Available online at [www.sciencedirect.com](http://www.sciencedirect.com/)

[Electronic Notes in Theoretical Computer Science 298 (2013) 7–28](http://dx.doi.org/10.1016/j.entcs.2013.09.005)

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Varieties and Covarieties of Languages (Extended Abstract)

Jan Rutten[1](#_bookmark0)

*CWI and Radboud University Amsterdam, The Netherlands*

Adolfo Ballester-Bolinches[2](#_bookmark0) Enric Cosme-Ll´opez[3](#_bookmark0)

*Departament d’A`lgebra Universitat de Val`encia Val`encia, Spain*

**Abstract**

Because of the isomorphism (*X × A*) *→ X ∼*= *X →* (*A → X*), the transition structure of a deterministic automaton with state set *X* and with inputs from an alphabet *A* can be viewed both as an algebra and as a coalgebra. This algebra-coalgebra duality goes back to Arbib and Manes, who formulated it as a duality between reachability and observability, and is ultimately based on Kalman’s duality in systems theory between controllability and observability. Recently, it was used to give a new proof of Brzozowski’s minimization algorithm for deterministic automata. Here we will use the algebra-coalgebra duality of automata as a common perspective for the study of both varieties and covarieties, which are classes of automata and languages defined by equations and coequations, respectively. We make a first connection with Eilenberg’s definition of varieties of languages, which is based on the classical, algebraic notion of varieties of (transition) monoids.

*Keywords:* Automata, variety, covariety, equation, coequation, algebra, coalgebra.

# Introduction

Because of the isomorphism

(*X × A*) *→ X ∼*= *X →* (*A → X*)

the transition structure of a deterministic automaton with state set *X* and with inputs from an alphabet *A* can be viewed both as an algebra [[11](#_bookmark15)] and as a coalgebra

1 Email: [janr@cwi.nl](mailto:janr@cwi.nl)

2 Email: [Adolfo.Ballester@uv.es](mailto:Adolfo.Ballester@uv.es)

3 Email: [enric.cosme@uv.es](mailto:enric.cosme@uv.es)

1571-0661 © 2013 Elsevier B.V. Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

<http://dx.doi.org/10.1016/j.entcs.2013.09.005>

[[19](#_bookmark26),[20](#_bookmark28)]. As a consequence, both the algebraic notion of *variety* and the coalgebraic notion of *covariety* apply. In this paper, we present a preliminary version of what is to become a systematic study of varieties and covarieties of automata and of formal languages.

We will define a variety of automata (viewed as algebras) in the usual way, as a class defined by *equations* [[12](#_bookmark16)]. A covariety of automata (viewed as coalgebras) will be a class defined by *coequations* [[20](#_bookmark28)]. Varieties and covarieties of automata will then be used to define varieties and covarieties of *languages*. Our notion of a variety of languages is different from the classical definition by Eilenberg [[12](#_bookmark16),[18](#_bookmark27)], and we will make some initial observations on how the two notions are related.

The setting of our investigations will be the following picture:

*A*

*rx*  *X*¸

2,¸, (1)

2*A*¸*∗*

1

*x*

J*∗*

z

*c*

*oc*

(This diagram will be explained in more detail in Section [3](#_bookmark2).) In the middle, we have the state set *X* of a given automaton. On the left, *A∗* is the set of all words over *A*, and on the right, 2*A∗* is the set of all languages over *A*. For every choice of a *point* (initial state) *x ∈ X*, the function *rx* sends any word *w* to the state *xw* reached from *x* on input *w*. And for every choice of a *colouring* (set of final states) *c* : *X →* 2, the function *oc* sends any state to the language it accepts.

Both the pointed automata *A∗* (with the empty word as point) and *X* with point *x*, are algebras. And both the coloured automata 2*A∗* (with colouring as explained later) and *X* with colouring *c*, are coalgebras. The unique existence of the function (in fact, a homomorphism of algebras) *rx* is induced by the *initiality* of *A∗*. And the unique existence of the function (a homomorphism of coalgebras) *oc* is induced by the *ﬁnality* of 2*A∗* .

(Sets of) equations will live in the left – algebraic – part of our diagram; in short, they correspond to *quotients* of *A∗*. And (sets of) coequations live in the right – coalgebraic – part of our diagram; they will correspond to *subautomata* of 2*A∗* . As a consequence, diagram ([1](#_bookmark1)) allows us to define both varieties and covarieties, and to study their properties from a common perspective.

The *algebra-coalgebra duality* of diagram ([1](#_bookmark1)) is a modern rendering of the dual- ity between *reachability* and *observability* of automata [[2](#_bookmark10),[1](#_bookmark9)], which ultimately goes back to Kalman’s duality between controllability and observability in system theory [[14](#_bookmark22),[15](#_bookmark23)]. (See also [[7](#_bookmark17),[9](#_bookmark18)] for further categorical generalisations.)

Recently [[6](#_bookmark14),[3](#_bookmark11)], this algebra-coalgebra duality of automata was used to give a new proof and various generalisations of Brzozowski’s minimization algorithm [[8](#_bookmark19)]. The present work goes in a different direction, focusing on (co)equations and

(co)varieties. Notably, we will further refine diagram ([1](#_bookmark1)) as follows:

1

J*∗* *r*

fre¸e z )

(*X, α*

z *X*¸

¸

cofree(*X, α*)

*o*

¸

\_

2

*A*

*∗*

*x*

*c*

2,¸,

*A*

*rx oc*

(For details, see Section [5](#_bookmark5).) The new diagram includes, for every automaton *X* with transition function *α* : *X → XA*, the (pointed) automaton free(*X, α*), which represents the *largest set of equations* satisfied by (*X, α*). And, dually, we will construct a (coloured) automaton cofree(*X, α*), which represents the *smallest set of coequations* satisfied by (*X, α*).

We already mentioned above that our definition of a variety of languages is different from Eilenberg’s, which is derived from the (classical, algebraic) notion of variety of *monoids*. A first step towards an understanding of the relation between the classical and the present notion of variety consists of the – elementary but to us somewhat surprising – observation that free(*X, α*) is isomorphic to the so-called *transition monoid* of *X* (which is called the *syntactic* monoid in case *X* is minimal) [[18](#_bookmark27)]. This observation furthermore implies that the coloured automaton cofree(*X, α*) can be viewed as a dual version of the transition monoid.

Much remains to be further understood. We already mentioned the connec- tion with Eilenberg’s variety theorem. Furthermore, we would also like to relate the present algebra-coalgebra perspective to recent developments on varieties of languages, notably [[13](#_bookmark21)] and [[4](#_bookmark12),[5](#_bookmark13)]. Finally, it should be possible to generalise the present setting, along the lines of [[6](#_bookmark14),[3](#_bookmark11)], from deterministic automata to other struc- tures such as Mealy machines, weighted automata etc.

# Preliminaries

Let *A* be a finite alphabet, in all our examples fixed to *{a, b}*. We write *A∗* for the set of all finite sequences (words) over *A*. We denote the empty word by *ε* and the concatenation of two words *v* and *w* by *vw*.

For sets *X* and *Z* we define *XZ* = *{g | g* : *Z → X}*. For sets *X, Y, Z* and functions *f* : *X → Y* we define *f Z* : *XZ → Y Z* by *f Z*(*g*)= *f ◦ g*.

We define the *image* and the *kernel* of a function *f* : *X → Y* by

im(*f* )= *{y ∈ Y | ∃x ∈ X, f* (*x*)= *y }*

ker(*f* )= *{*(*x*1*, x*2) *∈ X × X | f* (*x*1)= *f* (*x*2) *}*

A *language L* over *A* is a subset *L ⊆ A∗* and we denote the set of all languages over *A* by

2*A∗* = *{L | L ⊆ A∗ }*

(ignoring here and sometimes below the difference between subsets and character- istic functions). For a language *L ⊆ A∗* and *a ∈ A* we define the *a-derivative* of *L*

by

*La* = *{v ∈ A∗ | av ∈ L}*

and we define, more generally,

*Lw* = *{v ∈ A∗ | wv ∈ L}*

We define the *initial value L*(0) of *L* by

*L*(0) = ⎧⎨ 1 if *ε ∈ L*

⎩ 0 if *ε /∈ L*

For a functor *F* : *Set → Set*, an *F-algebra* is a pair (*S, α*) consisting of a set *S*

and a function *α* : *F* (*S*) *→ S*. An *F-coalgebra* is a pair (*S, α*) with *α* : *S → F* (*S*).

We will be using the following functors:

*F* (*S*)= *SA*

*G*(*S*)= *S × A*

(2 *× F* )(*S*)=2 *× SA*

(1 + *G*)(*S*)=1 + (*S × A*)

*Automata*

An *automaton* is a pair (*X, α*) consisting of a (possibly infinite) set *X* of states and a transition function

*α* : *X → XA*

In pictures, we use the following notation:

*x a*  *y*¸



*⇔ α*(*x*)(*a*)= *y*

We will also write *xa* = *α*(*x*)(*a*) and, more generally,

*xε* = *x xwa* = *α*(*xw*)(*a*)

We observe that automata are *F-coalgebras*. Because there is, for any *A* and *X*, an isomorphism

(˜) : (*X → XA*) *→* ((*X × A*) *→ X*)

*α*˜(*x, a*)= *α*(*x*)(*a*)

automata are also *G-algebras* [[17](#_bookmark25)].

An automaton can be decorated by means of a *colouring* function

*c* : *X →* 2

using a basic set of colours 2 = *{*0*,* 1*}*. We call a state *x accepting* (or final) if

*c*(*x*) = 1, and non-accepting if *c*(*x*) = 0. We call a triple (*X, c, α*) a *coloured*

automaton. In pictures, we use a double circle to indicate that a state is accepting. For instance, in the following automaton

*a*



*x* ¸¸

v˛

*y a*

,

¸

*b*

*b*

the state *x* is accepting and the state *y* is not.

By pairing the functions *c* and *α*, we see that coloured automata are (2 *× F* )- coalgebras:

*⟨c, α⟩* : *X →* 2 *× XA*

An automaton can also have an *initial state x ∈ X*, here represented by a function

*x* :1 *→ X*

where 1 = *{*0*}*. We call a triple (*X, x, α*) a *pointed* automaton. By pairing the functions *x* and *α*˜, we see that pointed automata are (1 + *G*)-algebras:

[*x, α*˜]: (1 + (*X × A*)) *→ X*

We call a 4-tuple (*X, x, c, α*)a *pointed and coloured automaton*. We could depict it by either of the two following diagrams

1

*x*

*c*

z

2 ¸

1

*c*  2 ¸

z

*x*

*X X*,,

*α α*˜

J

*XA X × A*

We will be using the diagram on the left, which is just a matter of personal prefer- ence.

We observe further that pointed and coloured automata are simply called *au- tomata* in most of the literature on automata theory. A pointed and coloured automaton (*X, x, c, α*) is neither an algebra nor a coalgebra – because of *c* and *x*, respectively – which can be a cause of fascination and confusion alike.

*Homomorphisms, subautomata, bisimulations*

A function *h* : *X → Y* is a *homomorphism* between automata (*X, α*) and (*Y, β*) if it makes the following diagram commute:

*X*  *Y* ¸

*h*

*α β*

J J

*XA*  *Y* ¸*A*

*hA*

A homomorphism of pointed automata (*X, x, α*) and (*Y, y, β*) and of coloured au- tomata (*X, c, α*) and (*Y, d, β*) moreover respects initial values and colours, respec- tively:

*x*

*X*  *Y* ¸

*h*

2,¸,

*d*

1

*y*

J

z

*c*

*X*  *Y* ¸

*h*

If in the diagrams above *X ⊆ Y* , and (i) *h* is subset inclusion

*h* : *X ⊆ Y*

(and, moreover (ii) *x* = *y* or (iii) *c* = *d*), then we call *X* a (i) *subautomaton* of *Y*

(respectively (ii) *pointed* and (iii) *coloured subautomaton*).

For an automaton (*X, α*) and *x ∈ X*, the *subautomaton generated by x*, denoted

by

*⟨x⟩⊆ X*

consists of the smallest subset of *X* that contains *x* and is closed under transitions. We call a relation *R ⊆ X×Y* a *bisimulation of automata* if for all (*x, y*) *∈ X×Y* ,

(*x, y*) *∈ R ⇒ ∀a ∈ A,* (*xa, ya*) *∈ R*

(where *xa* = *σ*(*x*)(*a*) and *ya* = *τ* (*y*)(*a*)). For pointed automata (*X, x, α*) and (*Y, y, β*), *R* is a *pointed bisimulation* if, moreover, (*x, y*) *∈ R*. And for coloured automata (*X, x, α*) and (*Y, y, β*), *R* is a *coloured bisimulation* if, moreover, for all (*x, y*) *∈ X × Y* ,

(*x, y*) *∈ R ⇒ c*(*x*)= *d*(*y*)

A bisimulation *E ⊆ X × X* is called a bisimulation *on X*. If *E* is an equiv- alence relation then we call it a *bisimulation equivalence*. The quotient map of a bisimulation equivalence on *X* is a homomorphism of automata:

*X*  *q*  *X*¸*/E*

*α* [*α*]

J J

*XA*  (*X*¸*/E*)*A*

*qA*

with the obvious definitions of *X/E*, *q* and [*α*]. If the equivalence *E* is a pointed bisimulation on (*X, x, α*) or a coloured bisimulation on (*X, c, α*), then we moreover have, respectively,

1

[*x*]

J

*X*tx

¸*/E*

*x*

*X h*

2,¸,

[*c*]

*c*

*X*  *X*¸*/E*

*h*

with, again, the obvious definitions of [*x*] and [*c*].

For a homomorphism *h* : *X → Y* , ker(*h*) is a bisimulation equivalence on *X* and im(*h*) is a subautomaton of *Y* . Any homomorphism *h* factors through quotient and inclusion homomorphisms as follows:

*X*

*α*

*h*

[*α*]

*q*

*X*¸*/*ker(*h*)

*i*

*X*

J *qA*

*A*

¸

(*X/*ker(*h*))

J

*A*

*iA*

¸

¸

*Y*

J

*A*

z*Y* ¸˛

*β*

*hA*

Note that *X/*ker(*h*) *∼*= im(*h*). Because *q* is surjective and *i* is injective, the pair (*q, i*) is called an *epi-mono factorisation* of *h*.

*Congruence relations*

A *right congruence* is an equivalence relation *E ⊆ A∗ ×A∗* such that, for all (*v, w*) *∈*

*A∗ × A∗*,

(*v, w*) *∈ E ⇒ ∀u ∈ A∗,* (*vu, wu*) *∈ E*

A *left congruence* is an equivalence relation *E ⊆ A∗ × A∗* such that, for all (*v, w*) *∈*

*A∗ × A∗*,

(*v, w*) *∈ E ⇒ ∀u ∈ A∗,* (*uv, uw*) *∈ E*

We call *E* a *congruence* if it is both a right and a left congruence. Note that *E* is a right congruence iff it is a bisimulation equivalence on (*A∗, σ*).

*Products and coproducts of automata*

Automata (are both *G*-algebras and *F* -coalgebras and hence) have both products and coproducts, as follows.

* The *product* of two automata (*X, α*) and (*Y, β*) is given by (*X×Y, γ*) where *X×Y*

is the Cartesian product and where

*γ* : (*X × Y* ) *→* (*X × Y* )*A γ*((*x, y*))(*a*)= ( *α*(*x*)(*a*)*, β*(*y*)(*a*))

* The *coproduct* (or: sum) of two automata (*X, α*) and (*Y, β*) is given by (*X* + *Y, γ*) where *X* + *Y* is the disjoint union and where

*γ* : (*X* + *Y* ) *→* (*X* + *Y* )*A γ*(*z*)(*a*)= ⎧⎨ *α*(*z*)(*a*) if *z ∈ X*

⎩ *β*(*z*)(*a*) if *z ∈ Y*

Pointed automata (are (1 + *G*)-algebras and hence) have products, as fol- lows. The product of two pointed automata (*X, x, α*) and (*Y, y, β*) is given by (*X × Y,* (*x, y*)*, γ*) with (*X × Y, γ*) as above and with initial state

(*x, y*):1 *→ X × Y*

Coloured automata (are (2 *× F* )-coalgebras and hence) have coproducts, as fol- lows. The coproduct of two coloured automata (*X, c, α*) and (*Y, d, β*) is given by (*X* + *Y,* [*c, d*]*, γ*) with (*X* + *Y, γ*) as above and with colouring function

[*c, d*]: (*X* + *Y* ) *→* 2 [*c, d*](*z*)= ⎧⎨ *c*(*z*) if *z ∈ X*

⎩ *d*(*z*) if *z ∈ Y*

All of the above binary (co)products can be easily generalised to (co)products of arbitrary families of automata.

# Setting the scene

The set *A∗* forms a pointed automaton (*A∗, ε, σ*) with initial state *ε* and transition function *σ* defined by

*σ* : *A∗ →* (*A∗*)*A σ*(*w*)(*a*)= *wa*

It is *initial* in the following sense: for any given automaton (*X, α*), every choice of initial state *x* :1 *→ X* induces a unique function *rx* : *A∗ → X*, given by *rx*(*w*)= *xw*, that makes the following diagram commute:

1

*x*

J*∗*

z

*ε*

*A rx*  *X*¸

*σ α*

J J

(*A∗*)*A*  *X*¸*A*

(*rx*)*A*

This property makes (*A∗, ε, σ*) an *initial* (1 + *G*)*-algebra*. Equivalently, the automa- ton (*A∗, σ*) isa *G-algebra that is free on the set* 1. The function *rx* maps a word *w* to the state *xw* reached from the initial state *x* on input *w* and is therefore called the *reachability* map for (*X, x, α*).

The set 2*A∗* of languages forms a coloured automaton (2*A∗ , ε*?*,τ* ) with colouring function *ε*? defined by

*ε*? : 2*A∗ →* 2 *ε*?(*L*)= *L*(0)

and transition function *τ* defined by

*τ* : 2*A∗ →* (2*A∗* )*A τ* (*L*)(*a*)= *L*

*a*

It is *ﬁnal* in the following sense: for any given automaton (*X, α*), every choice of

colouring function *c* : *X →* 2 induces a unique function *oc*

: *X →* 2*A∗* , given by

*oc*(*x*)= *{w | c*(*xw*)= 1 *}*, that makes the following diagram commute:

2,¸,

*c*

*ε*?

*X*  2*A*¸*∗*

*oc*

*α τ*

J *∗*J

*XA*  (2¸*A* )*A*

(*oc*)*A*

This property makes (2*A∗ , ε*?*,τ* )a *ﬁnal* (2*×F* )*-coalgebra*. Equivalently, the automa-

ton (2*A∗ ,τ* ) is an *F-coalgebra that is cofree on the set* 2. The function *o*

*c*

maps a

state *x* to the language *oc*(*x*) accepted by *x*. Since the language *oc*(*x*) can be viewed as the observable behaviour of *x*, the function *oc* is called the *observability* map.

*The scene*

Summarizing, we have set the following scene for our investigations:

1

*x*

J*∗*

z

*c*

*ε*

*A rx*  *X*¸

*σ α*

J J

2,¸,

*ε*?

2*A*¸*∗*

*oc*

*τ*

*∗*J

(2)

(*A∗*)*A*  *X*¸*A*  (2¸*A* )*A*

(*rx*)*A* (*oc*)*A*

If the reachability map *rx* is *surjective* then we call (*X, x, α*) *reachable*. If the observability map *oc* is *injective* then we call (*X, c, α*) *observable*. And if *rx* is surjective *and oc* is injective then we call (*X, x, c, α*) (reachable and observable, or:) *minimal*.

For a given language *L ∈* 2*A∗* , there is the following variation of the picture above:

1

*L*

J*∗*

*h*

2*A*¸

vz

*∗*

*L*

*ε*

*A*

*ε*?

J

˛

2

where the lower *L* is in fact the characteristic function of *L ⊆ A∗*, and where the homomorphism *h* satisfies *h* = *rL* = *oL* and *h*(*w*)= *Lw*. As a consequence, we have *h*(*v*)= *h*(*w*) iff

*∀u ∈ A∗, vu ∈ L ⇔ wu ∈ L*

which we recognise as the celebrated *Myhill-Nerode* equivalence. A *minimal au-*

*tomaton accepting L* is now obtained by the epi-mono factorisation of *h*:

*ε*

1

*L*

*x*

J*∗* *q*

*A /*ker(*h*)

¸

*∗*

zz

*i*

2 ¸

z

*A*

*∗*

*c*

*L*

*A*

*ε*?

J¸

2

where *x* = *q◦ε* and *c* = *ε*? *◦i*. This minimal automaton is unique up-to isomorphism because epi-mono factorisations are. And because *A∗/*ker(*h*) *∼*= im(*h*), it is equal to

*⟨L⟩⊆* 2*A∗*

that is, the subautomaton of (2*A∗ , ε*?) generated by *L*.

In conclusion of this section, we observe that *⟨L⟩* is finite iff the language *L* is *rational*. This fact is a version [[8](#_bookmark19),[10](#_bookmark20)] of Kleene’s correspondence between finite automata and rational languages [[16](#_bookmark24)].

# Equations and coequations

We will be referring to the situation of ([2](#_bookmark3)).

**Definition 4.1** [equations] A *set of equations* is a bisimulation equivalence relation *E ⊆ A∗ × A∗* on the automaton (*A∗, σ*). We define (*X, x, α*) *|*= *E* – and say: *the pointed automaton* (*X, x, α*) *satisﬁes E* – by

(*X, x, α*) *|*= *E ⇔ ∀*(*v, w*) *∈ E, xv* = *xw*

Because

*∀*(*v, w*) *∈ E, xv* = *xw ⇔ E ⊆* ker(*rx*)

we have, equivalently, that (*X, x, α*) *|*= *E* iff the reachability map *rx* factors through

*A∗/E*:

1

J*∗* *q*

*A*

¸*/E*

*∗*

z

*h*

z

¸

*X*

*x*

[*ε*]

*ε*

*A*

*rx*

where the homomorphisms (of pointed automata) *q* and *h* are given by

*q*(*w*)= [*w*] *h*([*w*]) = *rx*(*w*)

We define (*X, α*) *|*= *E* – and say: *the automaton* (*X, α*) *satisﬁes E* – by (*X, α*) *|*= *E ⇔ ∀x* :1 *→ X,* (*X, x, α*) *|*= *E*

*⇔ ∀x ∈ X, ∀*(*v, w*) *∈ E, xv* = *xw*

*2*

Note that we consider *sets* of equations *E* and that (*v, w*) *∈ E* implies (*vu, wu*) *∈ E*, for all *v, w, u ∈ A∗*, because *E* is – by definition – a bisimulation relation on (*A∗, σ*). Still we shall sometimes consider also *single* equations (*v, w*) *∈ A∗ × A∗* and use the following shorthand:

(*X, x, α*) *|*= *v* = *w ⇔* (*X, x, α*) *|*= *Ev*=*w*

where *Ev*=*w* is defined as the smallest bisimulation equivalence on *A∗* containing (*v, w*). We shall use also variations such as

(*X, x, α*) *|*= *{v* = *w, t* = *u} ⇔* (*X, x, α*) *|*= *v* = *w ∧* (*X, x, α*) *|*= *t* = *u*

**Definition 4.2** [coequations]

A *set of coequations* is a subautomaton *D ⊆* 2*A∗* of the automaton (2*A∗ ,τ* ). We define (*X, c, α*) *|*= *D* – and say: *the coloured automaton* (*X, c, α*) *satisﬁes D* – by

(*X, c, α*) *|*= *D ⇔ ∀x ∈ X, oc*(*x*) *∈ D*

Because

*∀x ∈ X, oc*(*x*) *∈ D ⇔* im(*oc*) *⊆ D*

we have, equivalently, that (*X, c, α*) *|*= *D* iff the observability map *oc* factors through

*D*:

2,¸,

*c*

*ε*?

*ε*?

*X*  *h*  *D*¸ *i*  2*A*¸*∗*

*oc*

where the homomorphisms (of coloured automata) *h* and *i* are given by

*h*(*x*)= *oc*(*x*) *i*(*L*)= *L*

We define (*X, α*) *|*= *D* – and say: *the automaton* (*X, α*) *satisﬁes D* – by (*X, α*) *|*= *D ⇔ ∀c* : *X →* 2*,* (*X, c, α*) *|*= *D*

*⇔ ∀c* : *X →* 2*, ∀x ∈ X, oc*(*x*) *∈ D*

*2*

**Example 4.3** We consider the automaton (*Z, γ*) defined by the following diagram:

(*Z, γ*) =

*a*

*b*



*x* ¸¸

v˛

*y a*

,

¸

*b*

Here are some examples of equations: (*Z, x, γ*) *|*= *{b* = *ε, ab* = *ε, aa* = *a}*

(*Z, y, γ*) *|*= *{a* = *ε, ba* = *ε, bb* = *b}*

Taking the intersection of the (bisimulation equivalences generated by) these sets, we obtain that

(*Z, γ*) *|*= *{aa* = *a, bb* = *b, ab* = *b, ba* = *a}*

The above set of equations or, again more precisely, the bisimulation equivalence relation on (*A∗, σ*) generated by it, is the largest set of equations satisfied by (*Z, γ*). For examples of coequations, we consider the following 2 (out of all 4 possible)

coloured versions of (*Z, γ*):

(*Z, c, γ*)= *b*

*a*

*b*



*x* ¸¸

v˛

*y a*

,

¸

*a*

(*Z, d, γ*)= *b*



*x* ¸¸

v˛

*y a*

,

¸

*b*

(Thus *c*(*x*) = 1, *c*(*y*) = 0, *d*(*x*) = 0 and *d*(*y*) = 1.) The observability mappings *oc*

and *od* map these automata to

im(*oc*)=



j

(*a∗b*)*∗* ¸\_

*a*

z )

(*a∗b*)+

*b*

*b*

*a*

im(*od*)=

*b*

*a*

*b*

It follows that



)

(*b∗a*)+ ¸¸

*a*

z\_ j

(*b∗a*)*∗*

(*Z, c, γ*) *|*= *{*(*a∗b*)*∗,* (*a∗b*)+*}* (*Z, d, γ*) *|*= *{*(*b∗a*)*∗,* (*b∗a*)+*}*

*2*

# Free and cofree automata

Let (*X, α*) be an arbitrary automaton. We show how to construct an automaton that corresponds to the *largest set of equations* satisfied by (*X, α*). And, dually, we construct an automaton that corresponds to the *smallest set of coequations* satisfied by (*X, α*). For notational convenience, we assume *X* to be finite but nothing will depend on that assumption.

**Definition 5.1** [free automaton, Eq(*X, α*)] Let *X* = *{x*1*,..., xn}* be the set of states of a finite automaton (*X, α*). We define a pointed automaton free(*X, α*) in two steps, as follows:

* 1. First, we take the product of the *n* pointed automata (*X, xi, α*) that we obtain by letting the initial element *xi* range over *X*. This yields a pointed automaton (Π*X, x*¯*, α*¯) with

Π*X* =

*x*:1*→X*

*Xx ∼*= *Xn*

(where *Xx* = *X*), with *x*¯ = (*x*1*,..., xn*), and with *α*¯ : Π*X →* (Π*X*)*A* defined by

*α*¯(*y*1*,..., yn*)(*a*)= ((*y*1)*a,...,* (*yn*)*a*)

* 1. Next we define (free(*X, α*)*, x*¯*, α*¯) by free(*X, α*)= im(*rx*¯), where *rx*¯ is the reach- ability map for (Π*X, x*¯*, α*¯):

1

J*∗* *r*

¸

free(*X, α*)

zz

*i*

z ¸*n*

\_*X*

*x*¯

*x*¯

*ε*

*A*

*rx*¯

Furthermore, we define the following set of equations:

Eq(*X, α*)= ker(*r*)

where *r* is the reachability map for (free(*X, α*)*, x*¯*, α*¯). *2*

Note that

free(*X, α*) *∼*= *A∗/*Eq(*X, α*)

**Definition 5.2** [cofree automaton, coEq(*X, α*)] Let *X* = *{x*1*,..., xn}* be the set of states of a finite automaton (*X, α*). We define a coloured automaton cofree(*X, α*) in two steps, as follows:

1. First, we take the coproduct of the 2*n* coloured automata (*X, c, α*) that we obtain by letting *c* range over the set *X →* 2 of all colouring functions. This yields a coloured automaton (Σ*X, c*ˆ*, α*ˆ) with

Σ*X* = Σ *Xc*

*c*:*X→*2

(where *Xc* = *X*), and with *c*ˆ and *α*ˆ defined component-wise.

1. Next we define (cofree(*X, α*)*,* [*c*ˆ]*,* [*α*ˆ]) by cofree(*X, α*) = Σ*X/*ker(*oc*ˆ), where *oc*ˆ

is the observability map for (Σ*X, c*ˆ*, α*ˆ):

2,¸,

*c*ˆ

[*c*ˆ]

Σ*X*

*q*

¸

cofree(*X, α*)

*o*

¸

¸

2

*A*

*∗*

*ε*?

*oc*ˆ

and where [*c*ˆ] and [*α*ˆ] are the extensions of *c*ˆ and *α*ˆ to equivalence classes.

Furthermore, we define

coEq(*X, α*)= im(*o*)

where *o* is the observability map for (cofree(*X, α*)*,* [*c*ˆ]*,* [*α*]). *2*

Note that

cofree(*X, α*) *∼*= coEq(*X, α*)

**Theorem 5.3** *The set* Eq(*X, α*) *is the* largest *set of equations satisﬁed by* (*X, α*)*. The set* coEq(*X, α*) *is the* smallest *set of coequations satisﬁed by* (*X, α*)*.* *2*

**Example 5.4** [Example [4.3](#_bookmark4) continued] We consider our previous example

(*Z, γ*) =

*a*

*b*



*x* ¸¸

v˛

*y*

,

¸

*a*

*b*

The product of (*Z, x, γ*) and (*Z, y, γ*) is:

*a*



*a*

¸*y*) ¸,

(*y,*

j

*a*

¸,

(*x, y*)

*a*

*b*

(*y, x*)

*b*

(*x*¸*, x*) ¸,*b*

J

,,

(Π*Z,* (*x, y*)*, γ*¯) =

*b*

Taking im(*r*(*x,y*)) yields the part that is reachable from (*x, y*):

*a*



j

*a*  ¸

(*y, y*)

¸,

(free(*Z, γ*)*,* (*x, y*)*, γ*¯) =

(*x, y*) *a b*

J

*b*  (*x*¸*, x*)

,,

*b*

The set Eq(*Z, γ*) is defined as ker(*r*(*x,y*)), and consists of (the smallest bisimulation equivalence on (*A∗, σ*) generated by)

Eq(*Z, γ*)= *{aa* = *a, bb* = *b, ab* = *b, ba* = *a}*

This is the largest set of equations satisfied by (*Z, γ*).

Next we turn to coequations. The coproduct of all 4 coloured versions of (*Z, γ*)

is

(Σ*Z, c*ˆ*, γ*ˆ) =

*b*

*b*

*a*

*b a*



¸

*x*

1

¸˛

z*y*\_

1 ¸¸

*a*



¸

*x*3

¸˛

z*y*\_

3 ¸¸

*a*

*a*

*b*



¸

*x*

2

¸˛

z*y*\_

2 ¸¸

*a*

*b a*



¸

*x*4

¸˛

z*y*\_

4 ¸¸

*a*

*b*

*b b*

The observability map *o* : Σ*Z →* 2*A∗* is given by

*c*ˆ

*oc*ˆ(*x*1) *oc*ˆ(*y*1) *oc*ˆ(*x*2) *oc*ˆ(*y*2) *oc*ˆ(*x*3) *oc*ˆ(*y*3) *oc*ˆ(*x*4) *oc*ˆ(*y*4)

*∅ ∅* (*a∗b*)*∗* (*a∗b*)+ (*b∗a*)+ (*b∗a*)*∗*

*A∗*

*A∗*

Computing the quotient Σ*Z/*ker(*oc*ˆ) yields:

(cofree(*Z, γ*)*,* [*c*ˆ]*,* [*γ*ˆ]) =

*a,b b a*



j

*{x*1*, y*1*}*



j *a*

*{x*2*}* ¸¸

z˛ j

*{y*2*}*

*b*

*a,b b a*



j

*{x*4*, y*4*}*



j *a*

*{x*3*}* ¸¸

z˛ j

*{y*3*}*

*b*

The image of this automaton under the reachability map *o* : cofree(*Z, γ*) *→* 2*A∗* is

coEq(*Z, γ*)=

*a,b*

et

*∅*

*a,b*



*A∗*

%r

*b a*

*b*



(*a∗b*)*∗* ¸\_

j

*a*

z\_ )

(*a∗b*)+

*b a*



(*b∗a*)+ ¸\_

)

*a*

z\_ j

(*b∗a*)*∗*

(3)

*b*

This is the smallest set of coequations satisfied by (*Z, γ*). *2*

Summarizing the present section, we have obtained, for every automaton (*X, α*), the following refinement of ([2](#_bookmark3)):

1 *x*

*ε x*¯

*c*

[*c*ˆ]

2,¸,

*ε*?

J vz

*∗*

v˛ *A∗*

*A r* fre¸e(*X, α*)

*σ α*¯

J J

*X*¸

*α*

J

co¸free(*X, α*)

[*α*ˆ]

J

*o*  2 ¸

*τ*

*∗*J

(*A∗*)*A*  fre¸e(*X, α*)*A*  *X*¸*A*  co¸free(*X, α*)*A*  (2¸*A* )*A*

where *x* ranges over the elements of *X* and *c* ranges over all possible colourings of

*X*. The free and cofree automata represent the largest set of equations and the smallest set of coequations satisfied by (*X, α*):

Eq(*X, α*)= ker(*r*) coEq(*X, α*)= im(*o*)

Note that the free and cofree automata are constructed for the automaton (*X, α*),

without point and without colouring. In conclusion, let us mention again that all of the above easily generalises to *inﬁnite X*.

# Varieties and covarieties

We define varieties and covarieties by means of equations and coequations, first for automata and next for languages.

**Definition 6.1** [variety of automata] For every set *E* of equations we define the

*variety VE* by

*VE* = *{* (*X, α*) *|* (*X, α*) *|*= *E }*

*2*

**Definition 6.2** [covariety of automata] For every set *D* of coequations we define the *covariety CD* by

*CD* = *{* (*X, α*) *|* (*X, α*) *|*= *D }*

*2*

Every variety of automata defines a set of languages, which we will again call a variety. Dually, every covariety of automata defines a set of languages , which we will again call a covariety.

**Definition 6.3** [variety and covariety of languages] Let *VE* be a variety of au- tomata. We define the *variety of languages L*(*VE*) by

*L*(*V* )= *{ L ∈* 2*A∗ | ⟨L⟩∈ V }*

*E E*

(where *⟨L⟩* is the subautomaton of (2*A∗ ,τ* ) generated by *L*). Dually, let *C* be a covariety of automata. We define the *covariety of languages L*(*CD*) by

*D*

*L*(*C* )= *{ L ∈* 2*A∗ | ⟨L⟩∈ C }*

*D D*

*2*

**Proposition 6.4** *Every variety VE is closed under the formation of* subautomata*,* homomorphic images*, and* products*.* *2*

**Proposition 6.5** *Every covariety CD is closed under the formation of* subau- tomata*,* homomorphic images*, and* coproducts*.* *2*

**Proposition 6.6** *A covariety CD is generally* not *closed under products.*

**Proof.** We give an example of a covariety that is not closed under products. We recall from Example [5.4](#_bookmark6) the automaton

(*Z, γ*) =

*a*

*b*



*x* ¸¸

v˛

*y*

,

¸

*a*

*b*

We saw that (*Z, γ*) *|*= *D*, with *D* = coEq(*Z, γ*) as in ([3](#_bookmark7)). The product of (*Z, γ*) with itself is

*a*



*a*

¸*y*) ¸,

(*y,*

j

*a*

¸,

(*x, y*)

*a*

*b*

(*y, x*)

*b*

(*x*¸*, x*) ¸,*b*

J

,,

(*Z*2*, γ*¯) =

*b*

We define a colouring *c* : *Z*2 *→* 2 by

*c*

*c*((*x, y*)) *c*((*y, y*)) *c*((*x, x*)) *c*((*y, x*))

0

1

1

0

This colouring *c* induces the observability map *o* : *Z*2 *→* 2*A∗* , given by

*oc*((*x, y*)) *oc*((*y, y*)) *oc*((*x, x*)) *oc*((*y, x*)) *A*+ *A∗ A∗ A*+

Because *A*+ */∈ D*, the automaton (*Z*2*, γ*¯) *|*= *D*. Thus *CD* is not closed under products. *2*

**Corollary 6.7** Not *every covariety CD is also a variety.* *2*

Here are some elementary properties of (co)equations and (covarieties).

**Proposition 6.8** *For every set of equations E ⊆ A∗ × A∗,*

*L*(*V* )= *{L ∈* 2*A∗ | ∀*(*v, w*) *∈ E*˜*, L* = *L }*

*E v w*

*where E*˜ *is the smallest congruence relation containing E.* *2*

**Theorem 6.9 (on equations and varieties)** *Let E ⊆ A∗ × A∗ be a set of equa- tions. The following statements are equivalent:*

1. *E is a congruence*
2. *E* = Eq(*X, α*) *for some automaton* (*X, α*)
3. (*A∗/E,* [*σ*]) *|*= *E*
4. Eq(*A∗/E,* [*σ*]) = *E*

*(with σ as in (*[*2*](#_bookmark3)*)). Furthermore, any of the above implies:*

1. *L*(*V* )= *{L ∈* 2*A∗ | ∀*(*v, w*) *∈ E, L* = *L }.*

*E v w*

*2*

**Theorem 6.10 (on coequations and covarieties)** *Let D ⊆* 2*A∗ be a set of co- equations. The following statements are equivalent:*

1. *D* = coEq(*X, α*) *for some automaton* (*X, α*)
2. (*D, τ* ) *|*= *D*
3. coEq(*D, τ* )= *D*
4. *L*(*CD*)= *D*

*(with τ as in (*[*2*](#_bookmark3)*)).* *2*

**Corollary 6.11** *Every variety of languages L*(*VE*) *is also a covariety of lan- guages.* *2*

**Example 6.12** [Example [5.4](#_bookmark6) continued] Recall the automaton

(*Z, γ*) =

*a*

*b*



*x* ¸¸

v˛

*y a*

,

¸

*b*

and recall

coEq(*Z, γ*)=

*a,b b a*

et

*∅*



(*a∗b*)*∗* ¸\_

j

*a*

z\_ )

(*a∗b*)+

*b*

*a,b b a*



*A∗*

%r



(*b∗a*)+ ¸\_

)

*a*

z\_ j

(*b∗a*)*∗*

*b*

The smallest *covariety* containing (*Z, γ*) is

*C*coEq(*Z,γ*)

It contains the languages

*L*(*C*coEq )= *{ ∅,* (*a∗b*)*∗,* (*a∗b*)+*,* (*b∗a*)*∗,* (*b∗a*)+*, A∗ }*

(*Z,γ*)

The smallest *variety* containing (*Z, γ*) is

*V*Eq(*Z,γ*)

were we recall that Eq(*Z, γ*) is the smallest bisimulation equivalence (in fact, a congruence) generated by the set

*{aa* = *a, bb* = *b, ab* = *b, ba* = *a}*

We have

*L*(*V*Eq(*Z,γ*))= *{L ∈* 2*A∗ |* (*Laa* = *La*) *∧* (*Lbb* = *Lb*) *∧* (*Lab* = *Lb*) *∧* (*Lba* = *La*) *}*

= *{ ∅,* 1*,* (*a∗b*)*∗,* (*a∗b*)+*,* (*b∗a*)*∗,* (*b∗a*)+*, A*+*, A∗ }*

The latter set of languages can be, equivalently, determined using the fact that

*V*Eq(*Z,γ*) = *C*coEq( (*A∗,σ*)*/*Eq(*Z,γ*))

= *C*coEq( free(*Z,γ*))

To this end, we recall that

*a*



j

*a*  ¸

(*y, y*)

¸,

(free(*Z, γ*)*,* (*x, y*)*, γ*¯) =

(*x, y*) *a b*

J

*b*  (*x*¸*, x*)

,,

*b*

and compute coEq( free(*Z, γ*) ) by means of the following table, which contains all possible colourings *c* of free(*Z, γ*), together with the corresponding value of *oc*:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *c* | *c*((*x, y*)) | *c*((*y, y*)) | *c*((*x, x*)) | *oc*((*x, y*)) | *oc*((*y, y*)) | *oc*((*x, x*)) |
| *c*1 | 0 | 0 | 0 | *∅* | *∅* | *∅* |
| *c*2 | 0 | 0 | 1 | (*a∗b*)+ | (*a∗b*)+ | (*a∗b*)*∗* |
| *c*3 | 0 | 1 | 0 | (*b∗a*)+ | (*b∗a*)*∗* | (*b∗a*)+ |
| *c*4 | 0 | 1 | 1 | *A*+ | *A∗* | *A∗* |
| *c*5 | 1 | 0 | 0 | 1 | *∅* | *∅* |
| *c*6 | 1 | 0 | 1 | (*a∗b*)*∗* | (*a∗b*)+ | (*a∗b*)*∗* |
| *c*7 | 1 | 1 | 0 | (*b∗a*)*∗* | (*b∗a*)*∗* | (*b∗a*)+ |
| *c*8 | 1 | 1 | 1 | *A∗* | *A∗* | *A∗* |

In the end, this leads to the same set of languages. We conclude this example by observing that

*L*(*C*coEq(*Z,γ*)) *⊆ L*(*V*Eq(*Z,γ*))

as expected. *2*

**Example 6.13** Here we focus on a single given language, say: *L* = (*a∗b*)*∗*. A minimal automaton for *L* is

(*Z, x, c, γ*) =

*a*

*b*



*x* ¸¸

v˛

*y a*

,

¸

*b*

It follows from Example [6.12](#_bookmark8) that the smallest covariety of languages containing *L*

is

*L*(*C*coEq(*Z,γ*))= *{ ∅,* (*a∗b*)*∗,* (*a∗b*)+*,* (*b∗a*)*∗,* (*b∗a*)+*, A∗ }*

and that the smallest variety containing *L* is

*L*(*V*Eq )= *{ ∅,* 1*,* (*a∗b*)*∗,* (*a∗b*)+*,* (*b∗a*)*∗,* (*b∗a*)+*, A*+*, A∗ }*

(*Z,γ*)

*2*

**Example 6.14** Here are some further examples of varieties and covarieties.

* 1. The smallest congruence generated by *{ a* = *ε, b* = *ε }* is *E* = *A∗ × A∗*. As a consequence,

*L*(*VE*)= *{ ∅, A∗ }*

The same for *E* = *{ b* = *ε, ab* = *ε, aa* = *a }*.

* 1. If *E* is the smallest congruence generated by *{aa* = *ε, b* = *ε }*, then

*L*(*VE*)= *{ ∅,* ((*ab∗a*)+ *b*)*∗,* ((*ab∗a*)+ *b*)*∗ab∗, {a, b}∗ }*

* 1. If *E* is the smallest congruence generated by *{aa* = *ε, bb* = *ε }*, then the variety

*L*(*VE*) is infinite and contains both rational and non-rational languages.

*A∗*

* 1. For *D* =2 , the covariety *CD* contains *all* automata (*X, α*).
  2. For *D* = rat(2*A∗* ),

*CD* = *{*(*X, α*) *|* (*X, α*) is finitely generated *}*

that is, all (*X, α*) such that *⟨x⟩⊆ X* is finite, for all *x ∈ X*.

* 1. If *D* = *{ {a},* 1*, ∅}* then *CD* = *∅*.

# Transition monoids

For every (rational) language, one can construct its so-called *syntactic monoid* (that is, the transition monoid of its minimal automaton). Next every (classical, algebraic) variety *V* of monoids determines a class of languages *L* by the requirement that its syntactic monoid belongs to *V* . This is, in short, Eilenberg’s definition of a variety of languages. In this section, we take a first step towards an understanding of the relation between Eilenberg’s definition and the present one, by the observation that free(*X, α*), for every automaton (*X, α*), is isomorphic to its transition monoid.

A *monoid* (*M, ·,* 1) consists of a set *M* , a binary operation of multiplication that is associative, and a unit 1 with *m ·* 1 = 1 *· m* = *m*. For every set, there is the monoid

defined by

(*XX, ·,* 1*X* )

*XX* = *{φ | φ* : *X → X }* 1*X* (*x*)= *x f · g* = *g ◦ f*

Because of the isomorphism

*X → XA ∼*= *A → XX*

we have for every automaton (*X, α*) and *a ∈ A* a function

*a*˜ : *X → X a*˜(*x*)= *α*(*x*)(*a*)= *xa*

We use it to define for every automaton (*X, α*) a pointed automaton

(*XX,* 1*X, α*˜)

*α*˜(*φ*)(*a*)= *φ · a*˜

Next we define the *transition monoid* (cf. [[18](#_bookmark27)])

(trans(*X, α*)*,* 1*X, α*˜)

by trans(*X, α*)= im(*r*1 ), the image of the reachability map of (*XX,* 1*X, α*˜):

*X*

1

J*∗* *r*

¸ zz

trans(*X, α*)

*i*  z¸ *X*

*X*¸

1*X*

1*X*

*ε*

*A*

*r*1*X*

(where *r*(*a*1 *··· an*)= *a*˜1 *··· a*˜*n*, for *a*1 *··· an ∈ A∗*).

**Theorem 7.1** *For an automaton* (*X, α*)*,*

(free(*X, α*)*, x*¯*, α*¯) *∼*= (trans(*X, α*)*,* 1*X, α*˜)

**Proof.** Let *X* = *{x*1*,..., xn}*. For every *y*¯ *∈* free(*X, α*) we define

*φy*¯ : *X → X φy*¯(*xi*)= *yi*

Then *φ*(*y*¯)= *φy*¯ defines an isomorphism of pointed automata. *2*

This elementary observation should form the basis for a detailed comparison of the present definition of variety of languages and Eilenberg’s definition.

# References

1. M.A. Arbib and E.G. Manes. Adjoint machines, state-behaviour machines, and duality. *Journal of Pure and Applied Algebra*, 6:313–344, 1975.
2. M.A. Arbib and H.P. Zeiger. On the relevance of abstract algebra to control theory. *Automatica*, 5:589–606, 1969.
3. F. Bonchi, M. Bonsangue, H. Hansen, P. Panangaden, J. Rutten, and A. Silva. Algebra-coalgebra duality in Brzozowski’s minimization algorithm. 2013. Submitted.
4. A. Ballester-Bolinches, J.-E. Pin, and X. Soler-Escriva. Formations of finite monoids and formal languages: Eilenberg’s variety theorem revisited. *Forum Mathematicum*, 2012.
5. A. Ballester-Bolinches, J.-E. Pin, and X. Soler-Escriva. Languages associated with saturated formations of groups. *Forum Mathematicum*, 2013.
6. F. Bonchi, M. Bonsangue, J. Rutten, and A. Silva. Brzozowski’s algorithm (co)algebraically. In

R. Constable and A. Silva, editors, *Logic and Program Semantics.*, volume 7230 of *LNCS*, pages 12–23, 2012.

1. M. Bidoit, R. Hennicker, and A. Kurz. On the duality between observability and reachability. In Furio Honsell and Marino Miculan, editors, *FoSSaCS*, volume 2030 of *Lect. Notes in Comp. Sci.*, pages 72–87. Springer, 2001.
2. J.A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964.
3. C. Cirstea. On specification logics for algebra-coalgebra structures: Reconciling reachability and observability. In *Proceedings FoSSaCS*, pages 82–97, 2002.
4. J.H. Conway. *Regular algebra and finite machines*. Chapman and Hall, 1971.
5. S. Eilenberg. *Automata, languages and machines (Vol. A)*. Pure and applied mathematics. Academic Press, 1974.
6. S. Eilenberg. *Automata, languages and machines (Vol. B)*. Pure and applied mathematics. Academic Press, 1976.
7. M. Gehrke, S. Grigorieff, and J.-E. Pin. Duality and equational theory of regular languages. In

*Proceedings ICALP*, volume 5126 of *LNCS*, pages 246–257, 2008.

1. R. Kalman. On the general theory of control systems. *IRE Transactions on Automatic Control*, 4(3):110–110, 1959.
2. R. E. Kalman, P. L. Falb, and M. A. Arbib. *Topics in Mathematical Systems Theory*. McGraw Hill, 1969.
3. S.C. Kleene. Representation of events in nerve nets and finite automata. In Shannon and McCarthy, editors, *Automata Studies*, pages 3–41. Princeton Univ. Press, 1956.
4. E.G. Manes and M.A. Arbib. *Algebraic approaches to program semantics*. Texts and monographs in computer science. Springer-Verlag, 1986.
5. J.-E. Pin. Syntactic semigroups. *Handbook of language theory, Vol. I*, pages 679–746, 1997.
6. J.J.M.M. Rutten. Automata and coinduction (an exercise in coalgebra). In D. Sangiorgi and

R. de Simone, editors, *Proceedings of CONCUR’98*, volume 1466 of *LNCS*, pages 194–218, 1998.

1. J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000. Fundamental Study.