-Kripke structure cannot exist due to the following lemma of Lambek [Lam68]. Its base set T would have to be in bijective correspondence with $\mathbb{P}(\Phi) \times \mathbb{P}(T)$, which is impossible, since $\mathbb{P}(T)$ has strictly larger cardinality than T for any set T:

Lemma 3.4. *If the terminal coalgebra exists, then its structure map is bijective.*

Proof. Suppose that $\mathcal{T} = (T, \alpha)$ is the terminal F-coalgebra, we shall construct an inverse to α. Applying F, we obtain a coalgebra $F(\mathcal{T})$ on the base set $F(T)$ with structure map $F(\alpha)$. Observe that α is at the same time a homomorphism from $\mathcal{T}$ to $\mathcal{F}(\mathcal{T})$. Since $\mathcal{T}$ is terminal, there must also be homomorphism $\beta : F(\mathcal{T}) \to \mathcal{T}$. Now $\beta \circ \alpha$ and $id_{\mathcal{T}}$ are two homomorphisms from $\mathcal{T}$ to $\mathcal{T}$, hence $id_T = \beta \circ \alpha$.

$$\begin{tikzcd} T \arrow[r, "\alpha"] \arrow[d, "\alpha"'] & F(T) \arrow[r, "\beta"] \arrow[d, "F(\alpha)"] & T \arrow[d, "\alpha"] \\ F(T) \arrow[r, "F(\alpha)"] & F(F(T)) \arrow[r, "F(\beta)"] & F(T) \end{tikzcd}$$

Applying F to this equation, and using that β is a homomorphism, we also find:

$$id_{F(T)} = F(id_T) = F(\beta \circ \alpha) = F(\beta) \circ F(\alpha) = \alpha \circ \beta.$$

□

3.5. **Bounded Functors.** The reason why there is no terminal Kripke structure lies in the uncontrolled growth of the powerset functor. Indeed, as this chapter will show, we can have a terminal Kripke Structure, if we impose a bound on the number of successors a given state is allowed to have. Mathematically, we replace the powerset functor in the definition of Kripke structures by $\mathbb{P}_\kappa(-)$, where $\mathbb{P}_\kappa(X)$, for any set X, is the set of all subsets of X with cardinality less than the cardinality κ. Of practical relevance is the case of "image finite" Φ-Kripke structures, which are coalgebras of type $\mathbb{P}(\Phi) \times \mathbb{P}_\omega(-)$.

Definition 3.2. *A functor F is called* bounded *by some cardinal κ, if for every F-coalgebra $\mathcal{A}$ and every $a \in A$ there is a subcoalgebra $\mathcal{S}$ of $\mathcal{A}$ with $a \in S$ and $|S| < \kappa$.*

This criterion is easy to check for the examples encountered so far. For black boxes, automata, programs, and Kripke Structures, we find that the set of all states *reachable* from a given state a in finitely many steps, forms a subcoalgebra. Thus (the type functor of) black boxes, programs, image finite Kripke Structures, and Σ-automata are bounded by ω, resp. $|\Sigma^*|$. In all these cases, we can construct the terminal coalgebra using the following theorem:

Theorem 3.5. *If the type F is bounded, then the terminal F-coalgebra exists.*

Proof. Let us a start with a functor F, bounded by κ, and let $(\mathcal{U}_i)_{i\in I}$ be the family of (up to isomorphism) all F-coalgebras of cardinality at most κ. Take their sum $\mathcal{S} = \Sigma_{i\in I}\mathcal{U}_i$ and let $\mathcal{T}$ be its smallest homomorphic image (the pushout of all homomorphisms with domain $\mathcal{S}$). We claim that $\mathcal{T}$ is terminal.

To check this, let $\mathcal{A} = (A, \alpha)$ be any F-coalgebra. Since F is bounded, we can find for every $a \in A$ some subcoalgebra $\mathcal{V}_a \leq \mathcal{A}$ with $a \in V_a$ and $|V_a| < \kappa$. Now each $\mathcal{V}_a$ is isomorphic to an appropriate $\mathcal{U}_i$, so we get both a homomorphism $\psi : \Sigma_{a\in A}\mathcal{V}_a \to \mathcal{S}$ and a surjective homomorphism $\varphi : \Sigma_{a\in A}\mathcal{V}_a \twoheadrightarrow \mathcal{A}$.

We form the pushout $(\mathcal{W}, \varphi', \psi')$ of φ with ψ, then φ' is onto, i.e. $\mathcal{W}$ is a homomorphic image of $\mathcal{S}$. It follows that there exists a homomorphism $\phi : \mathcal{W} \to \mathcal{T}$. Now $\phi \circ \psi'$ is a homomorphism from $\mathcal{A}$ to $\mathcal{T}$. It is routine to check uniqueness.

$$\begin{tikzcd}
& \mathcal{A} \arrow[r, dashed, "\psi'"] & \mathcal{W} \arrow[r, dotted, "\phi"] & \mathcal{T} \\
& \Sigma_{a\in A}\mathcal{V}_a \arrow[u, two heads, "\varphi"'] \arrow[r, "\psi"] & \mathcal{S} \arrow[u, dashed, two heads, "\varphi'"] \arrow[ur] & \\
\mathcal{V}_a \arrow[uur, bend left] \arrow[ur] \arrow[urr, bend right] & & &
\end{tikzcd}$$

□

4. A Birkhoff style result

If the functor F is bounded, we can take a fixed set C (the members of which we call "co-variables" or "colors") and consider the functor $F_C : \mathcal{S}et \to \mathcal{S}et$ with $X \mapsto C \times F(X)$. It is bounded, too, so its terminal coalgebra $\mathcal{T}_C$ exists.

The structure map of $\mathcal{T}_C$ combines an F-coalgebra structure $\alpha : T_C \to F(T_C)$ with a "coloring" map $\varepsilon : T_C \to C$. Being terminal as an F_C-coalgebra is the same as saying that $\mathcal{T}_C$, as an F-coalgebra, is "co-free over C" as follows:

Definition 4.1. *An Fcoalgebra $\mathcal{T}_C = (T_C, \alpha)$ together with a map $\varepsilon : T_C \to C$ is called* cofree *over the color set C, if for every F-coalgebra $\mathcal{A}$ and every set map $g : A \to C$ there is a unique homomorphic extension, i.e. a unique homomorphism $\tilde{g} : \mathcal{A} \to \mathcal{T}_C$ with $g = \varepsilon \circ \tilde{g}$.*

$$\begin{tikzcd}
& C \\
A \arrow[ur, "g"] \arrow[r, "\tilde{g}"] & T_C \arrow[u, dashed, "\varepsilon"']
\end{tikzcd}$$

If the elements of the terminal coalgebra are interpreted as behaviors, we can think of the elements of $\mathcal{T}_C$ as "behavior patterns". In fact, they turn out to play the same role that equations play in the dual theory of universal algebra, so we shall also use the term "co-equation".

For $t \in T_C$ and $\mathcal{A}$ any F-coalgebra, we say that $\mathcal{A}$ *satisfies* t, in symbols

$$\mathcal{A} \models t,$$

if for every map $g : A \to C$ we have $t \notin \tilde{g}[A]$, i.e. if every homomorphism $\varphi : \mathcal{A} \to \mathcal{T}_C$ *avoids* t. Such a definition of satisfaction by "avoidance" is not uncommon in many fields of mathematics, such as e.g. graph theory or lattice theory.

Any set E of behavior patterns, i.e. any set of co-equations, defines a class of coalgebras, namely those, satisfying each $t \in E$:

$$Mod(E) = \{\mathcal{A} \in \mathcal{S}et_F \mid \forall t \in E.\mathcal{A} \models t\}.$$

This is called the *co-equational class* defined by E.

Each co-equational class is a *covariety*, i.e. closed under taking subcoalgebras ($\mathcal{S}$), homomorphic images ($\mathcal{H}$) and sums (Σ). It is easy to see that a class $\mathcal{K}$ is a covariety, iff $\mathcal{K} = \mathcal{SH}\Sigma(\mathcal{K})$, but, more importantly, for each covariety $\mathcal{K}$ one can find a set E of co-equations defining $\mathcal{K}$. This is the coalgebraic analog to the famous theorem of Birkhoff:

Theorem 4.1. *If F is bounded, then a class $\mathcal{K}$ of F-coalgebras is a covariety if and only if it is a co-equational class.*

Proof. It is straightforward to check that a co-equational class is closed under homomorphic images and under sums. To show closure under subcoalgebras, one needs to check that every homomorphism $\varphi : \mathcal{S} \to \mathcal{T}_C$ from a subcoalgebra $\mathcal{S} \leq \mathcal{A}$ can be extended to a homomorphism $\psi : \mathcal{A} \to \mathcal{T}_C$. For this, we first extend the set map $\varepsilon \circ \varphi : S \to C$ to some set map $g : A \to C$ with $g \circ \subseteq_S^A = \varepsilon \circ \varphi$, and then choose $\psi := \tilde{g}$, the homomorphic extension g.

For the other direction, choose a color set C which is at least as large as the bound κ of the functor F. Given a covariety $\mathcal{K}$, let

$$E := \{t \in \mathcal{T}_C \mid \forall \mathcal{A} \in \mathcal{K}.\mathcal{A} \models t\}$$

be the set of all co-equations with color set C, that are true in all of $\mathcal{K}$. Clearly, $\mathcal{K} \subseteq Mod(E)$, so it remains to show $Mod(E) \subseteq \mathcal{K}$.

For every $t \in (T_C - E)$ there is a coalgebra $\mathcal{A}_t \in \mathcal{K}$ and a homomorphism $\varphi_t : \mathcal{A}_t \to \mathcal{T}_C$ so that $t \in \varphi_t[A_t]$. Hence $(T_C - E)$ is a homomorphic image of a sum of coalgebras from $\mathcal{K}$, in particular, it is a subcoalgebra of $\mathcal{T}_C$, belonging to $\mathcal{K}$.

Let now any $\mathcal{B} \in Mod(E)$ be given. For any $b \in B$, we find a subcoalgebra $\mathcal{S}_b \leq \mathcal{B}$ with $b \in S_b$ and $|S_b| < \kappa$. Choose an injective mapping $g_b : S_b \to C$, then its homomorphic extension $\tilde{g_b} : \mathcal{S}_b \to \mathcal{T}_C$ will be injective, too. Consequently, $\mathcal{S}_b$ is isomorphic to a subcoalgebra of $\mathcal{T}_C$. Since $\mathcal{S}_b \in Mod(E)$, it follows $S_b \subseteq (T_C - E)$. Hence every $\mathcal{S}_b$, and, consequently, $\mathcal{B}$ is in $\mathcal{K}$. □

This version of Birkhoff's theorem is still lacking any syntactical component. Bounded functors F can be characterized by means of surjective natural transformations η from a functor of the form $D \times (-)^M$ with appropriate fixed sets D and M(see [GS00b]). The elements of the final $D \times (-)^M$-coalgebra can be understood as infinite M-branching and D-labeled trees, so co-equations can actually be represented as equivalence classes of such trees (see [Gum01a]).

Whether these further mathematical investigations will bear fruit in computer science, remains to be seen. So far, it is well recognized that many data types are coalgebraic in nature and that co-recursive specification and verification methods and tools (see [HHJT98]) are appropriate to deal with them.

4.1. **Historical note.** The earliest papers on coalgebras defined them as straightforward dualizations of classical universal algebras [Drb69], i.e. a coalgebra was a set A with a collection of maps $\alpha_i : A \to n_i \cdot A$ into the n_i-fold direct sum of A. However, this notion was too simple minded and, most of all, it was lacking any reasonable applications. The more useful category theoretical notion, using arbitrary $\mathcal{S}et$-functors as types, was considered by Aczel and Mendler[AM89] and Barr[Bar93].

A comprehensive structure theory of universal coalgebra was formulated by J. Rutten in [Rut00] for type functors "weakly preserving pullbacks". In [Gum99a] the theory was generalized and extended to work with arbitrary type functors. The structure theoretic effect of the (weak) preservation conditions, as assumed in [AM89] and [Rut00], was characterized in [GS00a].

L. Moss has introduced in [Mos99a], see also [Mos99b], a modal logic for coalgebras whose type functor weakly preserve pullbacks. The first Birkhoff characterization was given in [Gum99b] – the syntactical side was added in [Gum01a].

References

[AM89] P. Aczel and N. Mendler, *A final coalgebra theorem*, Proceedings category theory and computer science (D.H. Pitt et al, ed.), Lecture Notes in Computer Science, Springer, 1989, pp. 357–365.

[Bar93] M. Barr, *Terminal coalgebras in well-founded set theory*, Theoretical Computer Science (1993), no. 114(2), 299–315.

[Drb69] K. Drbohlav, *On coalgebras*, Summer Session on the Theory of Ordered Sets and General Algebras, 1969, University of J.E. Puryne, Brno, pp. 81–87.

[GS00a] H.P. Gumm and T. Schröder, *Coalgebraic structure from weak limit preserving functors*, Coalgebraic Methods in Computer Science (H. Reichel, ed.), Electronic Notes in Theoretical Computer Science, vol. 33, Elsevier Science, 2000, pp. 113–133.

[GS00b] H.P. Gumm and T. Schröder, *Coalgebras of bounded type*, Tech. Report 25, Philipps-Universität Marburg, 2000, (to appear in Math. Structures in Computer Science).

[Gum99a] H.P. Gumm, *Elements of the general theory of coalgebras*, LUATCS 99, Rand Afrikaans University, Johannesburg, South Africa, 1999.

[Gum99b] H.P. Gumm, *Equational and implicational classes of coalgebras*, Theoretical Computer Science **260** (1999), 57–69.

[Gum01a] H.P. Gumm, *Birkhoffs variety theorem for coalgebras*, Contributions to General Algebra, vol. 11, J. Heyn Verlag, 2001, pp. 159–173.

[Gum01b] H.P. Gumm, *Functors for coalgebras*, Algebra Universalis (2001), no. 45, 135–147.

[HHJT98] U. Hensel, M. Huisman, B. Jacobs, and H. Tews, *Reasoning about classes in object-oriented languages: Logical models and tools*, European Symp. on Programming (Ch. Hankin, ed.), Lect. Notes in Computer Science, vol. 1381, Springer, 1998, pp. 105–121.

[Lam68] J. Lambek, *A fixpoint theorem for complete categories*, Mathematische Zeitschrift (1968), no. 103, 151161.

[Mos99a] L.S. Moss, *Coalgebraic logic*, Ann. Pure Appl. Logic **96** (1999), 277–317.

[Mos99b] L.S. Moss, *Erratum to "coalgebraic logic"*, Ann. Pure Appl. Logic **99** (1999), 241–259.

[PH97] J. Peterson and K. Hammond, *Report on the programming language Haskell: a non-strict, purely functional language, version 1.4*, Tech. Report YALEU/DCS/RR-1106, Yale University, 1997.

[Rut98] J.J.M.M. Rutten, *Automata and coinduction (an exercise in coalgebra)*, Tech. report, Centrum voor Wiskunde en Informatica, 1998.

[Rut00] J.J.M.M. Rutten, *Universal coalgebra: a theory of systems*, Theoretical Computer Science (2000), no. 249, 3–80.

[Trn69] V. Trnková, *Some properties of set functors*, Comm. Math. Univ. Carolinae (1969), no. 10,2, 323–352.

PHILIPPS-UNIVERSITÄT MARBURG, 35032 MARBURG, GERMANY
E-mail address: gumm@mathematik.uni-marburg.de