A Bialgebraic Review of Deterministic Automata, Regular Expressions and Languages

Bart Jacobs

Institute for Computing and Information Sciences, Radboud University Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands.
`B.Jacobs@cs.ru.nl` http://www.cs.ru.nl/B.Jacobs

To Joseph Goguen on the occasion of his 65th birthday

Abstract. This papers reviews the classical theory of deterministic automata and regular languages from a categorical perspective. The basis is formed by Rutten's description of the Brzozowski automaton structure in a coalgebraic framework. We enlarge the framework to a so-called bialgebraic one, by including algebras together with suitable distributive laws connecting the algebraic and coalgebraic structure of regular expressions and languages. This culminates in a reformulated proof via finality of Kozen's completeness result. It yields a complete axiomatisation of observational equivalence (bisimilarity) on regular expressions. We suggest that this situation is paradigmatic for (theoretical) computer science as the study of "generated behaviour".

1 Introduction

In the early seventies Joseph Goguen described automata within a categorical perspective (see for instance [11,12,13]), together with colleagues Arbib and Manes [1]. This paper fits in that tradition, using a more modern, bialgebraic setting, where algebra meets coalgebra. A bialgebra is a combined algebra and coalgebra $F(X) \to X \to G(X)$ on a common carrier (or state space) X, satisfying a certain compatibility requirement wrt. a distributive law connecting the two functors F, G. These bialgebras found application within the abstract, combined description of operational and denotational semantics started explicitly by Turi and Plotkin [35,34]—and more implicitly by Rutten and Turi [32]. This is now an active line of work [26,20,5,18].

Goguen has always shown an interest in methodological and philosophical issues surrounding computing. The work in this paper also lends itself to such reflections. It is often claimed that data processing is the subject of the discipline of computer science. We think it is more to the point to describe the subject of computer science as *generated behaviour*. This is the behaviour that can be observed on the outside, for instance via a screen or printer. It arises in interaction with the environment, as a result of the computer executing instructions.

This behaviouristic approach allows us to understand the relation with natural sciences: biology is about "spontaneous" behaviour, and physics concentrates on lifeless natural phenomena, without autonomous behaviour. The generated behaviour that we claim to be the subject of computer science arises by a computer executing a program according to strict operational rules. The behaviour is typically observed via the computer's I/O. Abstractly, the program can be understood as an element in an inductively defined set P of terms. This set thus forms a suitable initial algebra $F(P) \to P$, where the functor F captures the signature of the operations for forming programs. The operational rules for the behaviour of programs are described by a coalgebra $P \to G(P)$, where the functor G captures the kind of behaviour that can be displayed—such as deterministic, or with exceptions. We see that in abstract form, generated computer behaviour amounts to the repeated evaluation of an (inductively defined) coalgebra structure on an algebra of terms. Hence the bialgebras that form the basic structures used in this paper are at the heart of computer science.

One of the big challenges of computer science is to develop techniques for effectively establishing properties of generated behaviour. Often such properties are formulated positively as wanted, functional behaviour. But these properties may also be negative, like in computer security, where unwanted behaviour must be excluded. However, an appropriate logical view about program properties within the combined algebraic/coalgebraic setting has not been fully elaborated yet.

A distributive law is a natural transformation $FG \Rightarrow GF$ that describes (in the current setting) the proper interaction of term-formation and computational behaviour. The basic observation of [35,34], further elaborated [5], is that such natural transformations correspond to specification formats for operational rules on (inductively defined) programs. A bialgebra is an algebra-coalgebra pair satisfying a compatibility requirement wrt. a given distributive law. These bialgebras, as already claimed, form very fundamental structures in computing, because they combine algebraic structure with the associated computational behaviour. The compatibility requirement entails elementary properties like: observational equivalence (*i.e.* bisimulation wrt. the coalgebra) is a congruence (wrt. the algebra).

This paper concentrates on deterministic automata, regular expressions and languages. They form the very basic structures in computer science (see for instance [28]) which are studied early on in standard curricula in computing. The main contribution of this paper is the demonstration that these classic structures fit perfectly in the bialgebraic framework. In fact, they may be considered as a paradigmatic example. The paper does not contain new results on regular expressions / automata / languages as such, but on the way they can (or should) be organised. The proper mathematical language for this organisation is categorical. The reader is assumed to be familiar with basic notions like functor, natural transformation, (co)monad and adjunction, such as can be found in any introductory text on category theory. Our investigations take place in the category $\mathbf{Sets}$ of ordinary sets and functions. We are well aware that many results generalise to other categories, but we do not always strive for the highest level of generality.

There is already a large body of algebraic work on regular expressions, automata and languages, for instance within the context of regular algebras [9]. The coalgebraic perspective on this topic was introduced by Rutten [30,33,31], who demonstrated its fruitfulness especially for proving equalities via coinduction (using bisimulations). Rutten's work exploits the automaton structure on regular expressions introduced by Brzozowski [8,9]. Here we go a step further by developing the bialgebraic (combined algebraic-coalgebraic) perspective. This involves a number of new technical results:

– a general mechanism for obtaining distributive laws and bialgebras for deterministic automata in Section 3;
– a description of the free algebra and Brzozowski coalgebra structure on regular expressions as a bialgebra wrt. a (categorical) GSOS law in Subsection 4.3;
– a new proof of Kozen's completeness result [23,24] for regular expressions and languages in Section 5, by describing the coalgebra of regular expressions modulo equations as a final object. This shows that Kozen's axioms and rules give a complete axiomatisation of observational equivalence (bisimilarity) on regular expressions.

Throughout the paper we heavily rely on previous work, notably [35,31,24].

We expect that the bialgebraic picture that is emerging constitutes a paradigm which also applies to many more computational models (as already suggested in [35]). After all, regular expressions are extremely elementary, and capture only a very limited form of computation. Hence the bialgebraic paradigm is still in need of further instantiation, confirmation, and elaboration.

2 Deterministic Automata as Coalgebras

This section collects some standard facts about deterministic automata, described as coalgebras, in order to determine the setting and fix the notation.

We use two arbitrary sets A and B, where the elements of A may be understood as letters of an alphabet, and the elements of B as outputs. A **deterministic automaton** with A as input and B as output set consists of two functions:

$$\delta\colon X \longrightarrow X^A \quad \text{for transition} \qquad\qquad \varepsilon\colon X \longrightarrow B \quad \text{for output}$$

acting on a state space X. The transition function δ maps a state $x \in X$ and an input letter $a \in A$ to a successor state $x' = \delta(x)(a) \in X$. In that case one may write $x \xrightarrow{a} x'$. The output function ε gives for a state $x \in X$ the associated observable output $\varepsilon(x) \in B$.

The one-step transition function δ can be extended to a multiple-step transition function $\delta^\star$. The latter takes a state $x \in X$ and a sequence $\sigma \in A^\star$ of inputs to a successor state obtained by consecutively executing the steps in σ.

$$X \xrightarrow{\delta^\star} X^{A^\star} \qquad \text{defined as} \qquad \begin{cases} \delta^\star(x)(\langle\rangle) = x \\ \delta^\star(x)(a \cdot \sigma) = \delta^\star(\delta(x)(a))(\sigma) \end{cases} \tag{1}$$

This extended transition function $\delta^\star$ gives rise to the multiple-step transition notation: $x \xrightarrow{\sigma}{}^{*} x'$ stands for $x' = \delta^\star(x)(\sigma)$, and means that x' is the (non-immediate) successor state of x obtained by applying the inputs from the sequence $\sigma \in A^\star$, from left to right.

The behaviour $\mathsf{beh}(x)\colon A^\star \to B$ of a state $x \in X$ is then obtained as the function that maps a finite sequence $\sigma \in A^\star$ of inputs to the observable output

$$\mathsf{beh}(x)(\sigma) = \varepsilon(\delta^\star(x, \sigma)) \in B \tag{2}$$

The transition and output functions δ and ε of a deterministic automaton can be combined into a tuple $\langle\delta,\varepsilon\rangle\colon X \to X^A \times B$ forming a coalgebra of the functor $\mathcal{D} = \mathcal{D}_{A,B}$ given by $U \mapsto U^A \times B$. A coalgebra homomorphism from $\big(\langle\delta_1,\varepsilon_1\rangle\colon X_1 \to X_1^A \times B\big)$ to $\big(\langle\delta_2,\varepsilon_2\rangle\colon X_2 \to X_2^A \times B\big)$ consists of a function $h\colon X_1 \to X_2$ between the underlying state spaces satisfying:

$$\mathcal{D}(f) \circ \langle\delta_1,\varepsilon_1\rangle = \langle\delta_2,\varepsilon_2\rangle \circ f,$$

That is, $f^A \circ \delta_1 = \delta_2 \circ f$ and $\varepsilon_1 = \varepsilon_2 \circ f$. Or, more concretely, $f(\delta_1(x)(a)) = \delta_2(f(x))(a)$ and $\varepsilon_1(x) = \varepsilon_2(f(x))$, for all $x \in X$ and $a \in A$.

This describes morphisms in a category $\mathbf{CoAlg}(\mathcal{D})$. The following result, occurring for example in [2,29,16], is simple but often useful. It gives an explicit description of the final object in the category $\mathbf{CoAlg}(\mathcal{D})$. The proof is easy, and left to the reader.

Proposition 1. *The final coalgebra of the functor $\mathcal{D} = (-)^A \times B$ for deterministic automata is given by the set of behaviour functions $B^{A^\star}$, with structure:*

$$B^{A^\star} \xrightarrow{\quad\langle D, E\rangle\quad} \big(B^{A^\star}\big)^A \times B$$

given by:

$$D(\varphi)(a) = \lambda\sigma \in A^\star.\, \varphi(a \cdot \sigma) \qquad \textit{and} \qquad E(\varphi) = \varphi(\langle\rangle). \qquad \square$$

As is well-known—after Lambek—the structure map of a final coalgebra is an isomorphism. The carrier $B^{A^\star}$ of the final coalgebra collects all possible behaviours of deterministic automata. Two special cases are worth mentioning explicitly.

Example 1. Consider the above final coalgebra $B^{A^\star} \xrightarrow{\cong} \big(B^{A^\star}\big)^A \times B$ of the deterministic automata functor $\mathcal{D} = (-)^A \times B$.

1. When A is a singleton set $1 = \{0\}$, so that $A^\star = \mathbb{N}$, the resulting functor $\mathcal{D} = (-) \times B$ captures stream coalgebras $X \to X \times B$. Its final coalgebra is the set $B^{\mathbb{N}}$ of infinite sequences (streams) of elements of B, with $(\mathsf{tail}, \mathsf{head})$ structure,
$$B^{\mathbb{N}} \xrightarrow{\cong} B^{\mathbb{N}} \times B \quad \text{given by} \quad \varphi \longmapsto (\lambda n \in \mathbb{N}.\, \varphi(n+1),\, \varphi(0))$$
2. When $B = 2 = \{0,1\}$ describing final (or accepting) states of the automaton, the final coalgebra $B^{A^\star}$ is the set $\mathcal{L}(A) = \mathcal{P}(A^\star)$ of languages over the alphabet A, with structure:
$$\mathcal{L}(A) \xrightarrow{\cong} \mathcal{L}(A)^A \times 2 \quad \text{given by} \quad L \longmapsto (\lambda a \in A.\, D(L)(a),\, E(L))$$
where $D(L)(a)$ is the so-called a-derivative, introduced by Brzozowski [8], and defined as:
$$D(L)(a) = \{\sigma \in A^\star \mid a \cdot \sigma \in L\},$$
and where $E(L) = 1 \Longleftrightarrow \langle\rangle \in L$.
Given an arbitrary automaton $X \to X^A \times \{0,1\}$ of this type, the resulting behaviour map $\mathsf{beh}\colon X \to \mathcal{P}(A^\star)$ thus describes the language $\mathsf{beh}(x) \subseteq A^\star$ **accepted** by this automaton with $x \in X$ considered as initial state.

Both these final coalgebras $B^{\mathbb{N}}$ and $\mathcal{L}(A) = \mathcal{P}(A^\star)$ are studied extensively by Rutten, see [30,33,31]. One of the things that he emphasises is the use of bisimulation as a reasoning principle. Here we only sketch the main points, for deterministic automata.

Definition 1. *Consider two coalgebras $\langle\delta_1, \varepsilon_1\rangle\colon X_1 \to X_1^A \times B$ and $\langle\delta_2, \varepsilon_2\rangle\colon X_2 \to X_2^A \times B$. A* ***bisimulation*** *between them is a relation $R \subseteq X_1 \times X_2$ on the underlying state spaces that satisifies for all $x_1 \in X_1, x_2 \in X_2$,*

$$R(x_1, x_2) \Longrightarrow \begin{cases} \varepsilon_1(x_1) = \varepsilon_2(x_2), \text{ and} \\ R(\delta_1(x_1)(a), \delta_2(x_2)(a)), \text{ for all } a \in A. \end{cases}$$

We write $y_1 \leftrightarroweq y_2$ and call y_1, y_2 bisimilar if there is a bisimulation R with $R(y_1, y_2)$.

Bisimilarity expresses observational equality, that is, equality as far as one can observe with the available (coalgebraic) operations. This explains the following result.

Proposition 2. *In the situation of the previous definition one has: $y_1 \leftrightarroweq y_2$ if and only if $\mathsf{beh}_{\langle\delta_1,\varepsilon_1\rangle}(y_1) = \mathsf{beh}_{\langle\delta_2,\varepsilon_2\rangle}(y_2)$.*

Proof. The implication ($\Rightarrow$) is easy, since if $y_1 \leftrightarroweq y_2$, say via a bisimulation R with $R(y_1, y_2)$, then by induction, $R(\delta_1^\star(y_1)(\sigma), \delta_2^\star(y_2)(\sigma))$, for each $\sigma \in A^\star$. This yields $\mathsf{beh}_{\langle\delta_1,\varepsilon_1\rangle}(y_1) = \varepsilon_1(\delta_1^\star(y_1)(\sigma)) = \varepsilon_2(\delta_2^\star(y_2)(\sigma)) = \mathsf{beh}_{\langle\delta_2,\varepsilon_2\rangle}(y_2)$. For the reverse implication ($\Leftarrow$) one uses that the relation $\{(x_1, x_2) \mid \mathsf{beh}_{\langle\delta_1,\varepsilon_1\rangle}(x_1) = \mathsf{beh}_{\langle\delta_2,\varepsilon_2\rangle}(x_2)\}$ is a bisimulation. This follows directly because the beh maps are homomorphisms. □

States are thus bisimilar if and only if they are equal when mapped to the final coalgebras. Bisimulations provide a means to prove equations via "single-step" arguments. This makes coinductive reasoning similar to ordinary inductive approaches. See [15] for an abstract account of the underlying dualities.

Here is a very simple example—already using the regular algebra structure on languages from Example 3 later on. For each letter $a \in A$ one has $(1 + a)^* = a^*$ in $\mathcal{L}(A)$. This can be proven via the bisimulation $R = \{\langle(1 + a)^*, a^*\rangle\} \cup \{\langle\emptyset, \emptyset\rangle\}$.

At some stage we shall need the modal "eventually" operator $\Diamond$. Let $\langle\delta, \varepsilon\rangle\colon X \to X^A \times B$ be an arbitrary coalgebra / automaton. For a predicate (or subset) $P \subseteq X$ we define $\Diamond(P) \subseteq X$ as the set of all states that are reachable from P:

$$\Diamond(P) = \{\delta^\star(x)(\sigma) \mid x \in P, \sigma \in A^\star\}.$$

For a single state we write $\Diamond(x)$ for $\Diamond(\{x\})$. Note that $\Diamond(P)$ is a subcoalgebra / subautomaton, because it is by construction closed under transitions. It may be described as the least invariant containing P, see [17]. The greatest invariant $\Box(P)$ contained in P is the predicate $\{x \mid \forall\sigma \in A^\star.\, \delta^\star(x)(\sigma) \in P\}$.

3 Structured Output Sets and Distributive Laws

In [31, Section 9] the situation is studied where the output set B of a coalgebra $X \to X^A \times B$ is a semiring. This generalises the situations studied in [31] of final coalgebras of real-valued streams ($B = \mathbb{R}$) and languages ($B = 2$). It is shown that the sum and multiplication operations on B can be extended to the final coalgebras involved.

Here we go a step further and assume an algebra structure $\beta\colon T(B) \to B$, for a monad $T\colon \mathbf{Sets} \to \mathbf{Sets}$ with unit $\eta\colon \mathrm{id} \Rightarrow T$ and multiplication $\mu\colon T^2 \Rightarrow T$. Semirings then form a special case, see Subsection 3.4. We show how this T-algebra structure on the output set B induces a distributive law $T\mathcal{D} \Rightarrow \mathcal{D}T$, and a strengthened form of coinduction using "T-automata", following the approach of [36,5]. We shall give several illustrations involving different types of automata, for various concrete monads. These investigations go a bit beyond what is strictly needed for deterministic automata and regular languages.

To start, we recall that for an arbitrary monad T and functor G acting on the same category, a **distributive law** $\lambda\colon TG \Rightarrow GT$ is a natural transformation that interacts appropriately with the monads unit η and multiplication μ. This means that the following two diagrams commute.

$$\begin{tikzcd}
GX \arrow[r, "\eta_{GX}"] \arrow[dr, "G(\eta_X)"'] & TGX \arrow[d, "\lambda_X"] \\
& GTX
\end{tikzcd}
\qquad
\begin{tikzcd}
T^2GX \arrow[r, "T(\lambda_X)"] \arrow[d, "\mu_{GX}"'] & TGTX \arrow[r, "\lambda_{TX}"] & GT^2X \arrow[d, "G(\mu_X)"] \\
TGX \arrow[rr, "\lambda_X"'] & & GTX
\end{tikzcd}$$

Example 2. The next two illustrations will be used frequently. They both involve the so-called strength map.

1. For each functor T on the category $\mathbf{Sets}$ and for each set X there is a natural transformation $\mathsf{st}\colon T(-)^X \Rightarrow (-)^X T$. It is usually called **strength**, and given as map $T(Y^X) \to (TY)^X$ by the formula:

$$\mathsf{st}(u)(x) = T\big(\lambda h \in Y^X .\, h(x)\big)(u).$$

In case T happens to carry a monad structure, the strength map becomes a distributive law. The above two diagrams then translate into:

$$\begin{tikzcd}
Y^X \arrow[r, "\eta_{Y^X}"] \arrow[dr, "(\eta_Y)^X"'] & T(Y^X) \arrow[d, "\mathsf{st}"] \\
& (TY)^X
\end{tikzcd}
\qquad
\begin{tikzcd}
T^2(Y^X) \arrow[r, "T(\mathsf{st})"] \arrow[d, "\mu_{Y^X}"'] & T\big((TY)^X\big) \arrow[r, "\mathsf{st}"] & (T^2Y)^X \arrow[d, "(\mu_Y)^X"] \\
T(Y^X) \arrow[rr, "\mathsf{st}"'] & & (TY)^X
\end{tikzcd}$$

(The diligent reader may have noticed that strength is also natural in the functor, in the sense that for a natural transformation $\sigma\colon F \Rightarrow G$ one has $\mathsf{st}^G_{X,Y} \circ \sigma_{Y^X} = (\sigma_Y)^X \circ \mathsf{st}^F_{X,Y}$.)
One useful point about strength for monads is that it allows pointwise construction of algebras on function spaces: if $\alpha\colon T(Y) \to Y$ is an Eilenberg-Moore algebra, then so is $\alpha^X \circ \mathsf{st}\colon T(Y^X) \to (TY)^X \to Y^X$.

2. We have formulated the notion of a distributive law for a monad and a functor. There are several "obvious" variations, for instance for a functor and a comonad. The next example again involves strength, and is related to the final coalgebra construction in Proposition 1.
   To start, let $(M, \cdot, e)$ be an arbitrary monoid. It gives rise to a functor $(-)^M \colon \mathbf{Sets} \to \mathbf{Sets}$ that turns out to be a comonad. The counit $E_X \colon X^M \to X$ uses the monoids unit in $E_X(\varphi) = \varphi(e)$, and the comultiplication $C_X \colon X^M \to (X^M)^M$ works via the monoids multiplication in $C_X(\varphi) = \lambda a \in M.\, \lambda b \in M.\, \varphi(a \cdot b)$.
   We claim that for an arbitrary functor F, there is a distributive law $F(-)^M \Rightarrow (-)^M F$ over the comonad $(-)^M$. This law is again given by strength, and satisfies the following two "dual" properties.

$$\begin{tikzcd}
F(X^M) \arrow[r, "\mathsf{st}"] \arrow[dr, "F(E_X)"'] & (FX)^M \arrow[d, "E_{FX}"] \\
 & F(X)
\end{tikzcd}
\qquad
\begin{tikzcd}
F(X^M) \arrow[rr, "\mathsf{st}"] \arrow[d, "F(C_X)"] & & (FX)^M \arrow[d, "C_{FX}"] \\
F\big((X^M)^M\big) \arrow[r, "\mathsf{st}"'] & (F(X^M))^M \arrow[r, "\mathsf{st}^M"'] & \big((FX)^M\big)^M
\end{tikzcd}$$

   Why is all this relevant? Well, the final coalgebra structure described in Proposition 1 arises in this manner via the (free) monoid $(A^\star, \cdot, \langle\rangle)$ of strings with concatenation: its observation map $E \colon B^{A\star} \to B$ is precisely the above counit E_B, and its transition map $D \colon B^{A^\star} \to (B^{A^\star})^A$ arises from the comultiplication C_B, via restriction to singleton sequences: $D(\varphi)(a)(\sigma) = C(\varphi)(\langle a\rangle)(\sigma)$. The fact that strength forms a distributive law will be used in the proof of Proposition 4 below.

As stated in the beginning of this section, we assume an Eilenberg-Moore algebra $\beta \colon T(B) \to B$. By definition it satisfies the algebra laws $\beta \circ \eta = \text{id}$ and $\beta \circ T(\beta) = \beta \circ \mu$. Then we can define a distributive law of the monad T over the automata functor $\mathcal{D} = (-)^A \times B$ from the previous section, namely:

$$T\mathcal{D} \overset{\lambda}{\Longrightarrow} \mathcal{D}T \qquad \text{with components} \qquad T(X^A \times B) \xrightarrow{\lambda_X} (TX)^A \times B$$

This law is obtained as composite:

$$T(X^A \times B) \xrightarrow{\langle T(\pi_1), T(\pi_2)\rangle} T(X^A) \times TB \xrightarrow{\mathsf{st} \times \beta} (TX)^A \times B$$

The next result summarises what we have found so far.

Proposition 3. *Each Eilenberg-Moore algebra $T(B) \to B$ induces a distributive law $\lambda \colon T\mathcal{D} \Rightarrow \mathcal{D}T$ for the deterministic automata functor $\mathcal{D} = (-)^A \times B$.* □

When we have an arbitrary monad T, functor G, and a distributive law $\lambda \colon TG \Rightarrow GT$ the relevant associated notion is that of a **λ-bialgebra**: a pair of maps:

$$TX \xrightarrow{a} X \xrightarrow{b} GX$$

where:

– a is an Eilenberg-Moore algebra;
– a and b are compatible via λ, which means that the following diagram commutes.

$$\begin{tikzcd}
TX \arrow[r, "a"] \arrow[d, "T(b)"'] & X \arrow[r, "b"] & GX \\
TGX \arrow[rr, "\lambda_X"'] & & GTX \arrow[u, "G(a)"']
\end{tikzcd}$$

A **map of λ-bialgebras**, from $(TX \xrightarrow{a} X \xrightarrow{b} GX)$ to $(TY \xrightarrow{c} Y \xrightarrow{d} GY)$ is a map $f: X \to Y$ that is both a map of algebras and of coalgebras: $f \circ a = c \circ T(f)$ and $d \circ f = G(f) \circ b$.

The next two results are standard, see for *e.g.* [5,18], and are given without proof.

Lemma 1. *Assume a distributive law $\lambda: TG \Rightarrow GT$, and let $\zeta: Z \xrightarrow{\cong} GZ$ be a final coalgebra. It carries an Eilenberg-Moore algebra obtained by finality in:*

$$\begin{tikzcd}
GTZ \arrow[r, dashed, "G(\alpha)"] & GZ \\
TGZ \arrow[u, "\lambda_Z"] & \\
TZ \arrow[u, "T(\zeta)", "\cong"'] \arrow[r, dashed, "\alpha"'] & Z \arrow[uu, "\cong", "\zeta"']
\end{tikzcd}$$

The resulting pair $(TZ \xrightarrow{\alpha} Z \xrightarrow{\zeta} GZ)$ is then a final λ-bialgebra. □

Lemma 2. *In presence of a distributive law $\lambda: TG \Rightarrow GT$, there exists a bijective correspondence between GT-coalgebras $e: X \to GTX$ (also called equations) and λ-bialgebras $(T^2X \xrightarrow{\mu_X} TX \xrightarrow{d} GTX)$ with free algebra μ_X.*

Moreover, let $(TY \xrightarrow{a} Y \xrightarrow{b} GY)$ be a λ-bialgebra. Then there is a bijective correspondence between "solutions of e" $f: X \to Y$ in:

$$\begin{tikzcd}
GTX \arrow[r, "GT(f)"] & GTY \arrow[d, "G(a)"] \\
 & GY \\
X \arrow[uu, "e"] \arrow[r, "f"'] & Y \arrow[u, "b"]
\end{tikzcd}$$

and λ-bialgebra maps $g: TX \to Y$—for the associated equations and λ-bialgebras. □

Proposition 4. *The assumed algebra $\beta\colon TB \to B$ induces on the carrier $B^{A^\star}$ of the final $\mathcal{D}$-coalgebra from Proposition 1 another T-algebra via a pointwise construction, namely,*

$$\widehat{\beta} \overset{def}{=} \Big(T(B^{A^\star}) \xrightarrow{\mathsf{st}} (TB)^{A^\star} \xrightarrow{\beta^{A^\star}} B^{A^\star}\Big)$$

so that $E\colon B^{A^\star} \to B$ is a homomorphism of algebras. This $\widehat{\beta}$ is the unique coalgebra homomorphism from Lemma 1,

$$\begin{tikzcd}
\mathcal{D}T(B^{A^\star}) \arrow[rr, dashed, "\mathcal{D}(\widehat{\beta})"] & & \mathcal{D}(B^{A^\star}) \\
T\mathcal{D}(B^{A^\star}) \arrow[u, "\lambda_{B^{A^\star}}"] & & \\
T(B^{A^\star}) \arrow[u, "T(\langle D, E\rangle)", "\cong"'] \arrow[rr, dashed, "\widehat{\beta}"'] & & B^{A^\star} \arrow[uu, "\cong"', "\langle D, E\rangle"]
\end{tikzcd}$$

using the distributive law from Proposition 3. Hence, this $\widehat{\beta}$ together with the final coalgebra forms the final λ-bialgebra: $T(B^{A^\star}) \xrightarrow{\widehat{\beta}} B^{A^\star} \xrightarrow{\langle D,E\rangle} \mathcal{D}(B^{A^\star})$.

Proof. According to Lemma 1 it suffices to prove that $\widehat{\beta}$ is a homomorphism of coalgebras. Here we use that strength is a distributive law as described in Example 2.(2).

$$\begin{aligned}
&\mathcal{D}(\widehat{\beta}) \circ \lambda \circ T(\langle D, E\rangle) \\
&= \mathcal{D}(\beta^{A^\star} \circ \mathsf{st}) \circ \langle \mathsf{st} \circ T(\pi_1), \beta \circ T(\pi_2)\rangle \circ T(\langle D, E\rangle) \\
&= \langle (\beta^{A^\star})^A \circ \mathsf{st}^A \circ \mathsf{st} \circ T(D), \beta \circ T(E)\rangle \\
&= \langle (\beta^{A^\star})^A \circ D \circ \mathsf{st}, \beta \circ E \circ \mathsf{st}\rangle \\
&= \langle (\beta^{A^\star})^A \circ D, \beta \circ E\rangle \circ \mathsf{st} \\
&= \langle D, E\rangle \circ \beta^{A^\star} \circ \mathsf{st} \\
&= \langle D, E\rangle \circ \widehat{\beta}.
\end{aligned}$$

□

The coinduction principle associated with a final λ-bialgebra is called **λ-coinduction** in [5]. In the current situation, with the functor $\mathcal{D}$ for deterministic automata, the principle yields a strengthened form of coinduction for "T-automata".

Theorem 1. *For each T-automaton $\langle \delta, \varepsilon\rangle\colon X \to \mathcal{D}(TX) = (TX)^A \times B$—where B carries a T-algebra $\beta\colon TB \to B$—there is a unique map $\mathsf{beh}\colon X \to B^{A^\star}$ making the following diagram commute.*

$$\begin{tikzcd}
(TX)^A \times B = \mathcal{D}TX \arrow[rr, "\mathcal{D}T(\mathsf{beh})"] & & \mathcal{D}T(B^{A^\star}) \arrow[d, "\mathcal{D}(\widehat{\beta})"] \\
 & & \mathcal{D}(B^{A^\star}) \\
X \arrow[uu, "\langle \delta, \varepsilon\rangle"] \arrow[rr, "\mathsf{beh}"'] & & B^{A^\star} \arrow[u, "\langle D, E\rangle"']
\end{tikzcd}$$

Proof. This result is a direct consequence of Lemmas 1 and 2, but we like to give the concrete construction, as in the proof of Proposition 1. First we define an extension $\delta^\star \colon X \to (TX)^{A^\star}$ of δ like in (1) by induction:

$$\delta^\star(x)(\langle\rangle) = \eta(x) \qquad \delta^\star(x)(a\cdot\sigma) = \mu\Big[\mathsf{st}\Big(T(\delta^\star)\big(\delta(x)(a)\big)\Big)(\sigma)\Big].$$

Then we can define the required map as:

$$\mathsf{beh} = \Big(X \xrightarrow{\delta^\star} (TX)^{A^\star} \xrightarrow{(T\varepsilon)^{A^\star}} (TB)^{A^\star} \xrightarrow{\beta^{A^\star}} B^{A^\star}\Big). \qquad \square$$

In the remainder of this section we shall investigate several instantiations of the monad T in the results above.

3.1 The Identity Monad and Deterministic Automata

If we take $T = \mathrm{id}$, with $\beta = \mathrm{id}$ as identity algebra we get $\lambda = \mathrm{id}$ and $\widehat{\beta} = \mathrm{id}$, so that λ-coinduction is just the ordinary form of coinduction for deterministic automata.

3.2 The Powerset Monad and Non-deterministic Automata

In the above context we now consider the situation where the monad T is the powerset monad $\mathcal{P}$ and where the output set B is $2 = \{0,1\}$. An Eilenberg-Moore algebra of $\mathcal{P}$ is a complete lattice (see *e.g.* [25, Chapter VI.2, Exerice 1]), *i.e.* a poset with joins (and hence also meets) of all subsets. Since $2 = \mathcal{P}(1)$, we have a free monad structure $\bigcup \colon \mathcal{P}(2) \to 2$ given by union. The strength map $\mathsf{st}\colon \mathcal{P}(Y^X) \to \mathcal{P}(Y)^X$ is $\mathsf{st}(u)(x) = \{f(x) \mid f \in u\}$. The resulting distributive law, say $\lambda^{\mathcal{P}} \colon \mathcal{PD} \Rightarrow \mathcal{DP}$, is given by:

$$\begin{tikzcd} \mathcal{P}(X^A \times 2) \arrow[r, "\lambda^{\mathcal{P}}_X"] & \mathcal{P}(X)^A \times 2 \\ U \arrow[r, mapsto] & \langle \lambda a \in A.\, \{f(a) \mid \exists b.\, (f,b) \in U\},\ \exists f.\, (f,1) \in U\rangle \end{tikzcd}$$

The final coalgebra is in this case the set $2^{A^\star} = \mathcal{P}(A^\star) = \mathcal{L}(A)$ of languages over the "alphabet" A, see Example 1 (ii). The induced algebra structure $\mathcal{P}(\mathcal{L}(A)) \to \mathcal{L}(A)$ is simply union $\bigcup$.

The $\lambda^{\mathcal{P}}$-coinduction principle from Theorem 1 tells how a state x of a non-deterministic automaton is mapped to the associated language (that is accepted starting from x as initial state):

$$\begin{tikzcd} \mathcal{P}(X)^A \times 2 = \mathcal{DP}(X) \arrow[r, dashed] & \mathcal{DPL}(A) \arrow[d, "\mathcal{D}(\bigcup)"] \\ & \mathcal{DL}(A) = \mathcal{L}(A)^A \times 2 \\ X \arrow[uu] \arrow[r, dashed] & \mathcal{L}(A) \arrow[u, "\cong"] \end{tikzcd}$$

This was first noted in [5, Corollary 4.4.6].

3.3 The Multiset Monad and Weighted Automata

It is well-known that the Kleene-star or list monad $X \mapsto X^\star$ has monoids as Eilenberg-Moore algebras. The monad $\mathcal{M}$ for commutative monoids is given by multisets:

$$\mathcal{M}(X) = \{\varphi \in \mathbb{N}^X \mid \varphi \text{ has finite support}\},$$

where the support of φ is the set $\mathsf{supp}(\varphi) = \{x \in X \mid \varphi(x) \neq 0\}$. Such a φ can thus be represented as finite sum $n_1x_1 + \cdots + n_kx_k$ of elements $x_i \in X$ with "multiplicities" $n_i = \varphi(x_i) \in \mathbb{N}$. The action $\mathcal{M}(f)$ on such a representation is then simply $n_1f(x_1) + \cdots + n_kf(x_k)$. The unit of this monad is $x \mapsto 1x$ and multiplication is $n_1\varphi_1 + \cdots + n_k\varphi_k \mapsto \lambda x \in X.\, n_1\varphi_1(x) + \cdots + n_k\varphi_k(x)$.

An $\mathcal{M}$-automaton $\langle \delta, \varepsilon\rangle : X \to \mathcal{M}(X)^A \times 2$ is then a so-called weighted automaton. For a state $x \in X$ and letter $a \in A$ there may then be several result states x_i in the outcome $\delta(x)(a) = n_1x_1 + \cdots + n_kx_k$, each with a particular "weight" n_i.

The set 2 forms a commutative monoid via finite disjunctions $\top, \vee$—and also via conjunctions. The disjunctions induce a commutative monoid structure on $\mathcal{L}(A)$ given by union of languages. Since this is an idempotent monoid, the structure of multiplicities is ignored when a state is mapped to the associated language.

3.4 The Semiring Monad

A basic observation is that there is a distributive law of monads $\pi : (-)^\star \circ \mathcal{M} \Rightarrow \mathcal{M} \circ (-)^\star$ between the list and multiset monads. It is given by multiplication in $\mathbb{N}$:

$$\begin{tikzcd}
\mathcal{M}(X)^\star \arrow[r, "\pi_X"] & \mathcal{M}(X^\star) \\
\langle \varphi_1, \ldots, \varphi_n\rangle \arrow[r, mapsto] & \sum\{\varphi_1(x_1)\cdots\varphi_n(x_n)\langle x_1, \ldots, x_n\rangle \mid x_i \in \mathsf{supp}(\varphi_i)\}
\end{tikzcd}$$

$$= \lambda\langle y_1, \ldots, y_m\rangle \in X^\star.\ \begin{cases} 0 & \text{if } m \neq n \\ \varphi_1(y_1)\cdots\varphi_n(y_n) & \text{otherwise.} \end{cases}$$

With some perseverance one can prove that π is a natural transformation that commutes appropriately with the monad structures.

It is a standard result that in presence of a distributive law like $\pi : (-)^\star \circ \mathcal{M} \Rightarrow \mathcal{M} \circ (-)^\star$ the composite $\mathcal{M} \circ (-)^\star$ is again a monad, see for instance [6,19,4]. Moreover, the multiset monad $\mathcal{M}$ can be lifted to a monad $\overline{\mathcal{M}}$ on the category of $(-)^\star$-algebra (monoids), such that the algebras of the composite monad $\mathcal{M} \circ (-)^\star$ are the same as $\overline{\mathcal{M}}$-algebras. This functor $\overline{\mathcal{M}}$ maps a monoid $(X, \cdot, 1)$ to $(\mathcal{M}(X), \bullet, \eta(1))$ with multiplication $\bullet$ given by:

$$\varphi \bullet \psi = \textstyle\sum\{\varphi(x)\psi(y)(x \cdot y) \mid x \in \mathsf{supp}(\varphi), y \in \mathsf{supp}(\psi)\}.$$

An Eilenberg-Moore algebra $(\mathcal{M}(X), \bullet, \eta(1)) \to (X, \cdot, 1)$ for the monad $\overline{\mathcal{M}}$ consists of a commutative monoid $m : \mathcal{M}(X) \to X$ whose structure map m preserves the monoid structure. Such an algebra of the composite monad is thus a **semiring**. Therefore we call the monad the semiring monad, and write it as $\mathcal{S}(X) = \mathcal{M}(X^\star)$.

Rutten [31, Section 9] explicitly considers deterministic automata $X \to X^A \times B$ where the set B is a semiring, *i.e.* carries an Eilenberg-Moore algebra $\mathcal{S}(B) \to B$. This includes his main examples $B = \mathbb{R}$ and $B = 2$. In those cases the final coalgebra $B^{A^\star}$ is also a semiring, via pointwise construction. Theorem 1 yields for a "semiring" automaton $X \to \mathcal{S}(X)^A \times 2$ a mapping $X \to \mathcal{L}(A)$ to languages over A.

3.5 The Language Monad

The language monad $\mathcal{L}(X) = \mathcal{P}(X^\star)$ can be constructed similarly to the semiring monad $\mathcal{S}(X) = \mathcal{M}(X^\star)$, namely via a distributive law. The algebras of the language monad are Kleene algebras with arbitrary joins, also known as unital quantales, see [18] for more information. Theorem 1 then yields behaviours for states of "language automata" $X \to \mathcal{L}(X)^A \times B$. They resemble alternating automata [27].

4 Regular Expressions

As is well-known, regular expressions are built up from constants $0, 1$, letters $a \in A$ from a given alphabet A, sum $s + t$, composition $s \cdot t$ and Kleene-star s^*. These operations form an algebra of the functor:

$$\mathcal{R}(X) = 1 + 1 + (X \times X) + (X \times X) + X$$

where we ignore the alphabet for a moment—because it will turn up in the associated monad below. The initial algebra of this functor $\mathcal{R}$ is not so interesting: it consists of the (closed) terms that can be obtained from $0, 1$ via $+, \cdot, (-)^*$. Notice that at this stage there are no equations involved. They will appear in the next section.

Example 3. For an arbitrary set U, the set of languages $\mathcal{L}(U) = \mathcal{P}(U^\star)$ over U carries an $\mathcal{R}$-algebra structure $\mathcal{R}(\mathcal{L}(U)) \to \mathcal{L}(U)$. It is given by the familiar definitions

$$\begin{cases} \text{zero term case:} & 0 \longmapsto \emptyset \\ \text{one term case:} & 1 \longmapsto \{\langle\rangle\} \\ \text{sum case:} \ (L_1, L_2) & \longmapsto L_1 \cup L_2 \\ \text{product case:} \ (L_1, L_2) & \longmapsto \{\sigma_1 \cdot \sigma_2 \mid \sigma_i \in L_i\} \\ \text{star case:} & L \longmapsto \bigcup_{n \in \mathbb{N}} L^n. \end{cases}$$

since a single (algebra) map $\mathcal{R}(\mathcal{L}(U)) \to \mathcal{L}(U)$ jointly describes five maps of the form $1 \to \mathcal{L}(U)$, $1 \to \mathcal{L}(U)$, $\mathcal{L}(U) \times \mathcal{L}(U) \to \mathcal{L}(U)$, $\mathcal{L}(U) \times \mathcal{L}(U) \to \mathcal{L}(U)$ and $\mathcal{L}(U) \to \mathcal{L}(U)$, giving the individual operations of regular algebra.

For the special case where $U = \emptyset$ we get an algebra structure on $\mathcal{L}(\emptyset) = \mathcal{P}(\emptyset^\star) = \mathcal{P}(1) = 2$. This structure $\mathcal{R}(2) \to 2$ uses $0, \vee$ and $1, \wedge$ as additive and multiplicative monoids, and the constant map $x \mapsto 1$ as star operation.

Usually one considers regular expressions over an alphabet A. It means that the letters $a \in A$ are used as atoms to build up regular expressions. This can be done via the free monad $\mathcal{R}^*$ generated by $\mathcal{R}$. It is defined on a set A as the initial algebra of the functor $X \mapsto A + \mathcal{R}(X)$. We shall sometimes write Re_A for the carrier $\mathcal{R}^*(A)$ of regular expressions over A, or simply Re if the alphabet A is clear from the context. This set Re is built up inductively from $0, 1, a \in A$ using the regular operations $+, \cdot, (-)^*$.

We thus have an initiality isomorphism $[\eta_A, \tau_A] \colon A + \mathcal{R}(\mathsf{Re}) \xrightarrow{\cong} \mathsf{Re}$, where the map $\tau_A \colon \mathcal{R}(\mathcal{R}^*(A)) \to \mathcal{R}^*(A)$ is the free $\mathcal{R}$-algebra on A. The extension map $\sigma \colon \mathcal{R} \Rightarrow \mathcal{R}^*$ is then given by $\sigma = \tau \circ \mathcal{R}(\eta)$.

The next result collects the basics about this situation.

Lemma 3. *In the situation described above:*

1. *The functor $A \mapsto \mathcal{R}^*(A)$ is a monad, whose category of Eilenberg-Moore algebras is isomorphic to the category of $\mathcal{R}$-algebras. The multiplication of this monad is defined by initiality in:*

$$\begin{tikzcd}
\mathcal{R}^*A + \mathcal{R}\big(\mathcal{R}^*\mathcal{R}^*A\big) \arrow[r, dashed, "id + \mathcal{R}(\mu_A)"] \arrow[d, "{[\eta, \tau]}"', "\cong"] & \mathcal{R}^*A + \mathcal{R}(\mathcal{R}^*A) \arrow[d, "{[id, \tau]}"] \\
\mathcal{R}^*\mathcal{R}^*A \arrow[r, dashed, "\mu_A"'] & \mathcal{R}^*A
\end{tikzcd}$$

2. *The $\mathcal{R}$-algebra on 2 from Example 3 yields a distributive law $\lambda \colon \mathcal{R}^*\mathcal{D} \Rightarrow \mathcal{D}\mathcal{R}^*$ for the deterministic automaton functor $\mathcal{D} = (-)^A \times 2$.*

Proof. The first point is standard, and the second is a special case of Proposition 3. □

With this result, an $\mathcal{R}$-algebra from Example 3, say $r \colon \mathcal{R}(\mathcal{L}(U)) \to \mathcal{L}(U)$ corresponds to a unique Eilenberg-Moore algebra $\overline{r} \colon \mathcal{R}^*(\mathcal{L}(U)) \to \mathcal{L}(U)$ with $\overline{r} \circ \sigma = r$. Especially for $U = \emptyset$ this yields an algebra $\mathcal{R}^*(2) \to 2$ that will be used in (4) below. The multiplication μ maps a term $s(t_1, \ldots, t_n)$ built up from other terms $t_1, \ldots, t_n$ as atoms, to the term $s[t_1, \ldots, t_n]$ obtained by substituting these t_i into s.

Example 4. The standard interpretation of the set Re_A regular expressions over an alphabet A in the set $\mathcal{L}(A)$ of languages over A may be understood as the unique homomorphism of algebras:

$$\begin{tikzcd}
\mathcal{R}^*\mathcal{R}^*A \arrow[r, dashed, "\mathcal{R}^*(\llbracket - \rrbracket)"] \arrow[d, "\mu_A"'] & \mathcal{R}^*\big(\mathcal{L}(A)\big) \arrow[d] \\
\mathsf{Re}_A = \mathcal{R}^*A \arrow[r, dashed, "\llbracket - \rrbracket"'] & \mathcal{L}(A)
\end{tikzcd} \qquad \text{with} \quad \llbracket\, \eta(a) \,\rrbracket = \{\langle a \rangle\}.$$

The Eilenberg-Moore algebra on $\mathcal{L}(A)$ arises from the $\mathcal{R}$-algebra from Example 3. Freeness of μ_A and the inclusion $\{\langle - \rangle\} \colon A \to \mathcal{L}(A)$ does the rest.

Usually one does not make a clear distinction between an expression like $s = 1 + a^*ba^* \in \mathsf{Re}_A$ and its interpretation $\llbracket\, s \,\rrbracket = 1 \cup a^*ba^* \in \mathcal{L}(A)$. Here however, we like to keep the two apart, and use an explicit interpretation function $\llbracket - \rrbracket$.

4.1 Two Questions

Given this basic set-up, we ask ourselves the following two questions.

1. Is there a coalgebra/automaton structure $\langle D, E\rangle$ on regular expressions such that the above interpretation $[\![-]\!]$ is also a homomorphism of coalgebras, as in:

$$\begin{tikzcd}
\mathcal{R}^*\mathcal{R}^*A \arrow[r, "\mathcal{R}^*([\![-]\!])"] \arrow[d, "\mu_A"'] & \mathcal{R}^*\big(\mathcal{L}(A)\big) \arrow[d] \\
\mathsf{Re}_A = \mathcal{R}^*A \arrow[r, "{[\![-]\!]}"] \arrow[d, dotted, "\langle D, E\rangle"', "??"] & \mathcal{L}(A) \arrow[d, "\cong"', "\langle \delta, \varepsilon\rangle"] \\
(\mathcal{R}^*A)^A \times 2 \arrow[r, "{[\![-]\!]^A \times 2}"'] & \mathcal{L}(A)^A \times 2
\end{tikzcd} \tag{3}$$

2. Is this diagram a map between two κ-bialgebras, for a suitable distributive law κ.

We address this matter in the next two subsections. The first question can be answered positively, and involves Brzozowski's "derivative" and "non-empty word" operations on regular expressions from [8,9]. The second question will be solved by a special kind of distributive law, following the so-called GSOS format. It puts the concrete construction of Brzozowski in the general framework developed in [35].

4.2 Regular Expressions as Coalgebras

From a coalgebraic perspective the most interesting part of regular expressions is that they form a deterministic automaton $\langle D, E\rangle \colon \mathsf{Re} \to \mathsf{Re}^A \times 2 = \mathcal{D}(\mathcal{R}^*(A))$.

The output operation $E\colon \mathsf{Re} \to 2$ is obtained by freeness as the unique map in

$$\begin{tikzcd}
\mathcal{R}^*(\mathsf{Re}) \arrow[r, dashed, "\mathcal{R}^*(E)"] \arrow[d, "\mu"'] & \mathcal{R}^*(2) \arrow[d] \\
\mathsf{Re} \arrow[r, dashed, "E"'] & 2
\end{tikzcd} \qquad \text{with} \quad E(\eta(a)) = 0 \tag{4}$$

where the algebra structure $\mathcal{R}^*(2) \to 2$ is as described before Example 4. Commutation of the diagram (4) yields the equations $E(0) = 0$, $E(1) = 1$, $E(s + t) = E(s) \vee E(t)$, $E(s \cdot t) = E(s) \wedge E(t)$ and $E(s^*) = 1$. This operation E describes what is sometimes called the empty word property.

Since the values of $E(s) \in 2$ are either 0 or 1, we shall often treat $E(s)$ as a term in Re.

By induction on the structure of a term $s \in \mathsf{Re}$ one checks the first bi-implication:

$$E(s) = 1 \Longleftrightarrow \langle\rangle \in [\![\, s\,]\!] \Longleftrightarrow (\varepsilon \circ [\![-]\!])(s) = 1$$

Hence $\varepsilon \circ [\![-]\!] = E$, which is one part of the lower square in (3).

The "derivative" operation $D\colon \mathsf{Re} \to \mathsf{Re}^A$ is more complicated. It is due to Brzozowski [8], see also [9]. We shall use the common notation $D_a(s)$ for the successor term $D(s)(a)$. The derivative is defined by the following clauses (or rules).

$$\begin{aligned} D_a(0) &= 0 & D_a(s+t) &= D_a(s) + D_a(t) \\ D_a(1) &= 0 & D_a(s \cdot t) &= D_a(s) \cdot t + E(s) \cdot D_a(t) \\ D_a(b) &= \begin{cases} 1 \text{ if } b = a \\ 0 \text{ otherwise.} \end{cases} & D_a(s^*) &= D_a(s) \cdot s^*. \end{aligned} \tag{5}$$

Is this a proper inductive definition? The problem is in the clause for composition, where the term t is used in the subterm $D_a(s) \cdot t$ in original form. Similarly for s in the star case. Hence we cannot use an inductive/freeness definition like for E in (4). We have to use recursion to deal with the additional parameter. The remainder of this subsection elaborates the required formulation of recursion.

A categorical analysis of strengthened induction principles for a functor F is given in [36] in terms of distributive laws between F and a comonad—dual to the approach underlying Theorem 1. We shall use this approach in the current situation where F is the functor $A + \mathcal{R}(-)$ for regular expressions described in the beginning of this section and the comonad is simply $(-) \times D$ for a set D, with coalgebra $\Delta = \langle \mathrm{id}, \mathrm{id} \rangle$. We concentrate on the result, and refer to [36] for the distributive law involved.

Theorem 2 (Recursion following [36]). *An initial algebra $\alpha\colon F(D) \xrightarrow{\cong} D$ satisfies the following strengthened induction property: for each map $f\colon F(X \times D) \to X$ there is a unique map $h\colon D \to X$ making the following diagram commute.*

$$\begin{tikzcd} F(D \times D) \arrow[rr, "F(h \times D)"] & & F(X \times D) \arrow[dd, "f"] \\ F(D) \arrow[u, "F(\Delta)"] \arrow[d, "\alpha"', "\cong"] & & \\ D \arrow[rr, "h"'] & & X \end{tikzcd}$$

Proof. We shall give a direct proof, ignoring the distributivity properties involved. Let $f\colon F(X \times D) \to X$ therefore be given. Write $f' = \langle f, \alpha \circ F(\pi_2) \rangle\colon F(X \times D) \to X \times D$. It gives by initiality rise to a unique map $k\colon D \to X \times D$ with $k \circ \alpha = f' \circ F(k)$. Then $\pi_2 \circ k = \mathrm{id}$ by uniqueness of algebra maps $\alpha \to \alpha$. Hence we take $h = \pi_1 \circ k$. □

With this theorem the derivative operation $D\colon \mathsf{Re} \to \mathsf{Re}^A$ can be obtained by recursion from a map $[f_1, f_2]\colon A + \mathcal{R}(\mathsf{Re}^A \times \mathsf{Re}) \to \mathsf{Re}^A$ in:

$$\begin{tikzcd} A + \mathcal{R}(\mathsf{Re}) \arrow[r, dashed, "\mathrm{id} + \mathcal{R}(\langle D, \mathrm{id} \rangle)"] \arrow[d, "{[\eta, \tau]}"', "\cong"] & A + \mathcal{R}(\mathsf{Re}^A \times \mathsf{Re}) \arrow[d, "{[f_1, f_2]}"] \\ \mathsf{Re} \arrow[r, dashed, "D"'] & \mathsf{Re}^A \end{tikzcd} \tag{6}$$

The map $f_1\colon A \to \mathsf{Re}^A$ is defined as $f_1(a) = \lambda b \in A.\,\mathsf{if}\ b = a\ \mathsf{then}\ 1\ \mathsf{else}\ 0$. And $f_2\colon \mathcal{R}(\mathsf{Re}^A \times \mathsf{Re}) \to \mathsf{Re}^A$ is given by the following cases.

$$\begin{cases} \text{zero term case:} & 0 \longmapsto \lambda a \in A.\, 0 \\ \text{one term case:} & 1 \longmapsto \lambda a \in A.\, 0 \\ \text{sum case:} & (\langle \varphi_1, s_1\rangle, \langle \varphi_2, s_2\rangle) \longmapsto \lambda a \in A.\, \varphi_1(a) + \varphi_2(a) \\ \text{product case:} & (\langle \varphi_1, s_1\rangle, \langle \varphi_2, s_2\rangle) \longmapsto \lambda a \in A.\, \varphi_1(a)\cdot s_2 + E(s_1)\cdot \varphi_2(a) \\ \text{star case:} & (\varphi, s) \longmapsto \lambda a \in A.\, \varphi(a)\cdot s^*. \end{cases}$$

Commutation of the diagram (6) now yields the appropriate clauses (5) for the derivative function. Further, by induction on $s \in \mathsf{Re}$ one proves:

$$\begin{aligned} [\![\, D(s)(a) \,]\!] &= D([\![\, s \,]\!])(a) \quad \text{as in Example 1 (ii)} \\ &= \{\sigma \in A^\star \mid a\cdot\sigma \in [\![\, s \,]\!]\}. \end{aligned}$$

This means that $[\![\, - \,]\!]$ is a homomorphism of both algebras and coalgebras in (3). This settles our first question from Subsection 4.1. In particular, the operational semantics ($[\![\, - \,]\!]$ as coalgebra homomorphism) is compositional (*i.e.* is an algebra homomorphism).

We now turn to the second question from Subsection 4.1.

4.3 Regular Expressions as Bialgebras

Since the derivative operation $D\colon \mathsf{Re} \to \mathsf{Re}^A$ is defined by recursion (instead of induction), the distributive laws and bialgebras described in Section 3 do not work in this situation. Interestingly, the so-called GSOS format does work. It has been developed in syntactic form for process calculi [7,14], and formulated categorically in [35]. We follow the latter approach—see also [5]. The main point is that these GSOS laws have an extra parameter—like in recursion.

Definition 2. *For a monad T and functor G, a* ***GSOS law*** *is a distributive law of the form $\lambda\colon T(G \times id) \Rightarrow (G \times id)T$ with $\pi_2 \circ \lambda = T(\pi_2)$.*

A ***λ-model****, or* ***GSOS model****, for such a GSOS law λ, consists of an Eilenberg-Moore algebra $a\colon TX \to X$ and a coalgebra $b\colon X \to GX$ on the same state space, such that the pair $TX \xrightarrow{a} X \xrightarrow{\langle b, id\rangle} GX \times X$ is a λ-bialgebra; equivalently, such that the following diagram commutes.*

$$\begin{tikzcd} TX \arrow[r, "a"] \arrow[d, "T(\langle b, id\rangle)"'] & X \arrow[r, "b"] & GX \\ T(GX \times X) \arrow[rr, "\pi_1 \circ \lambda"'] & & GTX \arrow[u, "Ga"'] \end{tikzcd}$$

The formulation of GSOS law that we use is not quite the same as in [35]. The latter handles the special case where the monad T is free, *i.e.* of the form F^*. The next result shows that this special case of our definition is equivalent to the "natural transformation" formulation used in [35].

Proposition 5. *Let F be an arbitrary endofunctor with associated free monad F^*. There is a bijective correspondence between:*

$$\frac{\text{GSOS laws } \; F^*(G \times id) \overset{\lambda}{\Longrightarrow} (G \times id)F^*}{\text{natural transformations } \; F(G \times id) \underset{\rho}{\Longrightarrow} GF^*}$$

We use an overline-notation $\lambda \mapsto \overline{\lambda}$, $\rho \mapsto \overline{\rho}$ for this correspondence, in both directions.

Correspondingly, $F^*X \xrightarrow{a} X \xrightarrow{b} GX$ is a λ-model (as in Definition 2) if and only if the following diagram commutes.

$$\begin{tikzcd}
FX \arrow[r, "a \circ \sigma"] \arrow[d, "F(\langle b, id\rangle)"'] & X \arrow[r, "b"] & GX \\
F(GX \times X) \arrow[rr, "\overline{\lambda}"'] & & GF^*X \arrow[u, "Ga"']
\end{tikzcd}$$

In view of this result, we shall often also call a natural transformation $F(G \times \text{id}) \Rightarrow GF^*$ a GSOS law.

Proof. We only describe the constructions, and leave the details to the interested readers. For the correspondence between GSOS laws and natural transformations, first assume a GSOS law λ: $F^*(G \times \text{id}) \Rightarrow (G \times \text{id})F^*$. It gives rise to a natural transformation:

$$\overline{\lambda}_X = \Big(F(GX \times X) \xrightarrow{\sigma} F^*(GX \times X) \xrightarrow{\lambda_X} GF^*X \times F^*X \xrightarrow{\pi_1} GF^*X\Big)$$

Conversely, for ρ: $F(G \times \text{id}) \Rightarrow GF^*$ we define a distributive law $\overline{\rho} = \langle \rho_1, \rho_2 \rangle$: $F^*(G \times \text{id}) \Rightarrow (G \times \text{id})F^*$ where $\rho_2 = F^*(\pi_2)$ and ρ_1 is defined by recursion (following Theorem 2) in:

$$\begin{tikzcd}
\begin{pmatrix} (GX \times X) + \\ F\big(F^*(GX \times X)\big) \end{pmatrix} \arrow[r, dashed, "\text{id} + F(\langle \rho_1, \text{id} \rangle)"] \arrow[d, "{[\eta, \tau]}"', "\cong"] & \begin{pmatrix} (GX \times X) + \\ F\big(GF^*X \times F^*(GX \times X)\big) \end{pmatrix} \arrow[d, "{[G\eta \circ \pi_1, G\mu \circ \rho \circ F(\text{id} \times F^*(\pi_2))]}"] \\
F^*(GX \times X) \arrow[r, dashed, "\rho_1"'] & GF^*X
\end{tikzcd}$$

The equivalence with respect to models amounts for $F^*X \xrightarrow{a} X \xrightarrow{b} GX$ to:

$$\begin{tikzcd}
F^*X \arrow[r, "a"] \arrow[d, "F^*(\langle b, \mathrm{id}\rangle)"'] & X \arrow[r, "\langle b, \mathrm{id}\rangle"] & (G \times \mathrm{id})X \\
F^*(GX \times X) \arrow[rr, "\lambda"'] & & (G \times \mathrm{id})F^*X \arrow[u, "Ga \times a"']
\end{tikzcd}
\quad \text{iff} \quad
\begin{tikzcd}
FX \arrow[r, "a \circ \sigma"] \arrow[d, "F(\langle b, \mathrm{id}\rangle)"'] & X \arrow[r, "b"] & GX \\
F(GX \times X) \arrow[rr, "\overline{\lambda}"'] & & GF^*X \arrow[u, "Ga"']
\end{tikzcd}$$

The direction from left to right is straightforward, and the reverse direction requires the use of uniqueness in recursion. □

Example 5. The regular expression functor $\mathcal{R}(X) = 1 + 1 + (X \times X) + (X \times X) + X$ and the deterministic automaton functor $\mathcal{D}(X) = X^A \times 2$ are connected via a GSOS law:

$$\begin{array}{rcl}
\mathcal{R}(X^A \times 2 \times X) & \xrightarrow{\quad\rho_X\quad} & \mathcal{R}^*(X)^A \times 2 \\
0 & \xmapsto{\text{zero}} & (\lambda a \in A.\, 0, 0) \\
1 & \xmapsto{\text{one}} & (\lambda a \in A.\, 0, 1) \\
\langle(\varphi_1, b_1, x_1), (\varphi_2, b_2, x_2)\rangle & \xmapsto{\text{plus}} & (\lambda a \in A.\, \varphi_1(a) + \varphi_2(a), b_1 \vee b_2) \\
\langle(\varphi_1, b_1, x_1), (\varphi_2, b_2, x_2)\rangle & \xmapsto{\text{product}} & (\lambda a \in A.\, \varphi_1(a) \cdot x_2 + b_1 \cdot \varphi_2(a), b_1 \wedge b_2) \\
(\varphi, b, x) & \xmapsto{\text{star}} & (\lambda a \in A.\, \varphi(a) \cdot x^*, 1).
\end{array}$$

One recognises the clauses/rules for D and E as described in the previous subsection. Their format can thus be expressed via a GSOS law; see [5] for more information about such correspondences. We shall illustrate that this law is fundamental, in the sense that it induces familiar structure (and associated results) on regular expressions.

There are a number of general results about GSOS laws that put our running example in perspective. We shall concentrate on these results first, and return to the example of regular expressions at the end of this subsection. The next two results are the analogues for GSOS laws of Lemmas 1 and 2. The proof of the second one uses a form of recursion for Eilenberg-Moore algebras.

Lemma 4. *If we have a GSOS law $\lambda: T(G \times id) \Rightarrow (G \times id)T$, then a final coalgebra $\zeta: Z \xrightarrow{\cong} GZ$ induces a final λ-model with algebra $\alpha: TZ \to Z$ defined by coinduction:*

$$\begin{tikzcd}
GTZ \arrow[r, dashed, "G\alpha"] & GZ \\
T(GZ \times Z) \arrow[u, "\pi_1 \circ \lambda"] & \\
TZ \arrow[u, "T(\langle \zeta, id\rangle)"] \arrow[r, dashed, "\alpha"'] & Z \arrow[uu, "\cong", "\zeta"']
\end{tikzcd}$$

Proof. By uniqueness one obtains that α is an Eilenberg-Moore algebra. By construction the pair (α, ζ) is a λ-model. It is final because for an arbitrary λ-model $TX \xrightarrow{a} X \xrightarrow{b} GX$ the induced coalgebra map $X \to Z$ is also an algebra map—again proven by uniqueness. □

Lemma 5. *Given a GSOS law $\lambda: T(G \times id) \Rightarrow (G \times id)T$ there is a bijective correspondence between GT-coalgebras and λ-models with free algebra:*

$$\frac{\text{“equations”} \quad X \xrightarrow{e} GTX}{\lambda\text{-models} \quad TTX \xrightarrow[\mu]{} TX \xrightarrow[d]{} GTX}$$

and also between corresponding solutions and bialgebra maps.

Proof. The proof relies on the following “recursion” version of freeness for Eilenberg-Moore algebras: for each $f: X \to Y$ and $a: T(Y \times TX) \to Y$ there is a unique map g in:

$$\begin{tikzcd} T^2X \arrow[r, dashed, "T(\langle g, \mathrm{id}\rangle)"] \arrow[d, "\mu"'] & T(Y \times TX) \arrow[d, "a"] \\ TX \arrow[r, dashed, "g"'] & Y \end{tikzcd} \qquad \text{with} \quad g \circ \eta = f \tag{7}$$

provided that a satisfies $a \circ \eta = \pi_1$ and $a \circ \mu = a \circ T(\langle a, \mu \circ T(\pi_2)\rangle)$. The proof of this property is much like the proof of Theorem 2 and left to the reader.

We only describe the correspondence between equations and GSOS models, and leave the rest to the interested reader. Given $e: X \to GTX$ define $\overline{e}$ via (7) in:

$$\begin{tikzcd} T^2X \arrow[r, dashed, "T(\langle \overline{e}, \mathrm{id}\rangle)"] \arrow[d, "\mu"'] & T(GTX \times TX) \arrow[d, "G\mu \circ \pi_1 \circ \lambda"] \\ TX \arrow[r, dashed, "\overline{e}"'] & GTX \end{tikzcd} \qquad \text{with} \quad \overline{e} \circ \eta = e$$

By construction this forms a λ-model. In the reverse direction, given $d: TX \to GTX$ one takes $\overline{d} = d \circ \eta: X \to GTX$. Then $\overline{\overline{e}} = \overline{e} \circ \eta = e$. And $\overline{\overline{d}} = d$ follows by uniqueness, using that (μ, d) is a GSOS model: $G\mu \circ \pi_1 \circ \lambda \circ T(\langle d, \mathrm{id}\rangle) = d \circ \mu$. □

Remark 1. 1. If we apply the construction of the previous lemma starting from a law $\rho: F(G \times \mathrm{id}) \Rightarrow GF^*$ like in Proposition 5, then the GSOS model $F^*F^*X \xrightarrow{\mu} F^*X \xrightarrow{d} GF^*X$ associated with an equation $e: X \to GF^*X$ can be described via recursion (like in Theorem 2) as:

$$\begin{tikzcd}
X + F(F^*X) \arrow[r, dashed, "\mathrm{id} + F(\langle d, \mathrm{id}\rangle)"] \arrow[d, "{[\eta,\tau]}"', "\cong"] & X + F(GF^*X \times F^*X) \arrow[d, "{[e, G\mu \circ \rho]}"] \\
F^*X \arrow[r, dashed, "d"'] & GF^*X
\end{tikzcd}$$

This will be used later.

2. In [35, Proposition 5.1] it is shown that a (GSOS) law $\rho\colon F(G\times\mathrm{id}) \Rightarrow GF^*$ induces a lifting of the free monad F^* to the category $\mathbf{CoAlg}(G)$. The construction uses the previous point: it takes a coalgebra $b\colon X \to GX$ to the coalgebra-part of the bialgebra corresponding to the equation $G(\eta) \circ b\colon X \to GF^*X$.

With all these general GSOS results in place we are finally in a position to analyse the situation of regular expressions and languages, using the GSOS law from Example 5.

Theorem 3. *1. The "equation" $A \to \mathcal{D}(\mathcal{R}^*(A))$ that is given by the two maps*

$$\begin{array}{ccc} A \longrightarrow \mathcal{R}^*(A)^A & \qquad & A \longrightarrow 2 \\ a \longmapsto \lambda b \in A.\, \textit{if}\ b = a\ \textit{then}\ 1\ \textit{else}\ 0 & & a \longmapsto 0 \end{array}$$

corresponds by Lemma 5 to the free algebra and Brzozowski automaton structure on the set $\mathit{Re} = \mathcal{R}^*(A)$ of regular expressions:

$$\mathcal{R}^*(\mathit{Re}) \xrightarrow{\mu} \mathit{Re} \xrightarrow{\langle D, E\rangle} \mathit{Re}^A \times 2$$

2. The final $\mathcal{D}$-coalgebra $\mathcal{L}(A) \xrightarrow{\cong} \mathcal{L}(A)^A \times 2$ of languages yields by Lemma 4 the final bialgebra:

$$\mathcal{R}^*(\mathcal{L}(A)) \longrightarrow \mathcal{L}(A) \xrightarrow{\cong} \mathcal{L}(A)^A \times 2$$

with the standard algebra of regular expressions.

3. The interpretation $[\![-]\!]\colon \mathit{Re} \to \mathcal{L}(A)$ introduced via freeness in (3) can also be obtained as $\mathsf{beh}\colon \mathit{Re} \to \mathcal{L}(A)$ by finality using the previous two points.

4. Bisimilarity between regular expressions is a congruence: $s \leftrightarroweq s'$ and $t \leftrightarroweq t'$ implies $s + t \leftrightarroweq s' + t'$, $s \cdot t \leftrightarroweq s' \cdot t'$ and $s^* \leftrightarroweq s'^*$.

Proof. 1. Let's write $e\colon A \to \mathsf{Re}^A \times 2$ for the equation. We need to check that the Brzozowski structure $\langle D, E\rangle$ from Subsection 4.2 fits in the description in Remark 1.(1), *i.e.* that the following diagram commutes,

$$\begin{tikzcd}
A + \mathcal{R}(\mathsf{Re}) \arrow[r, "\mathrm{id} + \mathcal{R}(\langle\langle D, E\rangle, \mathrm{id}\rangle)"] \arrow[d, "{[\eta,\tau]}"', "\cong"] & A + \mathcal{R}(\mathsf{Re}^A \times 2 \times \mathsf{Re}) \arrow[d, "{[e, (\mu^A \times \mathrm{id}) \circ \rho]}"] \\
\mathsf{Re} \arrow[r, "\langle D, E\rangle"'] & \mathsf{Re}^A \times 2
\end{tikzcd}$$

where ρ is as described in Example 5. This diagram commutes because the Brzozowski structure $\langle D, E\rangle$ precisely follows the GSOS law ρ.

2. Similarly we need to show that the standard interpretation $\alpha\colon \mathcal{R}(\mathcal{L}(A)) \to \mathcal{L}(A)$ yields a commuting diagram in Lemma 4. This means that $\langle \delta, \varepsilon\rangle \circ \alpha = (\alpha^A \times \mathrm{id}) \circ \rho \circ \mathcal{R}(\langle\langle \delta, \varepsilon\rangle, \mathrm{id}\rangle)$, which can be checked easily—where $\langle \delta, \varepsilon\rangle$ is the final coalgebra structure on $\mathcal{L}(A)$.
3. Obvious, since $[\![-]\!]$ is also a map of coalgebras.
4. The bisimilarity relation $\underline{\leftrightarrow} \rightarrowtail \mathsf{Re} \times \mathsf{Re}$ is the equaliser e at the bottom row below, because of Proposition 2 and because $[\![-]\!] = \mathsf{beh}$ by the previous point.

$$\begin{tikzcd}
\mathcal{R}^*(\underline{\leftrightarrow}) \arrow[r, "d"] \arrow[d, dashed] & \mathcal{R}^*(\mathsf{Re}) \times \mathcal{R}^*(\mathsf{Re}) \arrow[r, shift left, "\mathcal{R}^*([\![-]\!]) \circ \pi_1"] \arrow[r, shift right, "\mathcal{R}^*([\![-]\!]) \circ \pi_2"'] \arrow[d, "\mu \times \mu"'] & \mathcal{R}^*(\mathcal{L}(A)) \arrow[d] \\
\underline{\leftrightarrow} \arrow[r, tail, "e"'] & \mathsf{Re} \times \mathsf{Re} \arrow[r, shift left, "{[\![-]\!] \circ \pi_1}"] \arrow[r, shift right, "{[\![-]\!] \circ \pi_2}"'] & \mathcal{L}(A)
\end{tikzcd}$$

The map $d = \langle \mathcal{R}^*(\pi_1 \circ e), \mathcal{R}^*(\pi_2 \circ e)\rangle$ induces an algebra structure on the relation $\underline{\leftrightarrow}$, as indicated. This makes $\underline{\leftrightarrow}$ a congruence. $\square$

The map $[\![-]\!]\colon \mathsf{Re} \to \mathcal{L}(A)$ defined by initiality is by construction "compositional", in the sense that it preserves the operations. This map describes what may be called the denotational semantics of regular expressions. In contrast, the map $\mathsf{beh}\colon \mathsf{Re} \to \mathcal{L}(A)$ obtained by finality describes the operational semantics, because it is induced by the dynamical (coalgebra) structure on regular expressions. The equality of denotational $[\![-]\!]$ and operational beh semantics in point 3 of the previous theorem says in particular that the operational semantics is compositional, so that for instance the behaviour of a sum expression is the sum of the behaviours of the two summands. Many coincidences of operational and denotational semantics are described in more concrete form in [3].

5 Regular Expressions with Equations

An equational logic for regular expressions is formulated by Kozen in [23], for which a completeness theorem is proved. An alternative proof of completenes (again by Kozen) is given in [24]. Here we shall give a coalgebraic review of the situation, which leads to a third completeness proof. It is similar, but shorter, than the proof in [24].

Throughout this section we fix a *finite* alphabet A. We shall indicate where we need this finiteness (in Definition 4).

The definition of **Kleene algebra** from [23] involves a particular formulation of the rules for the star operation. It requires for an algebra $[0, 1, +, \cdot, (-)^*]\colon \mathcal{R}(Y) \to Y$ that $(Y, 0, 1, +, \cdot)$ is an idempotent semiring in which the star axioms and rules in point 2 below hold.

One can also turn the set Re of regular expressions into a Kleene algebra via a suitable quotient. For clarity we shall use a special symbol $\doteq\, \subseteq \mathsf{Re} \times \mathsf{Re}$ for the least relation satisfying the next three points.

1. $(\mathsf{Re}, +, 0, \cdot, 1)$ is an idempotent semiring, *i.e.*
   - $(\mathsf{Re}, +, 0)$ is an idempotent commutative monoid, in which one defines a partial order by $s \leq t \Longleftrightarrow s + t \doteq t$.
   - $(\mathsf{Re}, \cdot, 1)$ is a monoid, where $\cdot$ preserves the additive monoid structure $+, 0$ in both arguments: $s \cdot (t + r) \doteq (s \cdot t) + (s \cdot r)$ and $(t + r) \cdot s \doteq (t \cdot s) + (r \cdot s)$, and also $s \cdot 0 \doteq 0$ and $0 \cdot s \doteq 0$.
2. The star inequalities and rules:

$$1 + s \cdot s^* \leq s^* \qquad 1 + s^* \cdot s \leq s^* \qquad \frac{s + t \cdot x \leq x}{t^* s \leq x} \qquad \frac{s + x \cdot t \leq x}{s \cdot t^* \leq x}$$

3. Axioms and rules making $\doteq$ a congruence, *i.e.* an equivalence relation preserved by the operations: $s \doteq s'$ and $t \doteq t'$ implies $s + t \doteq s' + t'$, $s \cdot t \doteq s' \cdot t'$ and $s^* = s'^*$.

We shall write $\mathsf{Re}/{\doteq}$ for the set of regular expressions modulo $\doteq$. By construction it forms a Kleene algebra. As usual, we often simply write s for the equivalence class $[s] = \{t \in \mathsf{Re} \mid t \doteq s\} \in \mathsf{Re}/{\doteq}$.

Of the many results that can be derived in Kleene algebras we shall need the following ones.

Lemma 6. *In an arbitrary Kleene algebra one has:*

1. $1 + s \cdot s^* = s^*$;
2. $s \cdot x = x \cdot t$ *implies* $s^* \cdot x = x \cdot t^*$.

And each term $s \in \mathsf{Re}$ *satisfies* $s \geq \sum_{a \in A} a \cdot D_a(s) + E(s)$.

Proof. The inequality $1 + s \cdot s^* \leq s^*$ is one of the star axioms. And $s^* \leq 1 + s \cdot s^*$ is obtained by applying a star rule to the inequality $1 + s \cdot x \leq x$ for $x = 1 + s \cdot s^*$.

For the second point it suffices to show: if $s \cdot x \leq x \cdot t$ then $s^* \cdot x \leq x \cdot t^*$. The latter can be obtained via a star rule from $x + s \cdot (x \cdot t^*) \leq x \cdot t^*$, which follows from the assumption $s \cdot x \leq x \cdot t$.

The final inequality $s \geq \sum_{a \in A} a \cdot D_a(s) + E(s)$ is obtained by induction on the structure of $s \in \mathsf{Re}$. □

The following two standard lemmas (see *e.g.* [9,24,31]) must be made explicit first.

Lemma 7. *1. The derivative operation on regular expressions preserves equality, i.e. satisfies* $s \doteq t \Longrightarrow D_a(s) \doteq D_a(t)$*, for each letter* a*. Similarly,* $s \doteq t \Longrightarrow E(s) = E(t)$.

The Brzozowski coalgebra structure $\langle D, E \rangle \colon \mathsf{Re} \to \mathsf{Re}^A \times 2$ *thus restricts to* $\langle D, E \rangle \colon (\mathsf{Re}/{\doteq}) \to (\mathsf{Re}/{\doteq})^A \times 2$*, making the quotient map* $[-] \colon \mathsf{Re} \twoheadrightarrow \mathsf{Re}/{\doteq}$ *a homomorphism of coalgebras.*

2. *If $s \doteq t$ then $[\![s]\!] = [\![t]\!]$, i.e. s, t yield the same languages. Hence the diagram (3) of bialgebras can be further refined by taking images:*

$$\begin{tikzcd}
\mathcal{R}(\mathit{Re}) \arrow[r] \arrow[d, "\mu"'] & \mathcal{R}(\mathit{Re}/\doteq) \arrow[r] \arrow[d] & \mathcal{R}(\mathcal{L}_r(A)) \arrow[r] \arrow[d] & \mathcal{R}(\mathcal{L}(A)) \arrow[d] \\
\mathit{Re} \arrow[r, two heads] \arrow[d, "\langle D, E\rangle"'] & \mathit{Re}/\doteq \arrow[r, two heads, "{[\![-]\!]}"] \arrow[d, "\langle D, E\rangle"'] & \mathcal{L}_r(A) \arrow[r, tail] \arrow[d] & \mathcal{L}(A) \arrow[d, "\cong"', "\langle \delta, \varepsilon\rangle"] \\
\mathit{Re}^A \times 2 \arrow[r] & (\mathit{Re}/\doteq)^A \times 2 \arrow[r] & \mathcal{L}_r(A)^A \times 2 \arrow[r] & \mathcal{L}(A)^A \times 2
\end{tikzcd} \qquad (8)$$

where $\mathcal{L}_r(A)$ is the subset of regular (also called rational) languages obtained as interpretation $[\![s]\!]$ of a regular expression s.

The completeness result of [24] states that the (restricted) homomorphism $[\![-]\!]$ in the middle of (8) is an isomorphism, see Theorem 4 below.

Proof. By induction on the length of derivations of $\doteq$. □

The derivative operation $D\colon \mathsf{Re} \to \mathsf{Re}^A$ yields a multiple derivative $D^\star\colon \mathsf{Re} \to \mathsf{Re}^{A^\star}$ like in (1). Similarly we get $D^\star\colon \mathsf{Re}/\doteq \to (\mathsf{Re}/\doteq)^{A^\star}$ for expressions modulo equations. We shall also use the subscript notation in these situations (and drop the star), so that $D_\sigma(s) = D^\star(s)(\sigma)$ with cases $D_{\langle\rangle}(s) = s$ and $D_{a\cdot\sigma}(s) = D_\sigma(D_a(s))$.

Lemma 8. *Expressions modulo equations have only finitely many successors: for each term/state $s \in \mathit{Re}$ the set $\Diamond(s) = \{D_\sigma(s) \mid \sigma \in A^\star\} \subseteq \mathit{Re}/\doteq$ of successors of s in the coalgebra $\mathit{Re}/\doteq \to (\mathit{Re}/\doteq)^A \times 2$ is finite.*

Proof. The basic terms are easy, since $\Diamond(0) = \{D_\sigma(0) \mid \sigma \in A^\star\} = \{0\}$, $\Diamond(1) = \{1, 0\}$ and $\Diamond(a) = \{a, 1, 0\}$. For the compound terms one first proves the following equations.

$$\begin{aligned}
D_\sigma(s+t) &\doteq D_\sigma(s) + D_\sigma(t) \\
D_\sigma(s \cdot t) &\doteq D_\sigma(s) \cdot t + \sum_{\tau\cdot\rho=\sigma;\rho\neq\langle\rangle} E(D_\tau(s)) \cdot D_\rho(t) \\
D_\sigma(s^*) &\doteq D_\sigma(1) + D_\sigma(s) \cdot s^* + \sum_{\tau\cdot\rho=\sigma;\tau,\rho\neq\langle\rangle} E(D_\tau(s)) \cdot D_\rho(s^*).
\end{aligned}$$

These equations are obtained by induction on the length of $\sigma \in A^\star$.

If we now write $\#\Diamond(s) \in \mathbb{N}$ for the number of elements of $\Diamond(s)$, then:

$$\begin{aligned}
\#\Diamond(0) &= 1 & \qquad \#\Diamond(s+t) &\leq \#\Diamond(s) \cdot \#\Diamond(t) \\
\#\Diamond(1) &= 1 & \#\Diamond(s\cdot t) &\leq \#\Diamond(s) \cdot 2^{\#\Diamond(t)} \\
\#\Diamond(a) &= 3 & \#\Diamond(s^*) &\leq \#\Diamond(s) \cdot 2^{\#\Diamond(s)}.
\end{aligned}$$

Hence we can conclude that each subset $\Diamond(s) \subseteq \mathsf{Re}/\doteq$ is finite. □

Next we shall define a category in which the Brzozowski automaton on $\mathsf{Re}/\doteq$ lives.

Definition 3. *We write* $\mathbf{DetAut}_{\mathit{fb}}$ *for the category of deterministic automata with finite behaviour. Objects are coalgebras* $\langle\delta,\varepsilon\rangle\colon X \to X^A \times 2$ *such that for each state* $x \in X$ *the set of successors* $\Diamond(x) = \{\delta^\star(x)(\sigma) \mid \sigma \in A^\star\} \subseteq X$ *is finite. Maps in* $\mathbf{DetAut}_{\mathit{fb}}$ *are the usual homomorphisms of coalgebras.*

(Notice that we leave the set A *of inputs implicit in the notation.)*

It is not hard to see that if $\big(X \to X^A \times 2\big) \overset{f}{\twoheadrightarrow} \big(Y \to Y^A \times 2\big)$ is a surjective coalgebra homomorphism where $X \to X^A \times 2$ is in $\mathbf{DetAut}_{\mathit{fb}}$, then so is $Y \to Y^A \times 2$. The reason is that $f(\delta^\star(x)(\sigma)) = \delta^\star(f(x))(\sigma)$, and so $\Diamond(f(x)) \subseteq f[\Diamond(x)]$. Hence the automaton structure $\mathcal{L}_r(A) \to \mathcal{L}_r(A)^A \times 2$ from (8) is also in the category $\mathbf{DetAut}_{\mathit{fb}}$, via the surjection $[\![-]\!]\colon \mathsf{Re}/\doteq \twoheadrightarrow \mathcal{L}_r(A)$.

A basic property of Kleene algebras is that an inequality $x \geq s \cdot x + t$ has a least solution $s^* t$, via the star rule and via $s^* \cdot t \geq s \cdot \big(s^* \cdot t\big) + t$. Even stronger, the latter is actually an equality, since $s \cdot \big(s^* \cdot t\big) + t \doteq \big(s \cdot s^* + 1\big) \cdot t \doteq s^* \cdot t$.

This can be generalised to equations in multiple variables, using the standard fact that square matrices in Kleene algebras form again Kleene algebras, and can be used to solve equations, see [23, Section 3]. A system of n equations:

$$x_i = s_{i1}x_1 + \cdots + s_{in}x_n + t_i$$

has a least solution that can be described as vector $S^* \cdot T$ where

$$S = \begin{pmatrix} s_{11} & \cdots & s_{1n} \\ & \vdots & \\ s_{n1} & \cdots & s_{nn} \end{pmatrix} \qquad \text{and} \qquad T = \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix}$$

describe the equation as $\overrightarrow{x} = S \cdot \overrightarrow{x} + T$ and the star operation S^* is in the Kleene algebra of $n \times n$ matrices.

Definition 4. *Let* $\langle\delta,\varepsilon\rangle\colon X \to X^A \times 2$ *be an arbitrary coalgebra with finite behaviour (i.e. an object of* $\mathbf{DetAut}_{\mathit{fb}}$*). With each state* $x \in X$ *we associate a term* $\ulcorner x \urcorner \in \mathsf{Re}/\doteq$ *in the following way.*

By assumption $\Diamond(x)$ *is finite, say* $\Diamond(x) = \{x_1, x_2, \ldots, x_n\}$ *where* $x_1 = x$*. An* $n \times n$ *transition matrix* $S_x = (s_{ij})$ *and an output vector* $T_x = (t_i)$ *over* $\mathsf{Re}/\doteq$ *are constructed with elements*

$$s_{ij} = \textstyle\sum\{a \in A \mid \delta(x_i)(a) = x_j\} \qquad \textit{and} \qquad t_i = \varepsilon(x_i).$$

We then take $\ulcorner x \urcorner \in \mathsf{Re}/\doteq$ *to be the first element of the least solution* $S_x^* \cdot T_x$ *of the associated equations. More formally, as vector product,* $\ulcorner x \urcorner = (1\,0\,\ldots\,0) \cdot S_x^* \cdot T_x$*.*

The sum $\sum$ in this definition exists because we have assumed that the alphabet A is finite. The sum over an empty set is 0, as usual. Notice that the ordering of the elements in $\Diamond(x)$ is not relevant.

One can understand S as a big square matrix $X \times X \to \mathsf{Re}/\doteq$ defined by $(x, x') \mapsto \sum\{a \mid \delta(a)(x) = x'\}$ like in [24]. The matrix S_x in the definition is then the restriction of S to $\{x_1, \ldots, x_n\} \subseteq X$.

Lemma 9. *The mapping $x \mapsto \ulcorner x \urcorner$ is a homomorphism of coalgebras.*

Proof. Consider $x = x_1 \in X$ as in Definition 4. We need to show:

$$E(\ulcorner x \urcorner) = \varepsilon(x) \quad \text{and} \quad D(\ulcorner x \urcorner)(a) = \ulcorner \delta(x)(a) \urcorner.$$

We notice that the vector of solutions in $\mathsf{Re}/\doteq$ can be described as $\overrightarrow{\ulcorner x_i \urcorner}$. Hence

$$\ulcorner x_1 \urcorner \doteq s_{11} \cdot \ulcorner x_1 \urcorner + \cdots + s_{1n} \cdot \ulcorner x_n \urcorner + \varepsilon(x_1),$$

where each s_{ij} is a sum of atoms/letters from A. Thus:

$$\begin{aligned}
E(\ulcorner x \urcorner) &= E(s_{11} \cdot \ulcorner x_1 \urcorner + \cdots + s_{1n} \cdot \ulcorner x_n \urcorner + \varepsilon(x_1)) \\
&= \big(E(s_{11}) \wedge E(\ulcorner x_1 \urcorner)\big) \vee \cdots \vee \big(E(s_{1n}) \wedge E(\ulcorner x_n \urcorner)\big) \vee E(\varepsilon(x_1)) \\
&= \big(0 \wedge E(\ulcorner x_1 \urcorner)\big) \vee \cdots \vee \big(0 \wedge E(\ulcorner x_n \urcorner)\big) \vee \varepsilon(x_1) \\
&= \varepsilon(x_1) \\
D(\ulcorner x \urcorner)(a) &= D(s_{11} \cdot \ulcorner x_1 \urcorner + \cdots + s_{1n} \cdot \ulcorner x_n \urcorner + \varepsilon(x_1))(a) \\
&= D(s_{11} \cdot \ulcorner x_1 \urcorner)(a) + \cdots + D(s_{1n} \cdot \ulcorner x_n \urcorner)(a) \\
&= D(s_{11})(a) \cdot \ulcorner x_1 \urcorner + E(s_{11}) \cdot D(\ulcorner x_1 \urcorner)(a) + \cdots + \\
&\qquad D(s_{1n})(a) \cdot \ulcorner x_n \urcorner + E(s_{1n}) \cdot D(\ulcorner x_n \urcorner)(a) \\
&= D(s_{11})(a) \cdot \ulcorner x_1 \urcorner + \cdots + D(s_{1n})(a) \cdot \ulcorner x_n \urcorner \\
&= \ulcorner x_i \urcorner \ \text{ if } \ \delta(x)(a) = x_i \\
&= \ulcorner \delta(x)(a) \urcorner.
\end{aligned}$$ □

By finality this homomorphism $\ulcorner - \urcorner$ yields a commuting diagram:

$$\begin{tikzcd}
\mathsf{Re}/\doteq \arrow[r, "{[\![-]\!]}"] & \mathcal{L}_r(A) \arrow[r, hook] & \mathcal{L}(A) \\
X \arrow[u, "\ulcorner - \urcorner"] \arrow[urr, "\mathsf{beh}"'] & &
\end{tikzcd}$$

In particular, when $X = \mathcal{L}_r(A)$, we see that $\ulcorner - \urcorner$ is a section of $[\![-]\!]$.

Corollary 1. *The coalgebra $\mathcal{L}_r(A) \to \mathcal{L}_r(A)^A \times 2$ is final in the category* ***DetAut**$_{fb}$*.

Proof. Given a coalgebra $X \to X^A \times 2$ in **DetAut**$_{fb}$ there is a composition of homomorphisms $[\![-]\!] \circ \ulcorner - \urcorner : X \to \mathsf{Re}/\doteq \to \mathcal{L}_r(A)$. If we have two homomorphisms $f, g: X \to \mathcal{L}_r(A)$, then by postcomposition with the inclusion $\mathcal{L}_r(A) \hookrightarrow \mathcal{L}(A)$ we get two homomorphisms to the final $(-)^A \times 2$ coalgebra—which must thus be equal. Hence also $f = g$. □

At this stage we can obtain Kleene's theorem [21], as point 2 below. Point 1 is [31, Theorem 10.1].

Corollary 2. *1. A language $L \in \mathcal{L}(A)$ is regular—i.e. belongs to $\mathcal{L}_r(A) \hookrightarrow \mathcal{L}(A)$—if and only if the set of derivatives $\diamondsuit(L)$ is finite.*

2. A language $L \in \mathcal{L}(A)$ is regular if and only if it is accepted by a finite automaton (i.e. an automaton with a finite state space).

Proof. 1. If $L \in \mathcal{L}_r(A)$, then $\diamondsuit(L)$ is finite because $\mathcal{L}_r(A)$ is in $\mathbf{DetAut}_{\mathit{fb}}$. Conversely, if $\diamondsuit(L)$ is finite, then $\diamondsuit(L)$ can be considered as a subcoalgebra $\diamondsuit(L) \hookrightarrow \mathcal{L}(A)$ that belongs to $\mathbf{DetAut}_{\mathit{fb}}$. Hence it factors as $\diamondsuit(L) \hookrightarrow \mathcal{L}_r(A)$.

2. If L is regular, then $\diamondsuit(L)$ is itself a finite automaton (by 1) with initial state $L \in \diamondsuit(L)$ whose behaviour $\mathsf{beh}(L) \in \mathcal{L}_r(A)$ is L itself. Conversely, if $L \in \mathcal{L}(A)$ is $\mathsf{beh}(x)$ for an initial state $x \in X$ of a finite automaton, then $\diamondsuit(x)$ is finite, so $L = \mathsf{beh}(x) \in \mathcal{L}_r(A)$ because $\mathcal{L}_r(A)$ is final in $\mathbf{DetAut}_{\mathit{fb}}$. □

The next two lemmas and their proofs are reformulations of results in [24].

Lemma 10. *If $f\colon X \to Y$ is a homomorphism in $\mathbf{DetAut}_{\mathit{fb}}$, then $\ulcorner f(x) \urcorner = \ulcorner x \urcorner$.*

Proof. If $\diamondsuit(x) = \{x_1, \ldots, x_n\}$ where $x_1 = x$, then $\diamondsuit(f(x)) = \{f(x_1), \ldots, f(x_n)\}$. The latter set may be smaller than the former. We shall consider the following three square matrices $S, \widehat{f}, S^f\colon \{1, \ldots, n\}^2 \to \mathsf{Re}/\doteq$.

$$S_{ij} = \sum\{a \mid x_i \xrightarrow{a} x_j\} \qquad \big(\widehat{f}\big)_{ij} = \begin{cases} 1 \text{ if } f(x_i) = f(x_j) \\ 0 \text{ otherwise.} \end{cases}$$
$$\big(S^f\big)_{ij} = \sum\{a \mid f(x_i) \xrightarrow{a} f(x_j)\}$$

Then there is an equality of matrix products:

$$\begin{aligned}
\big(S \cdot \widehat{f}\big)_{ij} &= \textstyle\sum_k S_{ik} \cdot (\widehat{f})_{kj} \\
&= \textstyle\sum\{\sum\{a \mid x_i \xrightarrow{a} z\} \mid z \in \diamondsuit(x) \wedge f(z) = f(x_j)\} \\
&= \textstyle\sum\{a \mid \exists z \in \diamondsuit(x).\, x_i \xrightarrow{a} z \wedge f(z) = f(x_j)\} \\
&= \textstyle\sum\{a \mid f(x_i) \xrightarrow{a} f(x_j)\} \\
&= \textstyle\sum\{a \mid \exists z \in \diamondsuit(x).\, f(z) = f(x_i) \wedge f(z) \xrightarrow{a} f(x_j)\} \\
&= \textstyle\sum\{\sum\{a \mid f(z) \xrightarrow{a} f(x_j)\} \mid z \in \diamondsuit(x) \wedge f(z) = f(x_i)\} \\
&= \textstyle\sum_k (\widehat{f})_{ik} \cdot (S^f)_{kj} \\
&= \big(\widehat{f} \cdot S^f\big)_{ij}.
\end{aligned}$$

Lemma 6 (2) now yields $S^* \cdot \widehat{f} = \widehat{f} \cdot (S^f)^*$. If we write T for the vector of elements $\varepsilon(x_i) = \varepsilon(f(x_i))$, then $\widehat{f} \cdot T = T$, since

$$\begin{aligned}
\big(\widehat{f} \cdot T\big)_i = \textstyle\sum_k (\widehat{f})_{ik} \cdot T_k &= \textstyle\sum\{\varepsilon(x_k) \mid f(x_k) = f(x_i)\} \\
&= \textstyle\sum\{\varepsilon(x_i) \mid f(x_k) = f(x_i)\} = \varepsilon(x_i) = T_i.
\end{aligned}$$

Hence:

$$\begin{aligned}
\ulcorner x \urcorner = (1\,0 \ldots 0) \cdot S^* \cdot T &= (1\,0 \ldots 0) \cdot S^* \cdot \widehat{f} \cdot T \\
&= (1\,0 \ldots 0) \cdot \widehat{f} \cdot (S^f)^* \cdot T \\
&= (\widehat{f}_{11}, \ldots, \widehat{f}_{1n}) \cdot (S^f)^* \cdot T \\
&= \textstyle\sum\{\ulcorner f(x_i) \urcorner \mid f(x_i) = f(x_1)\} = \ulcorner f(x) \urcorner.
\end{aligned}$$

The last equation holds even though S^f may be "too big" a matrix, describing too many equations. These additional equations however are repeated equations, which do not influence the least solution. $\square$

Lemma 11. *The homomorphism* $\ulcorner - \urcorner \colon \mathsf{Re}/\doteq \to \mathsf{Re}/\doteq$ *is the identity.*

Proof. We first establish the following points.

1. $\ulcorner s \urcorner \leq s$, for $s \in \mathsf{Re}/\doteq$;
2. $\ulcorner 1 \urcorner \doteq 1$ and $\ulcorner 0 \urcorner \doteq 0$;
3. $s \leq t$ implies $\ulcorner s \urcorner \leq \ulcorner t \urcorner$;
4. $s \leq \ulcorner s \urcorner$.

The first and fourth point then yield the required result.

As to the first point, for $s \in \mathsf{Re}/\doteq$ we obtain $\ulcorner s \urcorner$ via the recipe in Definition 4, namely by considering the successor states/derivatives $\diamondsuit(s) = \{s_1, \ldots, s_n\}$ and the associated transition matrix. By Lemma 6 these terms $s_1, \ldots, s_n$ satisfy the defining inequality for $\ulcorner s_i \urcorner$, so that $\ulcorner s_i \urcorner \leq s_i$, since $\ulcorner s_i \urcorner$ is the least solution.

The term 1 has one successor, namely 0. The associated single equation, following Definition 4, is $x = 1$, which has as (least) solution $\ulcorner 1 \urcorner \doteq 1$. Similarly $\ulcorner 0 \urcorner \doteq 0$.

For the third point we consider the product $X = \mathsf{Re}/\doteq \times \mathsf{Re}/\doteq$ as state space with two coalgebra structures $\langle D, E_1 \rangle, \langle D, E_2 \rangle \colon X \to X^A \times 2$, where

$$D(s,t)(a) = (D_a(s), D_a(t)) \qquad E_1(s,t) = E(s) \qquad E_2(s,t) = E(t).$$

The projections $\pi_i \colon X \to \mathsf{Re}/\doteq$ are then homomorphisms from $\langle D, E_i \rangle$ to $\langle D, E \rangle$. Hence Lemma 10 applies. Given elements $s, t \in \mathsf{Re}/\doteq$, let $S = S_{(s,t)}$ be the transition matrix associated with $(s,t) \in X$, and T_1, T_2 be the associated output vectors determined by the output functions E_1, E_2 respectively. Thus, if $s \leq t$, then $E_1(s,t) \leq E_2(s,t)$ and similarly for all successors of (s,t)—because D and E are order preserving. Hence $T_1 \leq T_2$, and thus:

$$\begin{aligned}
\ulcorner s \urcorner = \ulcorner \pi_1(s,t) \urcorner &= \ulcorner (s,t) \urcorner \quad \text{wrt. } \langle D, E_1 \rangle \\
&= (1\,0\,\ldots\,0) \cdot S^* \cdot T_1 \\
&\leq (1\,0\,\ldots\,0) \cdot S^* \cdot T_2 \\
&= \ulcorner (s,t) \urcorner \quad \text{wrt. } \langle D, E_2 \rangle \\
&= \ulcorner \pi_2(s,t) \urcorner \\
&= \ulcorner t \urcorner.
\end{aligned}$$

For the fourth point we proceed like in [24] and prove the stronger statement $\forall t \in \mathsf{Re}.\; s \cdot \ulcorner t \urcorner \leq \ulcorner s \cdot t \urcorner$ by induction on s. We are then done by taking $t = 1$, using point 2.

– $0 \cdot \ulcorner t \urcorner \doteq 0 \doteq \ulcorner 0 \urcorner \doteq \ulcorner 0 \cdot t \urcorner$.
– $1 \cdot \ulcorner t \urcorner \doteq \ulcorner t \urcorner \doteq \ulcorner 1 \cdot t \urcorner$.
– $$\begin{aligned}
\ulcorner b \cdot t \urcorner &\geq \textstyle\sum_{a \in A} a \cdot D_a(\ulcorner b \cdot t \urcorner) + E(\ulcorner b \cdot t \urcorner) && \text{by Lemma 6} \\
&\doteq \textstyle\sum_{a \in A} a \cdot \ulcorner D_a(b \cdot t) \urcorner + E(b \cdot t) && \text{by Lemma 9} \\
&\doteq b \cdot \ulcorner t \urcorner && \text{by point 2.}
\end{aligned}$$

– $$\begin{aligned}(s_1 + s_2) \cdot \ulcorner t \urcorner &\doteq s_1 \cdot \ulcorner t \urcorner + s_2 \cdot \ulcorner t \urcorner \\ &\leq \ulcorner s_1 \cdot t \urcorner + \ulcorner s_2 \cdot t \urcorner && \text{by induction hypothesis} \\ &\leq \ulcorner s_1 \cdot t + s_2 \cdot t \urcorner && \text{by point 3.} \\ &\doteq \ulcorner (s_1 + s_2) \cdot t \urcorner.\end{aligned}$$

– $$\begin{aligned}(s_1 \cdot s_2) \cdot \ulcorner t \urcorner &\doteq s_1 \cdot (s_2 \cdot \ulcorner t \urcorner) \\ &\leq s_1 \cdot \ulcorner s_2 \cdot t \urcorner && \text{by induction hypothesis} \\ &\leq \ulcorner s_1 \cdot (s_2 \cdot t) \urcorner && \text{by induction hypothesis} \\ &\doteq \ulcorner (s_1 \cdot s_2) \cdot t \urcorner.\end{aligned}$$

– Finally, $s^* \cdot \ulcorner t \urcorner \leq \ulcorner s^* \cdot t \urcorner$ is obtained by applying the star rule to:

$$\begin{aligned}\ulcorner t \urcorner + s \cdot \ulcorner s^* \cdot t \urcorner &\leq \ulcorner t \urcorner + \ulcorner s \cdot (s^* \cdot t) \urcorner && \text{by induction hypothesis} \\ &\leq \ulcorner t + s \cdot (s^* \cdot t) \urcorner && \text{by point 3.} \\ &\doteq \ulcorner (1 + s \cdot s^*) \cdot t \urcorner \\ &\doteq \ulcorner s^* \cdot t \urcorner && \text{by Lemma 6.}\end{aligned}$$ □

Theorem 4 (Completeness [23,24]). *The Brzozowski coalgebra $\mathsf{Re}/\!\doteq\, \to \mathsf{Re}/\!\doteq^A \times 2$ is final in* **DetAut**$_{fb}$. *Hence the (bialgebra) homomorphism $[\![-]\!]: \mathsf{Re}/\!\doteq\, \to \mathcal{L}_r(A)$ is an isomorphism.*

Proof. Each object $X \to X^A \times 2$ in **DetAut**$_{fb}$ yields a homomorphism $\ulcorner - \urcorner: X \to \mathsf{Re}/\!\doteq$ by Lemma 9. Suppose we have two homomorphisms $f, g: X \to \mathsf{Re}/\!\doteq$, then by Lemmas 10 and 11 we have:

$$f = \mathrm{id}_{\mathsf{Re}/\doteq} \circ f \overset{(11)}{=} \ulcorner - \urcorner \circ f \overset{(10)}{=} \ulcorner - \urcorner \overset{(10)}{=} \ulcorner - \urcorner \circ g \overset{(11)}{=} \mathrm{id}_{\mathsf{Re}/\doteq} \circ g = g.$$

Final object are unique up-to-isomorphism, so the coalgebra homomorphism $[\![-]\!] = \mathsf{beh}: \mathsf{Re}/\!\doteq\, \to \mathcal{L}_r(A)$ is an isomorphism by Corollary 1. □

Another way to formulate this result is: Kozen's axioms and rules give a complete axiomatisation of bisimilarity for regular expressions. Indeed, for $s, t \in \mathsf{Re}$,

$$\begin{aligned} s \leftrightarroweq t &\Longleftrightarrow \mathsf{beh}(s) = \mathsf{beh}(t) && \text{by Proposition 2} \\ &\Longleftrightarrow [\![s]\!] = [\![t]\!] && \text{by Theorem 3.(3)} \\ &\Longleftrightarrow [s] = [t] && \text{by Theorem 4, where } [-]: \mathsf{Re} \twoheadrightarrow \mathsf{Re}/\!\doteq \\ &\Longleftrightarrow s \doteq t. \end{aligned}$$

This gives a perfect bialgebraic match, where the equational logic on the algebra-side completely captures the observational equivalence on the coalgebra-side. Similar such results occur for instance within a line of work [10] in process algebra.

6 Conclusions

We have illustrated the effectiveness of the bialgebraic approach introduced by Turi and Plotkin [35] by showing how it neatly connects the elementary and classic structures of computer science, namely regular expressions, automata and languages. It thus forms a framework for what we consider to be the essence of computing: generated behaviour via matching algebra-coalgebra pairs. This framework may even guide developments in settings which are more complicated and possibly less well-developed, like extended regular expressions [22], or timed and probabilistic automata and their languages.

Acknowledgements

Thanks are due to Ichiro Hasuo for helpful discussions and for his valuable comments on the first draft of this paper.

References

1. M.A. Arbib and E.G. Manes. Foundations of system theory: Decomposable systems. *Automatica*, 10:285–302, 1974.
2. M.A. Arbib and E.G. Manes. *Algebraic Approaches to Program Semantics*. Texts and Monogr. in Comp. Sci.,. Springer, Berlin, 1986.
3. J.W. de Bakker and E. Vink. *Control Flow Semantics*. MIT Press, Cambridge, MA, 1996.
4. M. Barr and Ch. Wells. *Toposes, Triples and Theories*. Springer, Berlin, 1985. Revised and corrected version available from URL: `www.cwru.edu/artsci/math/wells/pub/ttt.html`.
5. F. Bartels. *On generalised coinduction and probabilistic specification formats. Distributive laws in coalgebraic modelling*. PhD thesis, Free Univ. Amsterdam, 2004.
6. J. Beck. Distributive laws. In B. Eckman, editor, *Seminar on Triples and Categorical Homolgy Theory*, number 80 in Lect. Notes Math., pages 119–140. Springer, Berlin, 1969.
7. B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. *Journ. ACM*, 42(1):232–268, 1988.
8. J.A. Brzozowski. Derivatives of regular expressions. *Journ. ACM*, 11(4):481–494, 1964.
9. J.H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.
10. W. Fokkink. On the completeness of the equations for the Kleene star in bisimulation. In M. Wirsing and M. Nivat, editors, *Algebraic Methodology and Software Technology*, number 1101 in Lect. Notes Comp. Sci., pages 180–194. Springer, Berlin, 1996.
11. J.A. Goguen. Minimal realization of machines in closed categories. *Bull. Amer. Math. Soc.*, 78(5):777–783, 1972.
12. J.A. Goguen. Realization is universal. *Math. Syst. Theor.*, 6(4):359–374, 1973.
13. J.A. Goguen. Discrete-time machines in closed monoidal categories. I. *Journ. Comp. Syst. Sci*, 10:1–43, 1975.
14. J.F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Inf. & Comp.*, 100(2):202–260, 1992.
15. C. Hermida and B. Jacobs. Structural induction and coinduction in a fibrational setting. *Inf. & Comp.*, 145:107–152, 1998.
16. B. Jacobs. Objects and classes, co-algebraically. In B. Freitag, C.B. Jones, C. Lengauer, and H.-J. Schek, editors, *Object-Orientation with Parallelism and Persistence*, pages 83–103. Kluwer Acad. Publ., 1996.

17. B. Jacobs. Exercises in coalgebraic specification. In R. Crole R. Backhouse and J. Gibbons, editors, *Algebraic and Coalgebraic Methods in the Mathematics of Program Construction*, number 2297 in Lect. Notes Comp. Sci., pages 237–280. Springer, Berlin, 2002.
18. B. Jacobs. Distributive laws for the coinductive solution of recursive equations. *Inf. & Comp.* 204(4), 2006, pages 561–587. Earlier version in number 106 in Elect. Notes in Theor. Comp. Sci.
19. P.T. Johnstone. Adjoint lifting theorems for categories of algebras. *Bull. London Math. Soc.*, 7:294–297, 1975.
20. M. Kick. Bialgebraic modelling of timed processes. In P. Widmayer *et al.*, editor, *International Colloquium on Automata, Languages and Programming*, number 2380 in Lect. Notes Comp. Sci., pages 525–536. Springer, Berlin, 2002.
21. S.C. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, number 34 in Annals of Mathematics Studies, pages 3–41. Princeton University Press, 1956.
22. S. Koushik and G. Rosu. Generating optimal monitors for extended regular expressions. In *Runtime Verification (RV'03)*, number 89(2) in Elect. Notes in Theor. Comp. Sci. Elsevier, Amsterdam, 2003.
23. D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Inf. & Comp.*, 110(2):366–390, 1994.
24. D. Kozen. Myhill-nerode relations on automatic systems and the completeness of Kleene algebra. In A. Ferreira and H. Reichel, editors, *Symposium on Theoretical Aspects of Computer Science*, number 2010 in Lect. Notes Comp. Sci., pages 27–38. Springer, Berlin, 2001.
25. S. Mac Lane. *Categories for the Working Mathematician*. Springer, Berlin, 1971.
26. M. Lenisa, J. Power, and H. Watanabe. Distributivity for endofunctors, pointed and copointed endofunctors, monads and comonads. In H. Reichel, editor, *Coalgebraic Methods in Computer Science*, number 33 in Elect. Notes in Theor. Comp. Sci. Elsevier, Amsterdam, 2000.
27. D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. *Theor. Comp. Sci.*, 54(2/3):267–276, 1987.
28. D. Perrin. Finite automata. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 1–55. Elsevier/MIT Press, 1990.
29. H. Reichel. An approach to object semantics based on terminal co-algebras. *Math. Struct. in Comp. Sci.*, 5:129–152, 1995.
30. J. Rutten. Automata and coinduction (an exercise in coalgebra). In D. Sangiorigi and R. de Simone, editors, *Concur'98: Concurrency Theory*, number 1466 in Lect. Notes Comp. Sci., pages 194–218. Springer, Berlin, 1998.
31. J. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theor. Comp. Sci.*, 308:1–53, 2003.
32. J. Rutten and D. Turi. Initial algebra and final coalgebra semantics for concurrency. In J.W. de Bakker, W.P. de Roever, and G. Rozenberg, editors, *A Decade of Concurrency*, number 803 in Lect. Notes Comp. Sci., pages 530–582. Springer, Berlin, 1994.
33. J.J.M.M. Rutten. Automata, power series, and coinduction: Taking input derivatives seriously (extended abstract). In J. Wiedermann, P. van Emde Boas, and M. Nielsen, editors, *International Colloquium on Automata, Languages and Programming*, number 1644 in Lect. Notes Comp. Sci., pages 645–654. Springer, Berlin, 1999.
34. D. Turi. *Functorial operational semantics and its denotational dual*. PhD thesis, Free Univ. Amsterdam, 1996.
35. D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *Logic in Computer Science*, pages 280–291. IEEE, Computer Science Press, 1997.
36. T. Uustalu, V. Vene, and A. Pardo. Recursion schemes from comonads. *Nordic Journ. Comput.*, 8(3):366–390, 2001.