Non-Deterministic Kleene Coalgebras

Silva, A.M.; Bonsangue, M.M.; Rutten, J.J.M.M.

Logical Methods in Computer Science
Vol. 6 (3:23) 2010, pp. 1–1–39
www.lmcs-online.org

Submitted Jan. 3, 2010
Published Sep. 9, 2010

NON-DETERMINISTIC KLEENE COALGEBRAS

ALEXANDRA SILVA [a], MARCELLO BONSANGUE [b], AND JAN RUTTEN [c]

[a] CWI, Amsterdam, The Netherlands
e-mail address: ams@cwi.nl

[b] LIACS, University of Leiden,The Netherlands
e-mail address: marcello@liacs.nl

[c] CWI (Amsterdam), VUA (Amsterdam) and RUN (Nijmegen) , The Netherlands
e-mail address: janr@cwi.nl

Abstract. In this paper, we present a systematic way of deriving (1) languages of (generalised) regular expressions, and (2) sound and complete axiomatizations thereof, for a wide variety of systems. This generalizes both the results of Kleene (on regular languages and deterministic finite automata) and Milner (on regular behaviours and finite labelled transition systems), and includes many other systems such as Mealy and Moore machines.

1. Introduction

In a previous paper [9], we presented a language to describe the behaviour of Mealy machines and a sound and complete axiomatization thereof. The defined language and axiomatization can be seen as the analogue of classical regular expressions [21] and Kleene algebra [22], for deterministic finite automata (DFA), or the process algebra and axiomatization for labelled transition systems (LTS) [28].

We now extend the previous approach and devise a framework wherein languages and axiomatizations can be uniformly derived for a large class of systems, including DFA, LTS and Mealy machines, which we will model as coalgebras.

Coalgebras provide a general framework for the study of dynamical systems such as DFA, Mealy machines and LTS. For a functor $\mathcal{G}\colon \mathbf{Set} \to \mathbf{Set}$, a $\mathcal{G}$-coalgebra or $\mathcal{G}$-system is a pair (S, g), consisting of a set S of states and a function $g\colon S \to \mathcal{G}(S)$ defining the "transitions" of the states. We call the functor $\mathcal{G}$ the *type* of the system. For instance, DFA can be modelled as coalgebras of the functor $\mathcal{G}(S) = 2 \times S^A$, Mealy machines are obtained by taking $\mathcal{G}(S) = (B \times S)^A$ and image-finite LTS are coalgebras for the functor $\mathcal{G}(S) = (\mathcal{P}_\omega(S))^A$, where $\mathcal{P}_\omega$ is finite powerset.

2000 ACM Subject Classification: F3.1, F3.2, F4.1.
Key words and phrases: Coalgebra, Kleene's theorem, axiomatization.
[a] The first author was partially supported by the Fundação para a Ciência e a Tecnologia, Portugal, under grant number SFRH/BD/27482/2006.

DOI:10.2168/LMCS-6 (3:23) 2010

Under mild conditions, functors $\mathcal{G}$ have a *final coalgebra* (unique up to isomorphism) into which every $\mathcal{G}$-coalgebra can be mapped via a unique so-called $\mathcal{G}$-*homomorphism*. The final coalgebra can be viewed as the universe of all possible $\mathcal{G}$-*behaviours*: the unique homomorphism into the final coalgebra maps every state of a coalgebra to a canonical representative of its behaviour. This provides a general notion of behavioural equivalence: two states are equivalent if and only if they are mapped to the same element of the final coalgebra. Instantiating the notion of final coalgebra for the aforementioned examples, the result is as expected: for DFA the final coalgebra is the set 2^{A^*} of all languages over A; for Mealy machines it is the set of causal functions $f\colon A^\omega \to B^\omega$; and for LTS it is the set of finitely branching trees with arcs labelled by $a \in A$ modulo bisimilarity. The notion of equivalence also specializes to the familiar notions: for DFA, two states are equivalent when they accept the same language; for Mealy machines, if they realize (or compute) the same causal function; and for LTS if they are bisimilar.

It is the main aim of this paper to show how the type of a system, given by the functor $\mathcal{G}$, is not only enough to determine a notion of behaviour and behavioural equivalence, but also allows for a uniform derivation of both a set of expressions describing behaviour and a corresponding axiomatization. The theory of universal coalgebra [31] provides a standard equivalence and a universal domain of behaviours, uniquely based on the functor $\mathcal{G}$. The main contributions of this paper are (1) the definition of a set of expressions $\mathsf{Exp}_\mathcal{G}$ describing $\mathcal{G}$-behaviours, (2) the proof of the correspondence between behaviours described by $\mathsf{Exp}_\mathcal{G}$ and locally finite $\mathcal{G}$-coalgebras (this is the analogue of Kleene's theorem), and (3) a corresponding sound and complete axiomatization, with respect to bisimulation, of $\mathsf{Exp}_\mathcal{G}$ (this is the analogue of Kleene algebra). All these results are solely based on the type of the system, given by the functor $\mathcal{G}$.

In a nutshell, we combine the work of Kleene with coalgebra, considering the class of non-deterministic functors. Hence, the title of the paper: non-deterministic Kleene coalgebras.

Organization of the paper. In Section 2 we introduce the class of non-deterministic functors and coalgebras. In Section 3 we associate with each non-deterministic functor $\mathcal{G}$ a generalized language $\mathsf{Exp}_\mathcal{G}$ of regular expressions and we present an analogue of Kleene's theorem, which makes precise the connection between $\mathsf{Exp}_\mathcal{G}$ and $\mathcal{G}$-coalgebras. A sound and complete axiomatization of $\mathsf{Exp}_\mathcal{G}$ is presented in Section 4. Section 5 contains two more examples of application of the framework and Section 6 shows a language and axiomatization for the class of polynomial and finitary coalgebras. Section 7 presents concluding remarks, directions for future work and discusses related work. This paper is an extended version of [11, 10]: it includes all the proofs, more examples and explanations, new material about polynomial and finitary functors and an extended discussion section.

2. Preliminaries

We give the basic definitions on non-deterministic functors and coalgebras and introduce the notion of bisimulation.

First we fix notation on sets and operations on them. Let **Set** be the category of sets and functions. Sets are denoted by capital letters $X, Y, \ldots$ and functions by lower case $f, g, \ldots$. We write $\emptyset$ for the empty set and the collection of all *finite* subsets of a set X is defined as $\mathcal{P}_\omega(X) = \{Y \subseteq X \mid Y \text{ finite}\}$. The collection of functions from a set X to a set

Y is denoted by Y^X. We write id_X for the identity function on set X. Given functions $f : X \to Y$ and $g : Y \to Z$ we write their composition as $g \circ f$. The product of two sets X, Y is written as $X \times Y$, with projection functions $X \xleftarrow{\pi_1} X \times Y \xrightarrow{\pi_2} Y$ .The set 1 is a singleton set typically written as $1 = \{*\}$ and it can be regarded as the empty product. We define $X \oplus Y$ as the set $X \uplus Y \uplus \{\bot, \top\}$, where $\uplus$ is the disjoint union of sets, with injections $X \xrightarrow{\kappa_1} X \uplus Y \xleftarrow{\kappa_2} Y$. Note that the set $X \oplus Y$ is different from the classical coproduct of X and Y (which we shall denote by $X + Y$), because of the two extra elements $\bot$ and $\top$. These extra elements will later be used to represent, respectively, underspecification and inconsistency in the specification of some systems. The intuition behind the need of these extra elements will become clear when we present our language of expressions and concrete examples, in Section 3.3.1, of systems whose type involves $\oplus$. Note that $X \oplus X \ncong 2 \times X \cong X + X$.

For each of the operations defined above on sets, there are analogous ones on functions. Let $f\colon X \to Y$, $f_1\colon X \to Y$ and $f_2\colon Z \to W$. We define the following operations:

$$f_1 \times f_2\colon X \times Z \to Y \times W \qquad f_1 \oplus f_2\colon X \oplus Z \to Y \oplus W$$
$$(f_1 \times f_2)(\langle x, z\rangle) = \langle f_1(x), f_2(z)\rangle \qquad (f_1 \oplus f_2)(c) = c,\ c \in \{\bot, \top\}$$
$$(f_1 \oplus f_2)(\kappa_i(x)) = \kappa_i(f_i(x)),\ i \in \{1, 2\}$$

$$f^A\colon X^A \to Y^A \qquad \mathcal{P}_\omega(f)\colon \mathcal{P}_\omega(X) \to \mathcal{P}_\omega(Y)$$
$$f^A(g) = f \circ g \qquad \mathcal{P}_\omega(f)(S) = \{f(x) \mid x \in S\}$$

Note that here we are using the same symbols that we defined above for the operations on sets. It will always be clear from the context which operation is being used.

In our definition of non-deterministic functors we will use constant sets equipped with an information order. In particular, we will use join-semilattices. A (bounded) join-semilattice is a set B equipped with a binary operation $\vee_{\mathsf{B}}$ and a constant $\bot_{\mathsf{B}} \in \mathsf{B}$, such that $\vee_{\mathsf{B}}$ is commutative, associative and idempotent. The element $\bot_{\mathsf{B}}$ is neutral with respect to $\vee_{\mathsf{B}}$. As usual, $\vee_{\mathsf{B}}$ gives rise to a partial ordering $\leq_{\mathsf{B}}$ on the elements of B:

$$b_1 \leq_{\mathsf{B}} b_2 \Leftrightarrow b_1 \vee_{\mathsf{B}} b_2 = b_2$$

Every set S can be mapped into a join-semilattice by taking B to be the set of all finite subsets of S with union as join.

Non-deterministic functors. Non-deterministic functors are functors $\mathcal{G}\colon \mathbf{Set} \to \mathbf{Set}$, built inductively from the identity and constants, using $\times$, $\oplus$, $(-)^A$ and $\mathcal{P}_\omega$.

Definition 2.1. The class *NDF* of *non-deterministic functors* on **Set** is inductively defined by putting:

$$NDF \ni \mathcal{G} ::= \mathsf{Id} \mid \mathsf{B} \mid \mathcal{G} \oplus \mathcal{G} \mid \mathcal{G} \times \mathcal{G} \mid \mathcal{G}^A \mid \mathcal{P}_\omega\mathcal{G}$$

where B is a finite (non-empty) join-semilattice and A is a finite set. ♣

Since we only consider finite exponents $A = \{a_1, \ldots, a_n\}$, the functor $(-)^A$ is not really needed, since it is subsumed by a product with n components. However, to simplify the presentation, we decided to include it.

Next, we show the explicit definition of the functors above on a set X and on a morphism $f\colon X \to Y$ (note that $\mathcal{G}(f)\colon \mathcal{G}(X) \to \mathcal{G}(Y)$).

$$\begin{array}{lll} \mathsf{Id}(X) = X & \mathsf{B}(X) = \mathsf{B} & (\mathcal{G}_1 \oplus \mathcal{G}_2)(X) = \mathcal{G}_1(X) \oplus \mathcal{G}_2(X) \\ \mathsf{Id}(f) = f & \mathsf{B}(f) = id_{\mathsf{B}} & (\mathcal{G}_1 \oplus \mathcal{G}_2)(f) = \mathcal{G}_1(f) \oplus \mathcal{G}_2(f) \\ (\mathcal{G}^A)(X) = \mathcal{G}(X)^A & (\mathcal{P}_\omega\mathcal{G})(X) = \mathcal{P}_\omega(\mathcal{G}(X)) & (\mathcal{G}_1 \times \mathcal{G}_2)(X) = \mathcal{G}_1(X) \times \mathcal{G}_2(X) \\ (\mathcal{G}^A)(f) = \mathcal{G}(f)^A & (\mathcal{P}_\omega\mathcal{G})(f) = \mathcal{P}_\omega(\mathcal{G}(f)) & (\mathcal{G}_1 \times \mathcal{G}_2)(f) = \mathcal{G}_1(f) \times \mathcal{G}_2(f) \end{array}$$

Typical examples of non-deterministic functors include $\mathcal{M} = (\mathsf{B} \times \mathsf{Id})^A$, $\mathcal{D} = 2 \times \mathsf{Id}^A$, $\mathcal{Q} = (1 \oplus \mathsf{Id})^A$ and $\mathcal{N} = 2 \times (\mathcal{P}_\omega\mathsf{Id})^A$, where $2 = \{0, 1\}$ is a two-element join semilattice with 0 as bottom element ($1 \vee 0 = 1$) and $1 = \{*\}$ is a one element join-semilattice. These functors represent, respectively, the type of Mealy, deterministic, partial deterministic and non-deterministic automata. In this paper, we will use the last three as running examples. In [9], we have studied in detail regular expressions for Mealy automata. Similarly to what happened there, we impose a join-semilattice structure on the constant functor. The product, exponentiation and powerset functors preserve the join-semilattice structure and thus do not need to be changed. This is not the case for the classical coproduct and thus we use $\oplus$ instead, which also guarantees that the join semilattice structure is preserved.

Next, we give the definition of the ingredient relation, which relates a non-deterministic functor $\mathcal{G}$ with its *ingredients*, *i.e.* the functors used in its inductive construction. We shall use this relation later for typing our expressions.

Definition 2.2. Let $\lhd \subseteq NDF \times NDF$ be the least reflexive and transitive relation on non-deterministic functors such that

$$\mathcal{G}_1 \lhd \mathcal{G}_1 \times \mathcal{G}_2, \quad \mathcal{G}_2 \lhd \mathcal{G}_1 \times \mathcal{G}_2, \quad \mathcal{G}_1 \lhd \mathcal{G}_1 \oplus \mathcal{G}_2, \quad \mathcal{G}_2 \lhd \mathcal{G}_1 \oplus \mathcal{G}_2, \quad \mathcal{G} \lhd \mathcal{G}^A, \quad \mathcal{G} \lhd \mathcal{P}_\omega\mathcal{G}$$

♣

Here and throughout this document we use $\mathcal{F} \lhd \mathcal{G}$ as a shorthand for $\langle \mathcal{F}, \mathcal{G} \rangle \in \lhd$. If $\mathcal{F} \lhd \mathcal{G}$, then $\mathcal{F}$ is said to be an *ingredient* of $\mathcal{G}$. For example, 2, Id, Id^A and $\mathcal{D}$ itself are all the ingredients of the deterministic automata functor $\mathcal{D} = 2 \times \mathsf{Id}^A$.

Non-deterministic coalgebras. A non-deterministic coalgebra is a pair $(S, f\colon S \to \mathcal{G}(S))$, where S is a set of states and $\mathcal{G}$ is a non-deterministic functor. The functor $\mathcal{G}$, together with the function f, determines the *transition structure* (or dynamics) of the $\mathcal{G}$-coalgebra [31]. Mealy, deterministic, partial deterministic and non-deterministic automata are, respectively, coalgebras for the functors $\mathcal{M} = (\mathsf{B} \times \mathsf{Id})^A$, $\mathcal{D} = 2 \times \mathsf{Id}^A$, $\mathcal{Q} = (1 \oplus \mathsf{Id})^A$ and $\mathcal{N} = 2 \times (\mathcal{P}_\omega\mathsf{Id})^A$.

A *$\mathcal{G}$-homomorphism* from a $\mathcal{G}$-coalgebra (S, f) to a $\mathcal{G}$-coalgebra (T, g) is a function $h\colon S \to T$ preserving the transition structure, *i.e.* such that $g \circ h = \mathcal{G}(h) \circ f$.

Definition 2.3. A $\mathcal{G}$-coalgebra (Ω, ω) is said to be *final* if for any $\mathcal{G}$-coalgebra (S, f) there exists a unique $\mathcal{G}$-homomorphism $\mathbf{beh}_S\colon S \to \Omega$. ♣

For every non-deterministic functor $\mathcal{G}$ there exists a final $\mathcal{G}$-coalgebra $(\Omega_\mathcal{G}, \omega_\mathcal{G})$ [31]. For instance, as we already mentioned in the introduction, the final coalgebra for the functor $\mathcal{D}$ is the set of languages 2^{A^*} over A, together with a transition function $d : 2^{A^*} \to 2 \times (2^{A^*})^A$ defined as $d(\phi) = \langle \phi(\epsilon), \lambda a \lambda w.\phi(aw) \rangle$. Here ϵ denotes the empty sequence and aw denotes the word resulting from prefixing w with the letter a. The notion of finality will play a key role later in providing a semantics to expressions.

Given a $\mathcal{G}$-coalgebra (S, f) and a subset V of S with inclusion map $i\colon V \to S$ we say that V is a subcoalgebra of S if there exists $g\colon V \to \mathcal{G}(V)$ such that i is a homomorphism.

Given $s \in S$, $\langle s \rangle = (T, t)$, denotes the smallest subcoalgebra generated by s, with T given by

$$T = \bigcap \{V \mid V \text{ is a subcoalgebra of } S \text{ and } s \in V\}$$

If the functor $\mathcal{F}$ preserves arbitrary intersections, then the subcoalgebra $\langle s \rangle$ exists. This will be the case for every functor considered in this paper. Moreover, all the functors we will consider preserve monos and thus the transition structure t is unique [31, Proposition 6.1].

We will write $\mathit{Coalg}(\mathcal{G})$ for the category of $\mathcal{G}$-coalgebras together with coalgebra homomorphisms. We also write $\mathit{Coalg}_{\mathbf{LF}}(\mathcal{G})$ for the category of $\mathcal{G}$-coalgebras that are *locally finite*. Objects are $\mathcal{G}$-coalgebras (S, f) such that for each state $s \in S$ the generated subcoalgebra $\langle s \rangle$ is finite. Maps are the usual homomorphisms of coalgebras.

Let (S, f) and (T, g) be two $\mathcal{G}$-coalgebras. We call a relation $R \subseteq S \times T$ a *bisimulation* [18] iff

$$\langle s, t \rangle \in R \Rightarrow \langle f(s), g(t) \rangle \in \overline{\mathcal{G}}(R)$$

where $\overline{\mathcal{G}}(R)$ is defined as $\overline{\mathcal{G}}(R) = \{\langle \mathcal{G}(\pi_1)(x), \mathcal{G}(\pi_2)(x) \rangle \mid x \in \mathcal{G}(R)\}$. We write $s \sim_{\mathcal{G}} t$ whenever there exists a bisimulation relation containing (s, t) and we call $\sim_{\mathcal{G}}$ the bisimilarity relation. We shall drop the subscript $\mathcal{G}$ whenever the functor $\mathcal{G}$ is clear from the context. For all non-deterministic $\mathcal{G}$-coalgebras (S, f) and (T, g) and $s \in S, t \in T$, it holds that $s \sim t \iff \mathbf{beh}_S(s) = \mathbf{beh}_T(t)$ (the left to right implication always holds, whereas the right to left implication only holds for certain classes of functors, which include the ones we consider in this paper [31, 35]).

3. A language of expressions for non-deterministic coalgebras

In this section, we generalize the classical notion of regular expressions to non-deterministic coalgebras. We start by introducing an untyped language of expressions and then we single out the well-typed ones via an appropriate typing system, thereby associating expressions to non-deterministic functors.

Definition 3.1 (Expressions)**.** Let A be a finite set, B a finite join-semilattice and X a set of fixed point variables. The set Exp of all *expressions* is given by the following grammar, where $a \in A$, $b \in \mathsf{B}$ and $x \in X$:

$$\varepsilon \quad ::= \quad \underline{\emptyset} \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid b \mid l\langle \varepsilon \rangle \mid r\langle \varepsilon \rangle \mid l[\varepsilon] \mid r[\varepsilon] \mid a(\varepsilon) \mid \{\varepsilon\}$$

where γ is a *guarded expression* given by:

$$\gamma \quad ::= \quad \underline{\emptyset} \mid \gamma \oplus \gamma \mid \mu x.\gamma \mid b \mid l\langle \varepsilon \rangle \mid r\langle \varepsilon \rangle \mid l[\varepsilon] \mid r[\varepsilon] \mid a(\varepsilon) \mid \{\varepsilon\}$$

The only difference between the BNF of γ and ε is the occurrence of x. ♣

In the expression $\mu x.\gamma$, μ is a binder for all the free occurrences of x in γ. Variables that are not bound are free. A *closed expression* is an expression without free occurrences of fixed point variables x. We denote the set of closed expressions by Exp^c.

Intuitively, expressions denote elements of the final coalgebra. The expressions $\underline{\emptyset}$, $\varepsilon_1 \oplus \varepsilon_2$ and $\mu x.\, \varepsilon$ will play a similar role to, respectively, the empty language, the union of languages and the Kleene star in classical regular expressions for deterministic automata. The expressions $l\langle \varepsilon \rangle$ and $r\langle \varepsilon \rangle$ refer to the left and right hand-side of products. Similarly, $l[\varepsilon]$ and $r[\varepsilon]$ refer to the left and right hand-side of sums. The expressions $a(\varepsilon)$ and $\{\varepsilon\}$ denote function application and a singleton set, respectively. We shall soon illustrate, by means

of examples, the role of these expressions. Here, it is already visible that our approach (to define a language) for the powerset functor differs from classical modal logic where $\Box$ and $\Diamond$ are used. This is a choice, justified by the fact that our goal is to have a "process algebra" like language instead of a modal logic one. It also explains why we only consider finite powerset: every finite set can be written as the finite union of its singletons.

Our language does not have any operator denoting intersection or complement (it only includes the sum operator $\oplus$). This is a natural restriction, very much in the spirit of Kleene's regular expressions for deterministic finite automata. We will prove that this simple language is expressive enough to denote exactly all locally finite coalgebras.

Next, we present a typing assignment system for associating expressions to non-deterministic functors. This will allow us to associate with each functor $\mathcal{G}$ the expressions $\varepsilon \in \mathsf{Exp}^c$ that are valid specifications of $\mathcal{G}$-coalgebras. The typing proceeds following the structure of the expressions and the ingredients of the functors.

Definition 3.2 (Type system)**.** We define a typing relation $\vdash \subseteq \mathsf{Exp} \times \mathit{NDF} \times \mathit{NDF}$ that will associate an expression ε with two non-deterministic functors $\mathcal{F}$ and $\mathcal{G}$, which are related by the ingredient relation ($\mathcal{F}$ is an ingredient of $\mathcal{G}$). We shall write $\vdash \varepsilon \colon \mathcal{F} \lhd \mathcal{G}$ for $\langle \varepsilon, \mathcal{F}, \mathcal{G} \rangle \in \vdash$. The rules that define $\vdash$ are the following:

$$\frac{}{\vdash \underline{\emptyset} \colon \mathcal{F} \lhd \mathcal{G}} \qquad \frac{}{\vdash b \colon \mathsf{B} \lhd \mathcal{G}} \qquad \frac{}{\vdash x \colon \mathcal{G} \lhd \mathcal{G}} \qquad \frac{\vdash \varepsilon \colon \mathcal{G} \lhd \mathcal{G}}{\vdash \mu x.\varepsilon \colon \mathcal{G} \lhd \mathcal{G}}$$

$$\frac{\vdash \varepsilon_1 \colon \mathcal{F} \lhd \mathcal{G} \quad \vdash \varepsilon_2 \colon \mathcal{F} \lhd \mathcal{G}}{\vdash \varepsilon_1 \oplus \varepsilon_2 \colon \mathcal{F} \lhd \mathcal{G}} \qquad \frac{\vdash \varepsilon \colon \mathcal{G} \lhd \mathcal{G}}{\vdash \varepsilon \colon \mathsf{Id} \lhd \mathcal{G}} \qquad \frac{\vdash \varepsilon \colon \mathcal{F} \lhd \mathcal{G}}{\vdash \{\varepsilon\} \colon \mathcal{P}_\omega \mathcal{F} \lhd \mathcal{G}} \qquad \frac{\vdash \varepsilon \colon \mathcal{F} \lhd \mathcal{G}}{\vdash a(\varepsilon) \colon \mathcal{F}^A \lhd \mathcal{G}}$$

$$\frac{\vdash \varepsilon \colon \mathcal{F}_1 \lhd \mathcal{G}}{\vdash l\langle \varepsilon \rangle \colon \mathcal{F}_1 \times \mathcal{F}_2 \lhd \mathcal{G}} \qquad \frac{\vdash \varepsilon \colon \mathcal{F}_2 \lhd \mathcal{G}}{\vdash r\langle \varepsilon \rangle \colon \mathcal{F}_1 \times \mathcal{F}_2 \lhd \mathcal{G}} \qquad \frac{\vdash \varepsilon \colon \mathcal{F}_1 \lhd \mathcal{G}}{\vdash l[\varepsilon] \colon \mathcal{F}_1 \oplus \mathcal{F}_2 \lhd \mathcal{G}} \qquad \frac{\vdash \varepsilon \colon \mathcal{F}_2 \lhd \mathcal{G}}{\vdash r[\varepsilon] \colon \mathcal{F}_1 \oplus \mathcal{F}_2 \lhd \mathcal{G}}$$

♣

Intuitively, $\vdash \varepsilon \colon \mathcal{F} \lhd \mathcal{G}$ (for a closed expression ε) means that ε denotes an element of $\mathcal{F}(\Omega_{\mathcal{G}})$, where $\Omega_{\mathcal{G}}$ is the final coalgebra of $\mathcal{G}$. As expected, there is a rule for each expression construct. The extra rule involving $\mathsf{Id} \lhd \mathcal{G}$ reflects the isomorphism between the final coalgebra $\Omega_{\mathcal{G}}$ and $\mathcal{G}(\Omega_{\mathcal{G}})$ (Lambek's lemma, cf. [31]). Only fixed points at the outermost level of the functor are allowed. This does not mean however that we disallow nested fixed points. For instance, $\mu x.\, a(x \oplus \mu y.\, a(y))$ would be a well-typed expression for the functor $\mathcal{D}$ of deterministic automata, as it will become clear below, when we will present more examples of well-typed and non-well-typed expressions. The presented type system is decidable (expressions are of finite length and the system is inductive on the structure of $\varepsilon \in \mathsf{Exp}$). Note that the rules above are meant to be read as an inductive definition rather than as an algorithm. In an eventual implementation, extra care is needed in the case $\mathcal{G} = \mathsf{Id}$, to avoid looping in the rule for $\mathsf{Id} \lhd \mathcal{G}$.

We can formally define the set of $\mathcal{G}$-expressions: (closed and guarded) well-typed expressions associated with a non-deterministic functor $\mathcal{G}$.

Definition 3.3 ($\mathcal{G}$-expressions)**.** Let $\mathcal{G}$ be a non-deterministic functor and $\mathcal{F}$ an ingredient of $\mathcal{G}$. We define $\mathsf{Exp}_{\mathcal{F} \lhd \mathcal{G}}$ by:

$$\mathsf{Exp}_{\mathcal{F} \lhd \mathcal{G}} = \{\varepsilon \in \mathsf{Exp}^c \mid \vdash \varepsilon \colon \mathcal{F} \lhd \mathcal{G}\}\,.$$

We define the set $\mathsf{Exp}_{\mathcal{G}}$ of well-typed $\mathcal{G}$-*expressions* by $\mathsf{Exp}_{\mathcal{G} \lhd \mathcal{G}}$. ♣

Let us instantiate the definition of $\mathcal{G}$-expressions to the functors of deterministic automata $\mathcal{D} = 2 \times \mathsf{Id}^A$.

Example 3.4 (Deterministic expressions)**.** Let A be a finite set of input actions and let X be a set of (recursion or) fixed point variables. The set $\mathsf{Exp}_{\mathcal{D}}$ of *deterministic expressions* is given by the set of closed and guarded (each variable occurs in the scope of $a(-)$) expressions generated by the following BNF grammar. For $a \in A$ and $x \in X$:

$$\begin{aligned} \mathsf{Exp}_{\mathcal{D}} \ni \varepsilon &::= \underline{\emptyset} \mid \varepsilon \oplus \varepsilon \mid \mu x.\varepsilon \mid x \mid l\langle \varepsilon_1 \rangle \mid r\langle \varepsilon_2 \rangle \\ \varepsilon_1 &::= \underline{\emptyset} \mid 0 \mid 1 \mid \varepsilon_1 \oplus \varepsilon_1 \\ \varepsilon_2 &::= \underline{\emptyset} \mid a(\varepsilon) \mid \varepsilon_2 \oplus \varepsilon_2 \end{aligned}$$

♠

Examples of well-typed expressions for the functor $\mathcal{D} = 2 \times \mathsf{Id}^A$ (with $2 = \{0, 1\}$ a two-element join-semilattice with 0 as bottom element; recall that the ingredients of $\mathcal{D}$ are 2, Id^A and $\mathcal{D}$ itself) include $r\langle a(\underline{\emptyset})\rangle$, $l\langle 1\rangle \oplus r\langle a(l\langle 0\rangle)\rangle$ and $\mu x.r\langle a(x)\rangle \oplus l\langle 1\rangle$. The expressions $l[1]$, $l\langle 1\rangle \oplus 1$ and $\mu x.1$ are examples of non well-typed expressions for $\mathcal{D}$, because the functor $\mathcal{D}$ does not involve $\oplus$, the subexpressions in the sum have different type, and recursion is not at the outermost level (1 has type $2 \lhd \mathcal{D}$), respectively.

It is easy to see that the closed (and guarded) expressions generated by the grammar presented above are exactly the elements of $\mathsf{Exp}_{\mathcal{D}}$. The most interesting case to check is the expression $r\langle a(\varepsilon)\rangle$. Note that $a(\varepsilon)$ has type $\mathsf{Id}^A \lhd \mathcal{D}$ as long as ε has type $\mathsf{Id} \lhd \mathcal{D}$. And the crucial remark here is that, by definition of $\vdash$, $\mathsf{Exp}_{\mathsf{Id} \lhd \mathcal{G}} \subseteq \mathsf{Exp}_{\mathcal{G}}$. Therefore, ε has type $\mathsf{Id} \lhd \mathcal{D}$ if it is of type $\mathcal{D} \lhd \mathcal{D}$, or more precisely, if $\varepsilon \in \mathsf{Exp}_{\mathcal{D}}$, which explains why the grammar above is correct.

At this point, we should remark that the syntax of our expressions differs from the classical regular expressions in the use of μ and action prefixing $a(\varepsilon)$ instead of star and full concatenation. We shall prove later that these two syntactically different formalisms are equally expressive (Theorems 3.12 and 3.14), but, to increase the intuition behind our expressions, let us present the syntactic translation from classical regular expressions to $\mathsf{Exp}_{\mathcal{D}}$ (this translation is inspired by [28]) and back.

Definition 3.5. The set of regular expressions is given by the following syntax

$$RE \ni r ::= \underline{0} \mid \underline{1} \mid a \mid r + r \mid r \cdot r \mid r^*$$

where $a \in A$ and $\cdot$ denotes sequential composition. We define the following translations between regular expressions and deterministic expressions:

$$\begin{array}{ll} (-)^\dagger \colon RE \to \mathsf{Exp}_{\mathcal{D}} & \\ (\underline{0})^\dagger & = \underline{\emptyset} \\ (\underline{1})^\dagger & = l\langle 1\rangle \\ (a)^\dagger & = r\langle a(l\langle 1\rangle)\rangle \\ (r_1 + r_2)^\dagger & = (r_1)^\dagger \oplus (r_2)^\dagger \\ (r_1 \cdot r_2)^\dagger & = (r_1)^\dagger[(r_2)^\dagger / l\langle 1\rangle] \\ (r^*)^\dagger & = \mu x.(r)^\dagger[x / l\langle 1\rangle] \oplus l\langle 1\rangle \end{array} \qquad \begin{array}{ll} (-)^\ddagger \colon \mathsf{Exp}_{\mathcal{D}} \to RE & \\ (\underline{\emptyset})^\ddagger & = \underline{0} \\ (l\langle \underline{\emptyset}\rangle)^\ddagger & = (l\langle 0\rangle)^\ddagger = (r\langle \underline{\emptyset}\rangle)^\ddagger = \underline{0} \\ (l\langle 1\rangle)^\ddagger & = \underline{1} \\ (l\langle \varepsilon_1 \oplus \varepsilon_2\rangle)^\ddagger & = (l\langle \varepsilon_1\rangle)^\ddagger + (l\langle \varepsilon_2\rangle)^\ddagger \\ (r\langle a(\varepsilon)\rangle)^\ddagger & = a \cdot (\varepsilon)^\ddagger \\ (r\langle \varepsilon_1 \oplus \varepsilon_2\rangle)^\ddagger & = (r\langle \varepsilon_1\rangle)^\ddagger + (r\langle \varepsilon_2\rangle)^\ddagger \\ (\varepsilon_1 \oplus \varepsilon_2)^\ddagger & = (\varepsilon_1)^\ddagger + (\varepsilon_2)^\ddagger \\ (\mu x.\varepsilon)^\ddagger & = \mathbf{sol}(\mathbf{eqs}(\mu x.\varepsilon)) \end{array}$$

The function **eqs** translates $\mu x.\varepsilon$ into a system of equations in the following way. Let $\mu x_1.\varepsilon_1, \ldots, \mu x_n.\varepsilon_n$ be all the fixed point subexpressions of $\mu x.\varepsilon$, with $x_1 = x$ and $\varepsilon_1 = \varepsilon$. We

define n equations $x_i = (\overline{\varepsilon}_i)^\dagger$, where $\overline{\varepsilon}_i$ is obtained from ε_i by replacing each subexpression $\mu x_i.\varepsilon_i$ by x_i, for all $i = 1, \ldots n$. The solution of the system, $\mathbf{sol}(\mathbf{eqs}(\mu x.\varepsilon))$, is then computed in the usual way (the solution of an equation of shape $x = rx + t$ is r^*t).

In [32], regular expressions were given a coalgebraic structure, using Brzozowski derivatives [13]. Later in this paper, we will provide a coalgebra structure to $\mathsf{Exp}_{\mathcal{D}}$, after which the soundness of the above translations can be stated and proved: $r \sim r^\dagger$ and $\varepsilon \sim \varepsilon^\ddagger$, where $\sim$ will coincide with language equivalence. ♣

Thus, the regular expression aa^* is translated to $r\langle a(\mu x.r\langle a(x)\rangle \oplus l\langle 1\rangle)\rangle$, whereas the expression $\mu x.r\langle a(r\langle a(x)\rangle)\rangle \oplus l\langle 1\rangle$ is transformed into $(aa)^*$.

We present next the syntax for the expressions in $\mathsf{Exp}_{\mathcal{Q}}$ and in $\mathsf{Exp}_{\mathcal{N}}$ (recall that $\mathcal{Q} = (1 \oplus \mathsf{Id})^A$ and $\mathcal{N} = 2 \times (\mathcal{P}_\omega \mathsf{Id})^A$).

Example 3.6 (Partial expressions)**.** Let A be a finite set of input actions and X be a set of (recursion or) fixed point variables. The set $\mathsf{Exp}_{\mathcal{Q}}$ of *partial expressions* is given by the set of closed and guarded expressions generated by the following BNF grammar. For $a \in A$ and $x \in X$:

$$\begin{aligned} \mathsf{Exp}_{\mathcal{Q}} \ni \varepsilon &::= \underline{\emptyset} \mid \varepsilon \oplus \varepsilon \mid \mu x.\varepsilon \mid x \mid a(\varepsilon_1) \\ \varepsilon_1 &::= \underline{\emptyset} \mid \varepsilon_1 \oplus \varepsilon_1 \mid l[\varepsilon_2] \mid r[\varepsilon] \\ \varepsilon_2 &::= \underline{\emptyset} \mid \varepsilon_2 \oplus \varepsilon_2 \mid * \end{aligned}$$

Intuitively, the expressions $a(l[*])$ and $a(r[\varepsilon])$ specify, respectively, a state which has no defined transition for input a and a state with an outgoing transition to another one specified by ε. ♠

Example 3.7 (Non-deterministic expressions)**.** Let A be a finite set of input actions and X be a set of (recursion or) fixed point variables. The set $\mathsf{Exp}_{\mathcal{N}}$ of *non-deterministic expressions* is given by the set of closed and guarded expressions generated by the following BNF grammar. For $a \in A$ and $x \in X$:

$$\begin{aligned} \mathsf{Exp}_{\mathcal{N}} \ni \varepsilon &::= \underline{\emptyset} \mid x \mid r\langle \varepsilon_2 \rangle \mid l\langle \varepsilon_1 \rangle \mid \varepsilon \oplus \varepsilon \mid \mu x.\varepsilon \\ \varepsilon_1 &::= \underline{\emptyset} \mid \varepsilon_1 \oplus \varepsilon_1 \mid 1 \mid 0 \\ \varepsilon_2 &::= \underline{\emptyset} \mid \varepsilon_2 \oplus \varepsilon_2 \mid a(\varepsilon') \\ \varepsilon' &::= \underline{\emptyset} \mid \varepsilon' \oplus \varepsilon' \mid \{\varepsilon\} \end{aligned}$$

Intuitively, the expression $r\langle a(\{\varepsilon_1\} \oplus \{\varepsilon_2\})\rangle$ specifies a state which has two outgoing transitions labelled with the input letter a, one to a state specified by ε_1 and another to a state specified by ε_2. ♠

We have defined a language of expressions which gives us an algebraic description of systems. We should also remark at this point that in the examples we strictly follow the type system to derive the syntax of the expressions. However, it is obvious that many simplifications can be made in order to obtain a more polished language. In particular, after the axiomatization we will be able to decrease the number of levels in the above grammars, since will we have axioms of the shape $a(\varepsilon) \oplus a(\varepsilon') \equiv a(\varepsilon \oplus \varepsilon')$. In Section 5, we will sketch two examples where we apply some simplification to the syntax.

The goal is now to present a generalization of Kleene's theorem for non-deterministic coalgebras (Theorems 3.12 and 3.14). Recall that, for regular languages, the theorem states that a language is regular if and only if it is recognized by a finite automaton. In order to achieve our goal we will first show that the set $\mathsf{Exp}_{\mathcal{G}}$ of $\mathcal{G}$-expressions carries a $\mathcal{G}$-coalgebra structure.

3.1. **Expressions are coalgebras.** In this section, we show that the set of $\mathcal{G}$-expressions for a given non-deterministic functor $\mathcal{G}$ has a coalgebraic structure $\delta_{\mathcal{G}} \colon \mathsf{Exp}_{\mathcal{G}} \to \mathcal{G}(\mathsf{Exp}_{\mathcal{G}})$. More precisely, we are going to define a function

$$\delta_{\mathcal{F}\lhd\mathcal{G}} : \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}} \to \mathcal{F}(\mathsf{Exp}_{\mathcal{G}})$$

for every ingredient $\mathcal{F}$ of $\mathcal{G}$, and then set $\delta_{\mathcal{G}} = \delta_{\mathcal{G}\lhd\mathcal{G}}$. Our definition of the function $\delta_{\mathcal{F}\lhd\mathcal{G}}$ will make use of the following.

Definition 3.8. For every $\mathcal{G} \in NDF$ and for every $\mathcal{F}$ with $\mathcal{F} \lhd \mathcal{G}$:

(i) we define a constant $\mathsf{Empty}_{\mathcal{F}\lhd\mathcal{G}} \in \mathcal{F}(\mathsf{Exp}_{\mathcal{G}})$ by induction on the syntactic structure of $\mathcal{F}$:

$$\begin{array}{lcllcl}
\mathsf{Empty}_{\mathsf{Id}\lhd\mathcal{G}} & = & \underline{\emptyset} & \mathsf{Empty}_{\mathcal{F}_1 \oplus \mathcal{F}_2\lhd\mathcal{G}} & = & \bot \\
\mathsf{Empty}_{\mathsf{B}\lhd\mathcal{G}} & = & \bot_{\mathsf{B}} & \mathsf{Empty}_{\mathcal{F}^A\lhd\mathcal{G}} & = & \lambda a.\mathsf{Empty}_{\mathcal{F}\lhd\mathcal{G}} \\
\mathsf{Empty}_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}} & = & \langle \mathsf{Empty}_{\mathcal{F}_1\lhd\mathcal{G}}, \mathsf{Empty}_{\mathcal{F}_2\lhd\mathcal{G}}\rangle & \mathsf{Empty}_{\mathcal{P}_{\mathcal{B}}\mathcal{F}\lhd\mathcal{G}} & = & \emptyset
\end{array}$$

(ii) we define a function $\mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}} \colon \mathcal{F}(\mathsf{Exp}_{\mathcal{G}}) \times \mathcal{F}(\mathsf{Exp}_{\mathcal{G}}) \to \mathcal{F}(\mathsf{Exp}_{\mathcal{G}})$ by induction on the syntactic structure of $\mathcal{F}$:

$$\begin{array}{lcl}
\mathsf{Plus}_{\mathsf{Id}\lhd\mathcal{G}}(\varepsilon_1, \varepsilon_2) & = & \varepsilon_1 \oplus \varepsilon_2 \\
\mathsf{Plus}_{\mathsf{B}\lhd\mathcal{G}}(b_1, b_2) & = & b_1 \vee_{\mathsf{B}} b_2 \\
\mathsf{Plus}_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}(\langle\varepsilon_1,\varepsilon_2\rangle, \langle\varepsilon_3,\varepsilon_4\rangle) & = & \langle \mathsf{Plus}_{\mathcal{F}_1\lhd\mathcal{G}}(\varepsilon_1,\varepsilon_3), \mathsf{Plus}_{\mathcal{F}_2\lhd\mathcal{G}}(\varepsilon_2,\varepsilon_4)\rangle \\
\mathsf{Plus}_{\mathcal{F}_1\oplus\mathcal{F}_2\lhd\mathcal{G}}(\kappa_i(\varepsilon_1), \kappa_i(\varepsilon_2)) & = & \kappa_i(\mathsf{Plus}_{\mathcal{F}_i\lhd\mathcal{G}}(\varepsilon_1,\varepsilon_2)), \quad i \in \{1,2\} \\
\mathsf{Plus}_{\mathcal{F}_1\oplus\mathcal{F}_2\lhd\mathcal{G}}(\kappa_i(\varepsilon_1), \kappa_j(\varepsilon_2)) & = & \top \quad i,j \in \{1,2\} \;\; and \;\; i \neq j \\
\mathsf{Plus}_{\mathcal{F}_1\oplus\mathcal{F}_2\lhd\mathcal{G}}(x, \top) & = & \mathsf{Plus}_{\mathcal{F}_1\oplus\mathcal{F}_2\lhd\mathcal{G}}(\top, x) = \top \\
\mathsf{Plus}_{\mathcal{F}_1\oplus\mathcal{F}_2\lhd\mathcal{G}}(x, \bot) & = & \mathsf{Plus}_{\mathcal{F}_1\oplus\mathcal{F}_2\lhd\mathcal{G}}(\bot, x) = x \\
\mathsf{Plus}_{\mathcal{F}^A\lhd\mathcal{G}}(f, g) & = & \lambda a.\ \mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}}(f(a), g(a)) \\
\mathsf{Plus}_{\mathcal{P}_{\mathcal{B}}\mathcal{F}\lhd\mathcal{G}}(s_1, s_2) & = & s_1 \cup s_2
\end{array}$$

Intuitively, one can think of the constant $\mathsf{Empty}_{\mathcal{F}\lhd\mathcal{G}}$ and the function $\mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}}$ as liftings of $\underline{\emptyset}$ and $\oplus$ to the level of $\mathcal{F}(\mathsf{Exp}_{\mathcal{G}})$. ♣

We need two more things to define $\delta_{\mathcal{F}\lhd\mathcal{G}}$. First, we define an order $\preceq$ on the types of expressions. For $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{G}$ non-deterministic functors such that $\mathcal{F}_1 \lhd \mathcal{G}$ and $\mathcal{F}_2 \lhd \mathcal{G}$, we define

$$(\mathcal{F}_1 \lhd \mathcal{G}) \preceq (\mathcal{F}_2 \lhd \mathcal{G}) \Leftrightarrow \mathcal{F}_1 \lhd \mathcal{F}_2$$

The order $\preceq$ is a partial order (structure inherited from $\lhd$). Note also that $(\mathcal{F}_1 \lhd \mathcal{G}) = (\mathcal{F}_2 \lhd \mathcal{G}) \Leftrightarrow \mathcal{F}_1 = \mathcal{F}_2$. Second, we define a measure $N(\varepsilon)$ based on the maximum number of nested unguarded occurrences of μ-expressions in ε and unguarded occurrences of $\oplus$. We say that a subexpression $\mu x.\varepsilon_1$ of ε occurs unguarded if it is not in the scope of one of the operators $l\langle-\rangle$, $r\langle-\rangle$, $l[-]$, $r[-]$, $a(-)$ or $\{-\}$.

Definition 3.9. For every guarded expression ε, we define $N(\varepsilon)$ as follows:

$$N(\underline{\emptyset}) = N(b) = N(a(\varepsilon)) = N(l\langle\varepsilon\rangle) = N(r\langle\varepsilon\rangle) = N(l[\varepsilon]) = N(r[\varepsilon]) = N(\{\varepsilon\}) = 0$$
$$N(\varepsilon_1 \oplus \varepsilon_2) = 1 + max\{N(\varepsilon_1),\, N(\varepsilon_2)\}$$
$$N(\mu x.\varepsilon) = 1 + N(\varepsilon)$$

♣

The measure N induces a partial order on the set of expressions: $\varepsilon_1 \ll \varepsilon_2 \Leftrightarrow N(\varepsilon_1) \leq N(\varepsilon_2)$, where $\leq$ is just the ordinary inequality of natural numbers.

Now we have all we need to define $\delta_{\mathcal{F}\lhd\mathcal{G}} \colon \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}} \to \mathcal{F}(\mathsf{Exp}_{\mathcal{G}})$.

Definition 3.10. For every ingredient $\mathcal{F}$ of a non-deterministic functor $\mathcal{G}$ and an expression $\varepsilon \in \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}$, we define $\delta_{\mathcal{F}\lhd\mathcal{G}}(\varepsilon)$ as follows:

$$\begin{array}{lcl}
\delta_{\mathcal{F}\lhd\mathcal{G}}(\underline{\emptyset}) & = & \mathsf{Empty}_{\mathcal{F}\lhd\mathcal{G}} \\
\delta_{\mathcal{F}\lhd\mathcal{G}}(\varepsilon_1 \oplus \varepsilon_2) & = & \mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}}(\delta_{\mathcal{F}\lhd\mathcal{G}}(\varepsilon_1), \delta_{\mathcal{F}\lhd\mathcal{G}}(\varepsilon_2)) \\
\delta_{\mathcal{G}\lhd\mathcal{G}}(\mu x.\varepsilon) & = & \delta_{\mathcal{G}\lhd\mathcal{G}}(\varepsilon[\mu x.\varepsilon/x]) \\
\delta_{\mathsf{Id}\lhd\mathcal{G}}(\varepsilon) & = & \varepsilon \quad \text{for } \mathcal{G} \neq \mathsf{Id} \\
\delta_{\mathsf{B}\lhd\mathcal{G}}(b) & = & b \\
\delta_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}(l\langle\varepsilon\rangle) & = & \langle \delta_{\mathcal{F}_1\lhd\mathcal{G}}(\varepsilon), \mathsf{Empty}_{\mathcal{F}_2\lhd\mathcal{G}}\rangle \\
\delta_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}(r\langle\varepsilon\rangle) & = & \langle \mathsf{Empty}_{\mathcal{F}_1\lhd\mathcal{G}}, \delta_{\mathcal{F}_2\lhd\mathcal{G}}(\varepsilon)\rangle \\
\delta_{\mathcal{F}_1\oplus\mathcal{F}_2\lhd\mathcal{G}}(l[\varepsilon]) & = & \kappa_1(\delta_{\mathcal{F}_1\lhd\mathcal{G}}(\varepsilon)) \\
\delta_{\mathcal{F}_1\oplus\mathcal{F}_2\lhd\mathcal{G}}(r[\varepsilon]) & = & \kappa_2(\delta_{\mathcal{F}_2\lhd\mathcal{G}}(\varepsilon)) \\
\delta_{\mathcal{F}^A\lhd\mathcal{G}}(a(\varepsilon)) & = & \lambda a'.\begin{cases} \delta_{\mathcal{F}\lhd\mathcal{G}}(\varepsilon) & \text{if } a = a' \\ \mathsf{Empty}_{\mathcal{F}\lhd\mathcal{G}} & \text{otherwise} \end{cases} \\
\delta_{\mathcal{P}_\mathbb{S}\mathcal{F}\lhd\mathcal{G}}(\{\varepsilon\}) & = & \{\, \delta_{\mathcal{F}\lhd\mathcal{G}}(\varepsilon) \,\}
\end{array}$$

Here, $\varepsilon[\mu x.\varepsilon/x]$ denotes syntactic substitution, replacing every free occurrence of x in ε by $\mu x.\varepsilon$. ♣

In order to see that the definition of $\delta_{\mathcal{F}\lhd\mathcal{G}}$ is well-formed, we have to observe that $\delta_{\mathcal{F}\lhd\mathcal{G}}$ can be seen as a function having two arguments: the type $\mathcal{F}\lhd\mathcal{G}$ and the expression ε. Then, we use induction on the Cartesian product of types and expressions with orders $\preceq$ and $\ll$, respectively. More precisely, given two pairs $\langle \mathcal{F}_1\lhd\mathcal{G}, \varepsilon_1\rangle$ and $\langle \mathcal{F}_2\lhd\mathcal{G}, \varepsilon_2\rangle$ we have an order

$$\langle \mathcal{F}_1\lhd\mathcal{G}, \varepsilon_1\rangle \leq \langle \mathcal{F}_2\lhd\mathcal{G}, \varepsilon_2\rangle \quad \Leftrightarrow \quad \begin{array}{ll} & \text{(i) } (\mathcal{F}_1\lhd\mathcal{G}) \preceq (\mathcal{F}_2\lhd\mathcal{G}) \\ \text{or} & \text{(ii) } (\mathcal{F}_1\lhd\mathcal{G}) = (\mathcal{F}_2\lhd\mathcal{G}) \text{ and } \varepsilon_1 \ll \varepsilon_2 \end{array} \tag{3.1}$$

Observe that in the definition above it is always true that $\langle \mathcal{F}'\lhd\mathcal{G}, \varepsilon'\rangle \leq \langle \mathcal{F}\lhd\mathcal{G}, \varepsilon\rangle$, for all occurrences of $\delta_{\mathcal{F}'\lhd\mathcal{G}}(\varepsilon')$ occurring in the right hand side of the equation defining $\delta_{\mathcal{F}\lhd\mathcal{G}}(\varepsilon)$. In all cases, but the ones that ε is a fixed point or a sum expression, the inequality comes from point (i) above. For the case of the sum, note that $\langle \mathcal{F}\lhd\mathcal{G}, \varepsilon_1\rangle \leq \langle \mathcal{F}\lhd\mathcal{G}, \varepsilon_1\oplus\varepsilon_2\rangle$ and $\langle \mathcal{F}\lhd\mathcal{G}, \varepsilon_2\rangle \leq \langle \mathcal{F}\lhd\mathcal{G}, \varepsilon_1\oplus\varepsilon_2\rangle$ by point (ii), since $N(\varepsilon_1) < N(\varepsilon_1\oplus\varepsilon_2)$ and $N(\varepsilon_2) < N(\varepsilon_1\oplus\varepsilon_2)$. Similarly, in the case of $\mu x.\varepsilon$ we have that $N(\varepsilon) = N(\varepsilon[\mu x.\varepsilon/x])$, which can easily be proved by (standard) induction on the syntactic structure of ε, since ε is guarded (in x), and this guarantees that $N(\varepsilon[\mu x.\varepsilon/x]) < N(\mu x.\varepsilon)$. Hence, $\langle \mathcal{G}\lhd\mathcal{G}, \varepsilon\rangle \leq \langle \mathcal{G}\lhd\mathcal{G}, \mu x.\varepsilon\rangle$. Also note that clause 4 of the above definition overlaps with clauses 1 and 2 (by taking $\mathcal{F} = \mathsf{Id}$). However, they give the same result and thus the function $\delta_{\mathcal{F}\lhd\mathcal{G}}$ is well-defined.

Definition 3.11. We define, for each non-deterministic functor $\mathcal{G}$, a $\mathcal{G}$-coalgebra

$$\delta_\mathcal{G} : \mathsf{Exp}_\mathcal{G} \to \mathcal{G}(\mathsf{Exp}_\mathcal{G})$$

by putting $\delta_\mathcal{G} = \delta_{\mathcal{G}\lhd\mathcal{G}}$. ♣

The function $\delta_\mathcal{G}$ can be thought of as the generalization of the well-known notion of Brzozowski derivative [13] for regular expressions and, moreover, it provides an operational semantics for expressions, as we shall see in Section 3.2.

The observation that the set of expressions has a coalgebra structure will be crucial for the proof of the generalized Kleene theorem, as will be shown in the next two sections.

3.2. **Expressions are expressive.** Having a $\mathcal{G}$-coalgebra structure on $\mathsf{Exp}_{\mathcal{G}}$ has two advantages. First, it provides us, by finality, directly with a natural semantics because of the existence of a (unique) homomorphism $\mathbf{beh}\colon \mathsf{Exp}_{\mathcal{G}} \to \Omega_{\mathcal{G}}$, that assigns to every expression ε an element $\mathbf{beh}(\varepsilon)$ of the final coalgebra $\Omega_{\mathcal{G}}$.

The second advantage of the coalgebra structure on $\mathsf{Exp}_{\mathcal{G}}$ is that it lets us use the notion of $\mathcal{G}$-bisimulation to relate $\mathcal{G}$-coalgebras (S, g) and expressions $\varepsilon \in \mathsf{Exp}_{\mathcal{G}}$. If one can construct a bisimulation relation between an expression ε and a state s of a given coalgebra, then the behaviour represented by ε is equal to the behaviour of the state s. This is the analogue of computing the language $L(r)$ represented by a given regular expression r and the language $L(s)$ accepted by a state s of a finite state automaton and checking whether $L(r) = L(s)$.

The following theorem states that every state in a locally finite $\mathcal{G}$-coalgebra can be represented by an expression in our language. This generalizes *half* of Kleene's theorem for deterministic automata: if a language is accepted by a finite automaton then it is regular (*i.e.* it can be denoted by a regular expression). The generalization of the other *half* of the theorem (if a language is regular then it is accepted by a finite automaton) will be presented in Section 3.3. It is worth to remark that in the usual definition of deterministic automaton the initial state of the automaton is included and, thus, in the original Kleene's theorem, it was enough to consider finite automata. In the coalgebraic approach, the initial state is not explicitly modelled and thus we need to consider locally-finite coalgebras: coalgebras where each state will generate a finite subcoalgebra.

Theorem 3.12. *Let $\mathcal{G}$ be a non-deterministic functor and let (S, g) be a locally-finite $\mathcal{G}$-coalgebra. Then, for any $s \in S$, there exists an expression $\langle\!\langle\, s \,\rangle\!\rangle \in \mathsf{Exp}_{\mathcal{G}}$ such that $s \sim \langle\!\langle\, s \,\rangle\!\rangle$.*

Proof. Let $s \in S$ and let $\langle s \rangle = \{s_1, \ldots, s_n\}$ with $s_1 = s$. We construct, for every state $s_i \in \langle s \rangle$, an expression $\langle\!\langle\, s_i \,\rangle\!\rangle$ such that $s_i \sim \langle\!\langle\, s_i \,\rangle\!\rangle$.

If $\mathcal{G} = \mathsf{Id}$, we set, for every i, $\langle\!\langle\, s_i \,\rangle\!\rangle = \underline{\emptyset}$. It is easy to see that $\{\langle s_i, \underline{\emptyset}\rangle \mid s_i \in \langle s \rangle\}$ is a bisimulation and, thus, we have that $s \sim \langle\!\langle\, s \,\rangle\!\rangle$.

For $\mathcal{G} \neq \mathsf{Id}$, we proceed in the following way. Let, for every i, $A_i = \mu x_i.\gamma^{\mathcal{G}}_{g(s_i)}$ where, for $\mathcal{F} \lhd \mathcal{G}$ and $c \in \mathcal{F}\langle s \rangle$, the expression $\gamma^{\mathcal{F}}_c \in \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}$ is defined by induction on the structure of $\mathcal{F}$:

$$\gamma^{\mathsf{Id}}_{s_i} = x_i \qquad \gamma^{\mathsf{B}}_b = b \qquad \gamma^{\mathcal{F}_1\times\mathcal{F}_2}_{\langle c,c'\rangle} = l\langle\gamma^{\mathcal{F}_1}_c\rangle \oplus r\langle\gamma^{\mathcal{F}_2}_{c'}\rangle \qquad \gamma^{\mathcal{F}^A}_f = \bigoplus_{a\in A} a(\gamma^{\mathcal{F}}_{f(a)})$$

$$\gamma^{\mathcal{F}_1 \oplus \mathcal{F}_2}_{\kappa_1(c)} = l[\gamma^{\mathcal{F}_1}_c] \qquad \gamma^{\mathcal{F}_1 \oplus \mathcal{F}_2}_{\kappa_2(c)} = r[\gamma^{\mathcal{F}_2}_c] \qquad \gamma^{\mathcal{F}_1 \oplus \mathcal{F}_2}_{\bot} = \underline{\emptyset} \qquad \gamma^{\mathcal{F}_1 \oplus \mathcal{F}_2}_{\top} = l[\underline{\emptyset}] \oplus r[\underline{\emptyset}]$$

$$\gamma^{\mathcal{P}_\omega\mathcal{F}}_C = \begin{cases} \bigoplus_{c\in C} \{\gamma^{\mathcal{F}}_c\} & C \neq \emptyset \\ \underline{\emptyset} & \text{otherwise} \end{cases}$$

Note that here the choice of $l[\underline{\emptyset}] \oplus r[\underline{\emptyset}]$ to represent inconsistency is arbitrary but *canonical*, in the sense that any other expression involving sum of $l[\varepsilon_1]$ and $r[\varepsilon_2]$ will be bisimilar. Formally, the definition of γ above is parametrized by a function from $\{s_1, \ldots, s_n\}$ to a fixed set of variables $\{x_1, \ldots, x_n\}$. It should also be noted that $\bigoplus_{i\in I} \varepsilon_i$ stands for $\varepsilon_1 \oplus (\varepsilon_2 \oplus (\varepsilon_3 \oplus \ldots))$ (this is a choice, since later we will axiomatize $\oplus$ to be commutative and associative).

Let $A^0_i = A_i$, define $A^{k+1}_i = A^k_i\{A^k_{k+1}/x_{k+1}\}$ and then set $\langle\!\langle\, s_i \,\rangle\!\rangle = A^n_i$. Here, $A\{A'/x\}$ denotes syntactic replacement (that is, substitution without renaming of bound variables in A which are also free variables in A'). The definition of $\langle\!\langle\, s_i \,\rangle\!\rangle$ does not depend in the

chosen order of $\{s_1, \ldots, s_n\}$: the expressions obtained are just different modulo renaming of variables.

Observe that the term

$$A_i^n = (\mu x_i.\gamma^{\mathcal{G}}_{g(s_i)})\{A_1^0/x_1\}\ldots\{A_n^{n-1}/x_n\}$$

is a closed term because, for every $j = 1, \ldots, n$, the term A_j^{j-1} contains at most $n-j$ free variables in the set $\{x_{j+1}, \ldots, x_n\}$.

It remains to prove that $s_i \sim \langle\!\langle\, s_i \,\rangle\!\rangle$. We show that $R = \{\langle s_i, \langle\!\langle\, s_i \,\rangle\!\rangle\rangle \mid s_i \in \langle s\rangle\}$ is a bisimulation. For that, we define, for $\mathcal{F} \lhd \mathcal{G}$ and $c \in \mathcal{F}\langle s\rangle$, $\xi^{\mathcal{F}}_c = \gamma^{\mathcal{F}}_c\{A_1^0/x_1\}\ldots\{A_n^{n-1}/x_n\}$ and the relation

$$R_{\mathcal{F}\lhd\mathcal{G}} = \{\langle c, \delta_{\mathcal{F}\lhd\mathcal{G}}(\xi^{\mathcal{F}}_c)\rangle \mid c \in \mathcal{F}\langle s\rangle\}.$$

Then, we prove that ① $R_{\mathcal{F}\lhd\mathcal{G}} = \overline{\mathcal{F}}(R)$ and ② $\langle g(s_i), \delta_{\mathcal{G}}(\langle\!\langle\, s_i \,\rangle\!\rangle)\rangle \in R_{\mathcal{G}\lhd\mathcal{G}}$.

① By induction on the structure of $\mathcal{F}$.

$\boxed{\mathcal{F} = \mathsf{Id}}$ Note that $R_{\mathsf{Id}\lhd\mathcal{G}} = \{\langle s_i, \xi^{\mathsf{Id}}_{s_i}\rangle \mid s_i \in \langle s\rangle\}$ which is equal to $\mathsf{Id}(R) = R$ provided that $\xi^{\mathsf{Id}}_{s_i} = \langle\!\langle\, s_i \,\rangle\!\rangle$. The latter is indeed the case:

$$\begin{array}{rcll}
\xi^{\mathsf{Id}}_{s_i} & = & \gamma^{\mathsf{Id}}_{s_i}\{A_1^0/x_1\}\ldots\{A_n^{n-1}/x_n\} & (\text{def. } \xi^{\mathsf{Id}}_{s_i}) \\
& = & x_i\{A_1^0/x_1\}\ldots\{A_n^{n-1}/x_n\} & (\text{def. } \gamma^{\mathsf{Id}}_{s_i}) \\
& = & A_i^{i-1}\{A_{i+1}^i/x_{i+1}\}\ldots\{A_n^{n-1}/x_n\} & (\{A_i^{i-1}/x_i\}) \\
& = & A_i^0\{A_1^0/x_1\}\ldots\{A_n^{n-1}/x_n\} & (\text{def. } A_i^{i-1}) \\
& = & \langle\!\langle\, s_i \,\rangle\!\rangle & (\text{def. } \langle\!\langle\, s_i \,\rangle\!\rangle)
\end{array}$$

$\boxed{\mathcal{F} = \mathsf{B}}$ Note that, for $b \in \mathsf{B}$, $\xi^{\mathsf{B}}_b = \gamma^{\mathsf{B}}_b\{A_1^0/x_1\}\ldots\{A_n^{n-1}/x_n\} = b$. Thus, we have that $R_{\mathsf{B}\lhd\mathcal{G}} = \{\langle s_i, \xi^{\mathsf{B}}_{s_i}\rangle \mid s_i \in \mathsf{B}\langle s\rangle\} = \{\langle b, b\rangle \mid b \in \mathsf{B}\} = \overline{\mathsf{B}}(R)$.

$\boxed{\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2}$

$$\begin{array}{rcll}
 & & \langle\langle u, v\rangle, \langle e, f\rangle\rangle \in \overline{\mathcal{F}_1 \times \mathcal{F}_2}(R) & \\
\iff & & \langle u, e\rangle \in \overline{\mathcal{F}_1}(R) \text{ and } \langle v, f\rangle \in \overline{\mathcal{F}_2}(R) & (\text{def. } \overline{\mathcal{F}_1 \times \mathcal{F}_2}) \\
\iff & & \langle u, e\rangle \in R_{\mathcal{F}_1\lhd\mathcal{G}} \text{ and } \langle v, f\rangle \in R_{\mathcal{F}_2\lhd\mathcal{G}} & (\text{ind. hyp.}) \\
\iff & & \langle u, e\rangle = \langle c, \delta_{\mathcal{F}_1\lhd\mathcal{G}}(\xi^{\mathcal{F}_1}_c)\rangle \text{ and } \langle v, f\rangle = \langle c', \delta_{\mathcal{F}_2\lhd\mathcal{G}}(\xi^{\mathcal{F}_2}_{c'})\rangle & (\text{def. } R_{\mathcal{F}_i\lhd\mathcal{G}}) \\
\iff & & \langle u, v\rangle = \langle c, c'\rangle \text{ and } \langle e, f\rangle = \delta_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}(l(\xi^{\mathcal{F}_1}_c) \oplus r(\xi^{\mathcal{F}_2}_{c'})) & (\text{def. } \delta_{\mathcal{F}\lhd\mathcal{G}}) \\
\iff & & \langle u, v\rangle = \langle c, c'\rangle \text{ and } \langle e, f\rangle = \delta_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}(\xi^{\mathcal{F}_1\times\mathcal{F}_2}_{\langle c,c'\rangle}) & (\text{def. } \xi^{\mathcal{F}}) \\
\iff & & \langle\langle u, v\rangle, \langle e, f\rangle\rangle \in R_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}} &
\end{array}$$

$\boxed{\mathcal{F} = \mathcal{F}_1 \oplus \mathcal{F}_2}$, $\boxed{\mathcal{F} = \mathcal{F}_1^A}$ and $\boxed{\mathcal{F} = \mathcal{P}_\omega\mathcal{F}_1}$: similar to $\mathcal{F}_1 \times \mathcal{F}_2$.

② We want to prove that $\langle g(s_i), \delta_{\mathcal{G}}(\langle\!\langle\, s_i \,\rangle\!\rangle)\rangle \in R_{\mathcal{G}\lhd\mathcal{G}}$. For that, we must show that $g(s_i) \in \mathcal{G}\langle s\rangle$ and $\delta_{\mathcal{G}}(\langle\!\langle\, s_i \,\rangle\!\rangle) = \delta_{\mathcal{G}}(\xi^{\mathcal{G}}_{g(s_i)})$. The former follows by definition of $\langle s\rangle$, whereas for the latter we observe that:

$$\begin{array}{rcll}
 & & \delta_{\mathcal{G}}(\langle\!\langle\, s_i \,\rangle\!\rangle) & \\
 & = & \delta_{\mathcal{G}}((\mu x_i.\gamma^{\mathcal{G}}_{g(s_i)})\{A_1^0/x_1\}\ldots\{A_n^{n-1}/x_n\}) & (\text{def. of } \langle\!\langle\, s_i \,\rangle\!\rangle) \\
 & = & \delta_{\mathcal{G}}(\mu x_i.\gamma^{\mathcal{G}}_{g(s_i)}\{A_1^0/x_1\}\ldots\{A_{i-1}^{i-2}/x_{i-1}\}\{A_{i+1}^i/x_{i+1}\}\ldots\{A_n^{n-1}/x_n\}) &
\end{array}$$

$$
\begin{aligned}
&= \delta_{\mathcal{G}}(\gamma^{\mathcal{G}}_{g(s_i)}\{A^0_1/x_1\}\dots\{A^{i-2}_{i-1}/x_{i-1}\}\{A^i_{i+1}/x_{i+1}\}\dots\{A^{n-1}_n/x_n\}[A^n_i/x_i]) \quad \text{(def. of } \delta_{\mathcal{G}}) \\
&= \delta_{\mathcal{G}}(\gamma^{\mathcal{G}}_{g(s_i)}\{A^0_1/x_1\}\dots\{A^{i-2}_{i-1}/x_{i-1}\}\{A^i_{i+1}/x_{i+1}\}\dots\{A^{n-1}_n/x_n\}\{A^n_i/x_i\}) \quad ([A^n_i/x_i] = \{A^n_i/x_i\}) \\
&= \delta_{\mathcal{G}}(\gamma^{\mathcal{G}}_{g(s_i)}\{A^0_1/x_1\}\dots\{A^{i-2}_{i-1}/x_{i-1}\}\{A^n_i/x_i\}\{A^i_{i+1}/x_{i+1}\}\dots\{A^{n-1}_n/x_n\}) \\
&= \delta_{\mathcal{G}}(\xi^{\mathcal{G}}_{g(s_i)})
\end{aligned}
$$

Here, note that $[A^n_i/x_i] = \{A^n_i/x_i\}$, because A^n_i has no free variables. The last two steps follow, respectively, because x_i is not free in $A^i_{i+1}, \dots, A^{n-1}_n$ and:

$$
\begin{aligned}
& \{A^n_i/x_i\}\{A^i_{i+1}/x_{i+1}\}\dots\{A^{n-1}_n/x_n\} \\
= \;& \{A^{i-1}_i\{A^i_{i+1}/x_{i+1}\}\dots\{A^{n-1}_n/x_n\}/x_i\}\{A^i_{i+1}/x_{i+1}\}\dots\{A^{n-1}_n/x_n\} \\
= \;& \{A^{i-1}_i/x_i\}\{A^i_{i+1}/x_{i+1}\}\dots\{A^{n-1}_n/x_n\} \qquad (3.2)
\end{aligned}
$$

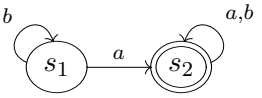
Equation (3.2) uses the syntactic identity

$$A\{B\{C/y\}/x\}\{C/y\} = A\{B/x\}\{C/y\}, \quad y \text{ not free in } C \qquad (3.3)$$

□

Let us illustrate the construction appearing in the proof of Theorem 3.12 by some examples. These examples will illustrate the similarity with the proof of Kleene's Theorem presented in most textbooks, where a regular expression denoting the language recognized by a state of a deterministic automaton is built using a system of equations.

Consider the following deterministic automaton over $A = \{a, b\}$, whose transition function g is given by the following picture ($\textcircled{\textcircled{s}}$ represents that the state s is final):



We define $A_1 = \mu x_1.\gamma^{\mathcal{D}}_{g(s_1)}$ and $A_2 = \mu x_2.\gamma^{\mathcal{D}}_{g(s_2)}$ where

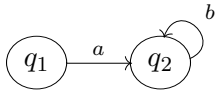
$$\gamma^{\mathcal{D}}_{g(s_1)} = l\langle 0\rangle \oplus r\langle b(x_1) \oplus a(x_2)\rangle \qquad \gamma^{\mathcal{D}}_{g(s_2)} = l\langle 1\rangle \oplus r\langle a(x_2) \oplus b(x_2)\rangle$$

We have $A^2_1 = A_1\{A^1_2/x_2\}$ and $A^2_2 = A_2\{A^0_1/x_1\}$. Thus, $\langle\!\langle\, s_2 \,\rangle\!\rangle = A_2$ and, since $A^1_2 = A_2$, $\langle\!\langle\, s_1 \,\rangle\!\rangle$ is the expression

$$\mu x_1.\, l\langle 0\rangle \oplus r\langle b(x_1) \oplus a(\mu x_2.\, l\langle 1\rangle \oplus r\langle a(x_2) \oplus b(x_2)\rangle)\rangle$$

By construction we have $s_1 \sim \langle\!\langle\, s_1 \,\rangle\!\rangle$ and $s_2 \sim \langle\!\langle\, s_2 \,\rangle\!\rangle$.

For another example, take the following partial automaton, also over a two letter alphabet $A = \{a, b\}$:



In the graphical representation of a partial automaton (S, p) we omit transitions for which $p(s)(a) = \kappa_1(*)$. In this case, this happens in q_1 for the input letter b and in q_2 for a.

We will have the equations

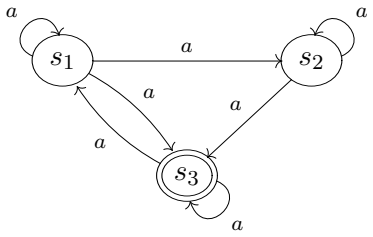
$$A_1 = A^0_1 = A^1_1 = \mu x_1.b(l[*]) \oplus a(r[x_2])$$
$$A_2 = A^0_2 = A^1_2 = \mu x_2.a(l[*]) \oplus b(r[x_2])$$

Thus:

$$\langle\!\langle\, s_1 \,\rangle\!\rangle = A_1^2 = \mu x_1.\, b(l[*]) \oplus a(r[\mu x_2.\, a(l[*]) \oplus b(r[x_2])])$$
$$\langle\!\langle\, s_2 \,\rangle\!\rangle = \mu x_2.a(l[*]) \oplus b(r[x_2])$$

Again we have $s_1 \sim \langle\!\langle\, s_1 \,\rangle\!\rangle$ and $s_2 \sim \langle\!\langle\, s_2 \,\rangle\!\rangle$.

As a last example, let us consider the following non-deterministic automaton, over a one letter alphabet $A = \{a\}$:



We start with the equations:

$$\begin{aligned} A_1 &= \mu x_1.l\langle 0\rangle \oplus r\langle a(\{x_1\} \oplus \{x_2\} \oplus \{x_3\})\rangle \\ A_2 &= \mu x_2.l\langle 0\rangle \oplus r\langle a(\{x_2\} \oplus \{x_3\})\rangle \\ A_3 &= \mu x_3.l\langle 1\rangle \oplus r\langle a(\{x_1\} \oplus \{x_3\})\rangle \end{aligned}$$

Then we have the following iterations:

$$\begin{aligned} A_1^1 &= A_1 \\ A_1^2 &= A_1\{A_2^1/x_2\} = \mu x_1.l\langle 0\rangle \oplus r\langle a(\{x_1\} \oplus \{A_2\} \oplus \{x_3\})\rangle \\ A_1^3 &= A_1\{A_2^1/x_2\}\{A_3^2/x_3\} = \mu x_1.l\langle 0\rangle \oplus r\langle a(\{x_1\} \oplus \{(A_2\{A_3^2/x_3\})\} \oplus \{A_3^2\})\rangle \end{aligned}$$

$$\begin{aligned} A_2^1 &= A_2\{A_1/x_1\} = A_2 \\ A_2^2 &= A_2\{A_1/x_1\} = A_2 \\ A_2^3 &= A_2\{A_1/x_1\}\{A_3^2/x_3\} = \mu x_2.l\langle 0\rangle \oplus r\langle a(\{x_2\} \oplus \{A_3^2\})\rangle \end{aligned}$$

$$\begin{aligned} A_3^1 &= A_3\{A_1/x_1\} = \mu x_3.l\langle 1\rangle \oplus r\langle a(\{A_1\} \oplus \{x_3\})\rangle \\ A_3^2 &= A_3\{A_1/x_1\}\{A_2^1/x_2\} = \mu x_3.l\langle 1\rangle \oplus r\langle a(\{(A_1\{A_2^1/x_2\})\} \oplus \{x_3\})\rangle \\ A_3^3 &= A_3^2 \end{aligned}$$

This yields the following expressions:

$$\langle\!\langle\, s_1 \,\rangle\!\rangle = \mu x_1.l\langle 0\rangle \oplus r\langle a(\{x_1\} \oplus \{\langle\!\langle\, s_2 \,\rangle\!\rangle\}) \oplus \{\langle\!\langle\, s_3 \,\rangle\!\rangle\})\rangle$$
$$\langle\!\langle\, s_2 \,\rangle\!\rangle = \mu x_2.l\langle 0\rangle \oplus r\langle a(\{x_2\} \oplus \{\langle\!\langle\, s_3 \,\rangle\!\rangle\})\rangle$$
$$\langle\!\langle\, s_3 \,\rangle\!\rangle = \mu x_3.l\langle 1\rangle \oplus r\langle a(\{\mu x_1.l\langle 0\rangle \oplus r\langle a(\{x_1\} \oplus \{\mu x_2.l\langle 0\rangle \oplus r\langle a(\{x_2\} \oplus \{x_3\})\rangle\} \oplus \{x_3\})\rangle\} \oplus \{x_3\})\rangle$$

3.3. **Finite systems for expressions.** Next, we prove the converse of Theorem 3.12, that is, we show how to construct a *finite* $\mathcal{G}$-coalgebra (S, g) from an arbitrary expression $\varepsilon \in \mathsf{Exp}_{\mathcal{G}}$, such that there exists a state $s \in S$ with $\varepsilon \sim_{\mathcal{G}} s$.

The immediate way of obtaining a coalgebra from an expression $\varepsilon \in \mathsf{Exp}_{\mathcal{G}}$ is to compute the subcoalgebra $\langle\varepsilon\rangle$, since we have provided the set $\mathsf{Exp}_{\mathcal{G}}$ with a coalgebra structure $\delta_{\mathcal{G}} \colon \mathsf{Exp}_{\mathcal{G}} \to \mathcal{G}(\mathsf{Exp}_{\mathcal{G}})$. However, the subcoalgebra generated by an expression $\varepsilon \in \mathsf{Exp}_{\mathcal{G}}$ by repeatedly applying $\delta_{\mathcal{G}}$ is, in general, infinite. Take for instance the deterministic expression $\varepsilon_1 = \mu x.\, r\langle a(x \oplus \mu y.\, r\langle a(y)\rangle)\rangle$ (for simplicity, we consider $A = \{a\}$ and below we will write,

in the second component of $\delta_{\mathcal{D}}$, an expression ε instead of the function mapping a to ε) and observe that:

$$\begin{array}{lcl}
\delta_{\mathcal{D}}(\varepsilon_1) & = & \langle 0, \varepsilon_1 \oplus \mu y.\, r\langle a(y)\rangle\rangle \\
\delta_{\mathcal{D}}(\varepsilon_1 \oplus \mu y.\, r\langle a(y)\rangle) & = & \langle 0, \varepsilon_1 \oplus \mu y.\, r\langle a(y)\rangle \oplus \mu y.\, r\langle a(y)\rangle\rangle \\
\delta_{\mathcal{D}}(\varepsilon_1 \oplus \mu y.\, r\langle a(y)\rangle \oplus \mu y.\, r\langle a(y)\rangle) & = & \langle 0, \varepsilon_1 \oplus \mu y.\, r\langle a(y)\rangle \oplus \mu y.\, r\langle a(y)\rangle \oplus \mu y.\, r\langle a(y)\rangle\rangle \\
 & \vdots &
\end{array}$$

As one would expect, all the new states are equivalent and will be identified by **beh** (the morphism into the final coalgebra). However, the function $\delta_{\mathcal{D}}$ does not make any state identification and thus yields an infinite coalgebra.

This phenomenon occurs also in classical regular expressions. It was shown in [13] that normalizing the expressions using the axioms for associativity, commutativity and idempotency was enough to guarantee finiteness[1]. We will show in this section that this also holds in our setting.

Consider the following axioms (only the first three are essential, but we include the fourth to obtain smaller coalgebras):

$$\begin{array}{ll}
(\mathit{Associativity}) & \varepsilon_1 \oplus (\varepsilon_2 \oplus \varepsilon_3) \equiv (\varepsilon_1 \oplus \varepsilon_2) \oplus \varepsilon_3 \\
(\mathit{Commutativity}) & \varepsilon_1 \oplus \varepsilon_2 \equiv \varepsilon_2 \oplus \varepsilon_1 \\
(\mathit{Idempotency}) & \varepsilon \oplus \varepsilon \equiv \varepsilon \\
(\mathit{Empty}) & \underline{\emptyset} \oplus \varepsilon \equiv \varepsilon
\end{array}$$

We define the relation $\equiv_{ACIE} \subseteq \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}} \times \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}$, written infix, as the least equivalence relation containing the four identities above. The relation $\equiv_{ACIE}$ gives rise to the (surjective) equivalence map $[\varepsilon]_{ACIE} = \{\varepsilon' \mid \varepsilon \equiv_{ACIE} \varepsilon'\}$. The following diagram shows the maps defined so far:

$$\begin{array}{ccc}
\mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}} & \xrightarrow{[-]_{ACIE}} & \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}/_{\equiv_{ACIE}} \\
{\scriptstyle \delta_{\mathcal{F}\lhd\mathcal{G}}}\downarrow & & \\
\mathcal{F}(\mathsf{Exp}_{\mathcal{G}}) & \xrightarrow[\mathcal{F}([-]_{ACIE})]{} & \mathcal{F}(\mathsf{Exp}_{\mathcal{G}}/_{\equiv_{ACIE}})
\end{array}$$

In order to complete the diagram, we next prove that $\equiv_{ACIE}$ is contained in the kernel of $\mathcal{F}([-]_{ACIE}) \circ \delta_{\mathcal{F}\lhd\mathcal{G}}$[2].

This will guarantee the existence of a function

$$\overline{\delta}_{\mathcal{F}\lhd\mathcal{G}} \colon \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}/_{\equiv_{ACIE}} \to \mathcal{F}(\mathsf{Exp}_{\mathcal{G}}/_{\equiv_{ACIE}})$$

which, when $\mathcal{F} = \mathcal{G}$, provides $\mathsf{Exp}_{\mathcal{G}}/_{\equiv}$ with a coalgebraic structure

$$\overline{\delta}_{\mathcal{G}} \colon \mathsf{Exp}_{\mathcal{G}}/_{\equiv_{ACIE}} \to \mathcal{G}(\mathsf{Exp}_{\mathcal{G}}/_{\equiv_{ACIE}})$$

(as before we write $\overline{\delta}_{\mathcal{G}}$ for $\overline{\delta}_{\mathcal{G}\lhd\mathcal{G}}$) and which makes $[-]_{ACIE}$ a homomorphism of coalgebras.

[1] Actually, to guarantee finiteness, similar to classical regular expressions, it is enough to eliminate double occurrences of expressions ε at the outermost level of an expression $\cdots \oplus \varepsilon \oplus \cdots \oplus \varepsilon \oplus \cdots$ (and to do this one needs the ACI axioms). Note that this is weaker than taking expressions modulo the ACI axioms: for instance, the expressions $\varepsilon_1 \oplus \varepsilon_2$ and $\varepsilon_2 \oplus \varepsilon_1$, for $\varepsilon_1 \neq \varepsilon_2$, would not be identified in the process above.

[2] This is equivalent to prove that $\mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}/_{\equiv_{ACIE}}$, together with $[-]_{ACIE}$, is the coequalizer of the projection morphisms from $\equiv_{ACIE}$ to $\mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}$.

Lemma 3.13. *Let $\mathcal{G}$ and $\mathcal{F}$ be non-deterministic functors, with $\mathcal{F} \lhd \mathcal{G}$. For all $\varepsilon_1, \varepsilon_2 \in \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}$,*

$$\varepsilon_1 \equiv_{ACIE} \varepsilon_2 \Rightarrow (\mathcal{F}([-]_{ACIE}))(\delta_{\mathcal{F}\lhd\mathcal{G}}(\varepsilon_1)) = (\mathcal{F}([-]_{ACIE}))(\delta_{\mathcal{F}\lhd\mathcal{G}}(\varepsilon_2))$$

Proof. In order to improve readability, in this proof we will use $[-]$ to denote $[-]_{ACIE}$.

It is enough to prove that for all $x_1, x_2, x_3 \in \mathcal{F}(\mathsf{Exp}_{\mathcal{G}})$:

① $\mathcal{F}([-])(\mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}}(\mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}}(x_1, x_2), x_3)) = \mathcal{F}([-])(\mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}}(x_1, \mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}}(x_2, x_3)))$
② $\mathcal{F}([-])(\mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}}(x_1, x_2)) = \mathcal{F}([-])(\mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}}(x_2, x_1))$
③ $\mathcal{F}([-])(\mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}}(x_1, x_1)) = \mathcal{F}([-])(x_1)$
④ $\mathcal{F}([-])(\mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}}(\mathsf{Empty}_{\mathcal{F}\lhd\mathcal{G}}, x_1)) = \mathcal{F}([-])(x_1)$

By induction on the structure of $\mathcal{F}$. We illustrate a few cases, the omitted ones are proved in a similar way.

$\boxed{\mathcal{F} = \mathsf{Id}}$ $x_1, x_2, x_3 \in \mathsf{Exp}_{\mathcal{G}}$

$$\begin{array}{lll}
① & [\mathsf{Plus}_{\mathsf{Id}\lhd\mathcal{G}}(\mathsf{Plus}_{\mathsf{Id}\lhd\mathcal{G}}(x_1, x_2), x_3)] & \\
= & [(x_1 \oplus x_2) \oplus x_3] & (\text{def. } \mathsf{Plus}) \\
= & [x_1 \oplus (x_2 \oplus x_3)] & (\textit{Associativity}) \\
= & [\mathsf{Plus}_{\mathsf{Id}\lhd\mathcal{G}}(x_1, \mathsf{Plus}_{\mathsf{Id}\lhd\mathcal{G}}(x_2, x_3))] & (\text{def. } \mathsf{Plus}) \\
 & & \\
④ & [\mathsf{Plus}_{\mathsf{Id}\lhd\mathcal{G}}(\mathsf{Empty}_{\mathsf{Id}\lhd\mathcal{G}}, x_1)] & \\
= & [\underline{\emptyset} \oplus x_1] & (\text{def. } \mathsf{Plus} \text{ and } \mathsf{Empty}) \\
= & [x_1] & (\mathsf{Empty})
\end{array}$$

$\boxed{\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2}$ $x_1 = \langle u_1, v_1\rangle, x_2 = \langle u_2, v_2\rangle \in (\mathcal{F}_1 \times \mathcal{F}_2)(\mathsf{Exp}_{\mathcal{G}})$

$$\begin{array}{lll}
② & (\mathcal{F}_1 \times \mathcal{F}_2)([-])(\mathsf{Plus}_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}(\langle u_1, v_1\rangle, \langle u_2, v_2\rangle)) & \\
= & \langle \mathcal{F}_1([-])(\mathsf{Plus}_{\mathcal{F}_1\lhd\mathcal{G}}(u_1, u_2)), \mathcal{F}_2([-])(\mathsf{Plus}_{\mathcal{F}_2\lhd\mathcal{G}}(v_1, v_2))\rangle & (\text{def. } \mathsf{Plus}) \\
= & \langle \mathcal{F}_1([-])(\mathsf{Plus}_{\mathcal{F}_1\lhd\mathcal{G}}(u_2, u_1)), \mathcal{F}_2([-])(\mathsf{Plus}_{\mathcal{F}_2\lhd\mathcal{G}}(v_2, v_1))\rangle & (\text{ind. hyp.}) \\
= & (\mathcal{F}_1 \times \mathcal{F}_2)([-])(\mathsf{Plus}_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}(\langle u_2, v_2\rangle, \langle u_1, v_1\rangle)) & (\text{def. } \mathsf{Plus}) \\
 & & \\
③ & (\mathcal{F}_1 \times \mathcal{F}_2)([-])(\mathsf{Plus}_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}(\langle u_1, v_1\rangle, \langle u_1, v_1\rangle)) & \\
= & \langle \mathcal{F}_1([-])(\mathsf{Plus}_{\mathcal{F}_1\lhd\mathcal{G}}(u_1, u_1)), \mathcal{F}_2([-])(\mathsf{Plus}_{\mathcal{F}_2\lhd\mathcal{G}}(v_1, v_1))\rangle & (\text{def. } \mathsf{Plus}) \\
= & \langle \mathcal{F}_1([-])(u_1), \mathcal{F}_2([-])(v_1)\rangle & (\text{ind. hyp.}) \\
= & (\mathcal{F}_1 \times \mathcal{F}_2)([-])(\langle u_1, v_1\rangle) &
\end{array}$$

$\boxed{\mathcal{F} = \mathcal{P}_\omega\mathcal{F}_1}$ $x_1, x_2, x_3 \in \mathcal{P}_\omega\mathcal{F}_1(\mathsf{Exp}_{\mathcal{G}})$

$$\begin{array}{lll}
① & \mathcal{P}_\omega\mathcal{F}_1([-])(\mathsf{Plus}_{\mathcal{P}_\omega\mathcal{F}_1\lhd\mathcal{G}}(x_1, \mathsf{Plus}_{\mathcal{P}_\omega\mathcal{F}_1\lhd\mathcal{G}}(x_2, x_3))) & \\
= & \mathcal{P}_\omega\mathcal{F}_1([-])(x_1 \cup (x_2 \cup x_3)) & (\text{def. } \mathsf{Plus}) \\
= & \mathcal{P}_\omega\mathcal{F}_1([-])((x_1 \cup x_2) \cup x_3) & \\
= & \mathcal{P}_\omega\mathcal{F}_1([-])(\mathsf{Plus}_{\mathcal{P}_\omega\mathcal{F}_1\lhd\mathcal{G}}(\mathsf{Plus}_{\mathcal{P}_\omega\mathcal{F}_1\lhd\mathcal{G}}(x_1, x_2), x_3)) & (\text{def. } \mathsf{Plus})
\end{array}$$

In the last but one step, we use the fact that, for any set X, $(\mathcal{P}_\omega(X), \cup, \emptyset)$ is a join-semilattice (hence, $x_1 \cup (x_2 \cup x_3) = (x_1 \cup x_2) \cup x_3$). Due to this fact, in the case $\mathcal{F} = \mathcal{P}_\omega\mathcal{F}_1$, in this particular proof, the induction hypothesis will not be used. □

Thus, we have a well-defined function

$$\overline{\delta}_{\mathcal{F}\lhd\mathcal{G}}\colon \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}/_{\equiv_{ACIE}} \to \mathcal{F}(\mathsf{Exp}_{\mathcal{G}}/_{\equiv_{ACIE}})$$

such that $\overline{\delta}_{\mathcal{F}\lhd\mathcal{G}}([\varepsilon]_{ACIE}) = (\mathcal{F}[-]_{ACIE})(\delta_{\mathcal{F}\lhd\mathcal{G}}(\varepsilon))$.

We are ready to state and prove the second half of Kleene's theorem.

Theorem 3.14. *Let $\mathcal{G}$ be a non-deterministic functor. For every $\varepsilon \in \mathsf{Exp}_{\mathcal{G}}$, there exists $\Delta_{\mathcal{G}}(\varepsilon) = (S, g)$ such that S is finite and there exists $s \in S$ with $\varepsilon \sim s$.*

Proof. For every $\varepsilon \in \mathsf{Exp}_{\mathcal{G}}$, we set $\Delta_{\mathcal{G}}(\varepsilon) = \langle [\varepsilon]_{ACIE}\rangle$ (recall that $\langle s\rangle$ denotes the smallest subcoalgebra generated by s). First note that, by Lemma 3.13, the map $[-]_{ACIE}$ is a homomorphism and thus $\varepsilon \sim [\varepsilon]_{ACIE}$. We prove, for every $\varepsilon \in \mathsf{Exp}_{\mathcal{G}}$, that the subcoalgebra $\langle [\varepsilon]_{ACIE}\rangle = (V, \overline{\delta}_{\mathcal{G}})$ has a finite state space V (here, $\overline{\delta}_{\mathcal{G}}$ actually stands for the restriction of $\overline{\delta}_{\mathcal{G}}$ to V). Again, in order to improve readability, below we will use $[-]$ to denote $[-]_{ACIE}$.

More precisely, we prove, for all $\varepsilon \in \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}$, the following inclusion

$$V \subseteq \overline{V} = \{[\varepsilon_1 \oplus \ldots \oplus \varepsilon_k] \mid \varepsilon_1, \ldots, \varepsilon_k \in cl(\varepsilon) \text{ all distinct}, \varepsilon_1, \ldots, \varepsilon_k \in \mathsf{Exp}_{\mathcal{G}}\} \tag{3.4}$$

Here, if $k = 0$ we take the sum above to be $\underline{\emptyset}$ and $cl(\varepsilon)$ denotes the smallest set containing all subformulas of ε and the unfoldings of μ (sub)formulas, that is, the smallest subset satisfying:

$$\begin{array}{lcllcl}
cl(\underline{\emptyset}) & = & \{\underline{\emptyset}\} & cl(l[\varepsilon_1]) & = & \{l[\varepsilon_1]\} \cup cl(\varepsilon_1)\\
cl(\varepsilon_1 \oplus \varepsilon_2) & = & \{\varepsilon_1 \oplus \varepsilon_2\} \cup cl(\varepsilon_1) \cup cl(\varepsilon_2) & cl(r[\varepsilon_1]) & = & \{r[\varepsilon_1]\} \cup cl(\varepsilon_1)\\
cl(\mu x.\varepsilon_1) & = & \{\mu x.\,\varepsilon_1\} \cup cl(\varepsilon_1[\mu x.\varepsilon_1/x]) & cl(a(\varepsilon_1)) & = & \{a(\varepsilon_1)\} \cup cl(\varepsilon_1)\\
cl(l\langle\varepsilon_1\rangle) & = & \{l\langle\varepsilon_1\rangle\} \cup cl(\varepsilon_1) & cl(\{\varepsilon_1\}) & = & \{\{\varepsilon_1\}\} \cup cl(\varepsilon_1)\\
cl(r\langle\varepsilon_1\rangle) & = & \{r\langle\varepsilon_1\rangle\} \cup cl(\varepsilon_1) & &&
\end{array}$$

Note that the set $cl(\varepsilon)$ is finite (the number of different unfoldings is finite) and has the property $\varepsilon \in cl(\varepsilon)$.

We prove the inclusion in equation (3.4) in the following way. First, we observe that $[\varepsilon] \in \overline{V}$, because $\varepsilon \in cl(\varepsilon)$. Then, we prove that $(\overline{V}, \overline{\delta}_{\mathcal{G}})$ (again, $\overline{\delta}_{\mathcal{G}}$ actually stands for the restriction of $\overline{\delta}_{\mathcal{G}}$ to $\overline{V}$) is a subcoalgebra of $(\mathsf{Exp}_{\mathcal{G}}, \overline{\delta}_{\mathcal{G}})$. Thus, $V \subseteq \overline{V}$, since V, the state space of $\langle[\varepsilon]\rangle$ is equal to the intersection of all subcoalgebras of $(\mathsf{Exp}_{\mathcal{G}}, \overline{\delta}_{\mathcal{G}})$ containing $[\varepsilon]$.

To prove that $(\overline{V}, \overline{\delta}_{\mathcal{G}})$ is a subcoalgebra we prove that, for $\varepsilon_1, \ldots, \varepsilon_k \in \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}$,

$$\varepsilon_1, \ldots, \varepsilon_k \in cl(\varepsilon) \text{ all distinct} \Rightarrow \overline{\delta}_{\mathcal{F}\lhd\mathcal{G}}([\varepsilon_1 \oplus \ldots \oplus \varepsilon_k]) \in \mathcal{F}(\overline{V}) \tag{3.5}$$

The intended result then follows by taking $\mathcal{F} = \mathcal{G}$.

We first prove two auxiliary results, by induction on the structure of $\mathcal{F}$:

① $(\mathcal{F}[-])(\mathsf{Empty}_{\mathcal{F}\lhd\mathcal{G}}) \in \mathcal{F}(\overline{V})$

② $(\mathcal{F}[-])(\mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}}(u, v)) \in \mathcal{F}(\overline{V}) \Leftrightarrow (\mathcal{F}[-])(u) \in \mathcal{F}(\overline{V})$ and $(\mathcal{F}[-])(v) \in \mathcal{F}(\overline{V})$

for $u, v \in \mathcal{F}(\mathsf{Exp}_{\mathcal{G}})$.

$\boxed{\mathcal{F} = \mathsf{Id}}$

① $(\mathcal{F}[-])(\mathsf{Empty}_{\mathcal{F}\lhd\mathcal{G}}) = [\underline{\emptyset}] \in \overline{V}$

② $(\mathcal{F}[-])(\mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}}(u, v)) = [u \oplus v] \in \overline{V} \Leftrightarrow [u] \in \overline{V}$ and $[v] \in \overline{V} \quad u, v \in \mathsf{Exp}_{\mathcal{G}}$

The right to left implication follows because, using the (*Associativity*), (*Commutativity*) and (*Idempotency*) axioms, we can rewrite $u \oplus v$ as $\varepsilon_1 \oplus \ldots \oplus \varepsilon_k$, with all $\varepsilon_1, \ldots, \varepsilon_k \in cl(\varepsilon)$ distinct.

$\boxed{\mathcal{F} = \mathsf{B}}$

① $(\mathsf{B}[-])(\mathsf{Empty}_{\mathsf{B}\triangleleft\mathcal{G}}) = \bot_{\mathsf{B}} \in \mathsf{B}(\overline{V})$

② $(\mathsf{B}[-])(\mathsf{Plus}_{\mathsf{B}\triangleleft\mathcal{G}}(u,v)) = u \vee v \in \mathsf{B}(\overline{V}) \Leftrightarrow u \in \mathsf{B}(\overline{V})$ and $v \in \mathsf{B}(\overline{V})$ $u, v \in \mathsf{B}(\mathsf{Exp}_{\mathcal{G}}) = \mathsf{B}$

$\boxed{\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2}$

① $(\mathcal{F}_1 \times \mathcal{F}_2[-])(\mathsf{Empty}_{\mathcal{F}_1\times\mathcal{F}_2\triangleleft\mathcal{G}})$
$= \langle (\mathcal{F}_1[-])(\mathsf{Empty}_{\mathcal{F}_1\triangleleft\mathcal{G}}), (\mathcal{F}_2[-])(\mathsf{Empty}_{\mathcal{F}_2\triangleleft\mathcal{G}})\rangle \in \mathcal{F}_1 \times \mathcal{F}_2(\overline{V})$

② $(\mathcal{F}_1 \times \mathcal{F}_2[-])(\mathsf{Plus}_{\mathcal{F}_1\times\mathcal{F}_2\triangleleft\mathcal{G}}(\langle u_1, u_2\rangle, \langle v_1, v_2\rangle)) =$
$\langle (\mathcal{F}_1[-])(\mathsf{Plus}_{\mathcal{F}_1\triangleleft\mathcal{G}}(u_1, v_1)), (\mathcal{F}_2[-])(\mathsf{Plus}_{\mathcal{F}_2\triangleleft\mathcal{G}}(u_2, v_2))\rangle \in \mathcal{F}_1 \times \mathcal{F}_2(\overline{V})$
$\overset{(IH)}{\Leftrightarrow} u_1, v_1 \in \mathcal{F}_1(\overline{V})$ and $u_2, v_2 \in \mathcal{F}_2(\overline{V})$
$\Leftrightarrow \langle u, v\rangle \in \mathcal{F}_1 \times \mathcal{F}_2(\overline{V}), \quad u = \langle u_1, u_2\rangle, v = \langle v_1, v_2\rangle \in \mathcal{F}_1 \times \mathcal{F}_2(\mathsf{Exp}_{\mathcal{G}})$

$\boxed{\mathcal{F} = \mathcal{F}_1 \oplus \mathcal{F}_2}$ and $\boxed{\mathcal{F} = \mathcal{F}_1^A}$: similar to $\mathcal{F}_1 \times \mathcal{F}_2$.

$\boxed{\mathcal{F} = \mathcal{P}_\omega\mathcal{F}_1}$

① $(\mathcal{P}_\omega\mathcal{F}[-])(\mathsf{Empty}_{\mathcal{P}_\omega\mathcal{F}\triangleleft\mathcal{G}}) = \emptyset \in \mathcal{P}_\omega\mathcal{F}(\overline{V})$

② $(\mathcal{P}_\omega\mathcal{F}[-])(\mathsf{Plus}_{\mathcal{P}_\omega\mathcal{F}\triangleleft\mathcal{G}}(u,v)) = ((\mathcal{P}_\omega\mathcal{F}[-])(u) \cup (\mathcal{P}_\omega\mathcal{F}[\_])(v)) \in \mathcal{P}_\omega\mathcal{F}(\overline{V})$
$\Leftrightarrow (\mathcal{P}_\omega\mathcal{F}[-](u)) \in \mathcal{P}_\omega\mathcal{F}(\overline{V})$ and $(\mathcal{P}_\omega\mathcal{F}[-](v)) \in \mathcal{P}_\omega\mathcal{F}(\overline{V})$

Using ②, we can simplify our proof goal (equation (3.5)) as follows:

$$\overline{\delta}_{\mathcal{F}\triangleleft\mathcal{G}}([\varepsilon_1 \oplus \ldots \oplus \varepsilon_k]) \in \mathcal{F}(\overline{V}) \Leftrightarrow (\mathcal{F}[-])(\delta_{\mathcal{F}\triangleleft\mathcal{G}}(\varepsilon_i)) \in \mathcal{F}(\overline{V}),\ \varepsilon_i \in cl(\varepsilon),\ i = 1, \ldots, k$$

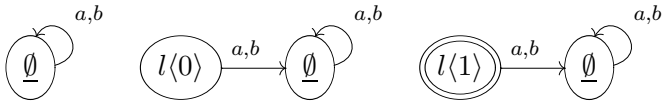
Next, using induction on the product of types of expressions and expressions (using the order in equation (3.1)), ① and ②, we prove that $(\mathcal{F}[-])(\delta_{\mathcal{F}\triangleleft\mathcal{G}}(\varepsilon_i)) \in \mathcal{F}(\overline{V})$, for any $\varepsilon_i \in cl(\varepsilon)$.

$(\mathcal{F}[-])(\delta_{\mathcal{F}\triangleleft\mathcal{G}}(\underline{\emptyset})) = (\mathcal{F}[-])(\mathsf{Empty}_{\mathcal{F}\triangleleft\mathcal{G}}) \in \mathcal{F}(\overline{V})$	(by ①)
$(\mathcal{F}[-])(\overline{\delta}_{\mathcal{F}\triangleleft\mathcal{G}}(\varepsilon_1 \oplus \varepsilon_2)) = (\mathcal{F}[-])(\mathsf{Plus}_{\mathcal{F}\triangleleft\mathcal{G}}(\delta_{\mathcal{F}\triangleleft\mathcal{G}}(\varepsilon_1), \delta_{\mathcal{F}\triangleleft\mathcal{G}}(\varepsilon_2)) \in \mathcal{F}(\overline{V})$	(IH and ②)
$(\mathcal{G}[-])(\delta_{\mathcal{G}\triangleleft\mathcal{G}}(\mu x.\varepsilon)) = (\mathcal{G}[-])(\delta_{\mathcal{G}\triangleleft\mathcal{G}}(\varepsilon[\mu x.\varepsilon/x])) \in \mathcal{G}(\overline{V})$	(IH)
$(\mathsf{Id}[-])(\delta_{\mathsf{Id}\triangleleft\mathcal{G}}(\varepsilon_i)) = [\varepsilon_i] \in \mathsf{Id}(\overline{V})$ for $\mathcal{G} \neq \mathsf{Id}$	($\varepsilon_i \in cl(\varepsilon)$)
$(\mathsf{B}[-])\delta_{\mathsf{B}\triangleleft\mathcal{G}}(b) = b \in \mathsf{B}(\overline{V})$	($\mathsf{B}(\overline{V}) = \mathsf{B}$)
$(\mathcal{F}_1 \times \mathcal{F}_2[-])(\delta_{\mathcal{F}_1\times\mathcal{F}_2\triangleleft\mathcal{G}}(l\langle\varepsilon\rangle))$ $= \langle (\mathcal{F}_1[-])(\delta_{\mathcal{F}_1\triangleleft\mathcal{G}}(\varepsilon)), (\mathcal{F}_2[-])(\mathsf{Empty}_{\mathcal{F}_2\triangleleft\mathcal{G}})\rangle \in \mathcal{F}_1 \times \mathcal{F}_2(\overline{V})$	(IH and ①)
$(\mathcal{F}_1 \times \mathcal{F}_2[-])(\delta_{\mathcal{F}_1\times\mathcal{F}_2\triangleleft\mathcal{G}}(r\langle\varepsilon\rangle))$ $= \langle (\mathcal{F}_1[-])(\mathsf{Empty}_{\mathcal{F}_1\triangleleft\mathcal{G}}), (\mathcal{F}_2[-])(\delta_{\mathcal{F}_2\triangleleft\mathcal{G}}(\varepsilon))\rangle \in \mathcal{F}_1 \times \mathcal{F}_2(\overline{V})$	(IH and ①)
$(\mathcal{F}_1 \oplus \mathcal{F}_2[-])(\delta_{\mathcal{F}_1\oplus\mathcal{F}_2\triangleleft\mathcal{G}}(l[\varepsilon])) = \kappa_1((\mathcal{F}_1[-])(\delta_{\mathcal{F}_1\triangleleft\mathcal{G}}(\varepsilon))) \in \mathcal{F}_1 \oplus \mathcal{F}_2(\overline{V})$	(IH)
$(\mathcal{F}_1 \oplus \mathcal{F}_2[-])(\delta_{\mathcal{F}_1\oplus\mathcal{F}_2\triangleleft\mathcal{G}}(r[\varepsilon])) = \kappa_2((\mathcal{F}_2[-])(\delta_{\mathcal{F}_2\triangleleft\mathcal{G}}(\varepsilon))) \in \mathcal{F}_1 \oplus \mathcal{F}_2(\overline{V})$	(IH)
$(\mathcal{F}^A[-])(\delta_{\mathcal{F}^A\triangleleft\mathcal{G}}(a(\varepsilon))) = \left(\lambda a'.\begin{cases} (\mathcal{F}[-])(\delta_{\mathcal{F}\triangleleft\mathcal{G}}(\varepsilon)) & \text{if } a = a' \\ \mathsf{Empty}_{\mathcal{F}\triangleleft\mathcal{G}} & \text{otherwise} \end{cases}\right) \in \mathcal{F}^A(\overline{V})$	(IH and ①)
$(\mathcal{P}_\omega\mathcal{F}[-])(\delta_{\mathcal{P}_\omega\mathcal{F}\triangleleft\mathcal{G}}(\{\varepsilon\})) = \{\, (\mathcal{F}[-])(\delta_{\mathcal{F}\triangleleft\mathcal{G}}(\varepsilon)) \,\} \in \mathcal{P}_\omega\mathcal{F}(\overline{V})$	(IH)

□

3.3.1. *Examples.* n this subsection we will illustrate the construction described in the proof of Theorem 3.14: given an expression $\varepsilon \in \mathsf{Exp}_{\mathcal{G}}$ we construct a $\mathcal{G}$-coalgebra (S, g) such that there is $s \in S$ with $s \sim \varepsilon$. For simplicity, we will consider deterministic and partial automata expressions over $A = \{a, b\}$.

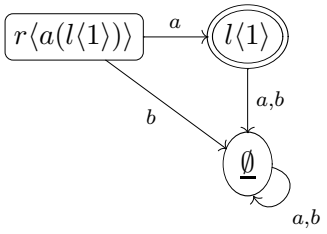
Let us start by showing the synthesised automata for the most simple deterministic expressions – $\underline{\emptyset}$, $l\langle 0\rangle$ and $l\langle 1\rangle$.



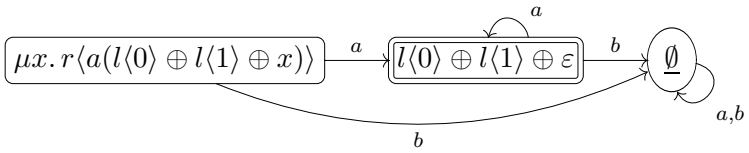
The first two automata recognize the empty language $\emptyset$ and the last the language $\{\epsilon\}$ containing only the empty word.

We note that the generated automata are not minimal (for instance, the automata for $l\langle 0\rangle$ and $\underline{\emptyset}$ are bisimilar). Our goal has been to generate a finite automaton from an expression. From this the minimal automaton can always be obtained by identifying bisimilar states.

The following automaton, generated from the expression $r\langle a(l\langle 1\rangle)\rangle$, recognizes the language {a},

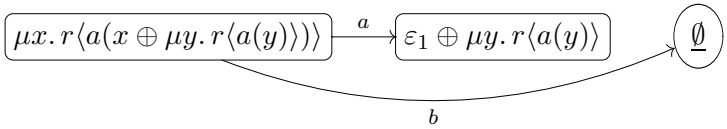


For an example of an expression containing fixed points, consider $\varepsilon = \mu x.\, r\langle a(l\langle 0\rangle \oplus l\langle 1\rangle \oplus x)\rangle$. One can easily compute the synthesised automaton:



and observe that it recognizes the language aa^*. Here, the role of the join-semilattice structure is also visible: $l\langle 0\rangle \oplus l\langle 1\rangle \oplus \varepsilon$ specifies that this state is supposed to be non-final ($l\langle 0\rangle$) and final ($l\langle 1\rangle$). The conflict of these two specifications is solved, when they are combined with $\oplus$, using the semilattice: because $1 \vee 0 = 1$ the state is set to be final.

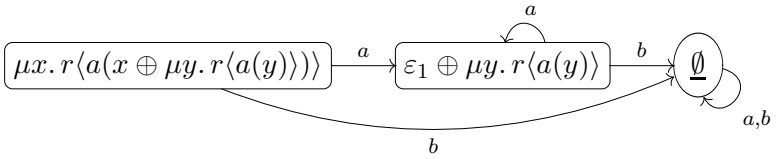
As a last example of deterministic expressions consider $\varepsilon_1 = \mu x.\, r\langle a(x \oplus \mu y.r\langle a(y)\rangle)\rangle$. Applying $\delta_{\mathcal{D}}$ to ε_1 one gets the following (partial) automaton:



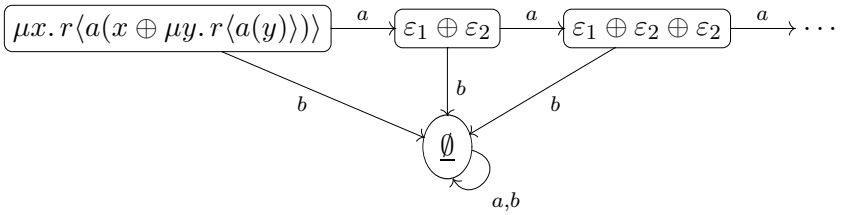
Calculating $\delta_{\mathcal{D}}(\varepsilon_1 \oplus \mu y.\, r\langle a(y)\rangle)$ yields

$$\begin{array}{rcl} \delta_{\mathcal{D}}(\varepsilon_1 \oplus \mu y.\, r\langle a(y)\rangle) & = & \langle 0, t\rangle \\ & \text{where} & t(a) = \varepsilon_1 \oplus \mu y.\, r\langle a(y)\rangle \oplus \mu y.\, r\langle a(y)\rangle\rangle \\ & & t(b) = \underline{\emptyset} \end{array}$$

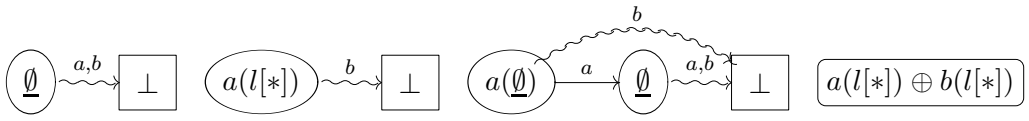
Note that the expression $\varepsilon_1 \oplus \mu y.\, r\langle a(y)\rangle \oplus \mu y.\, r\langle a(y)\rangle$ is in the same equivalence class as $\varepsilon_1 \oplus \mu y.\, r\langle a(y)\rangle$, which is a state that already exists. As we saw in the beginning of Section 3.1, by only applying $\delta_{\mathcal{D}}$, without ACI, one would always generate syntactically different states which instead of the automaton computed now:



would yield the following infinite automaton (with $\varepsilon_2 = \mu y.\, r\langle a(y)\rangle$):

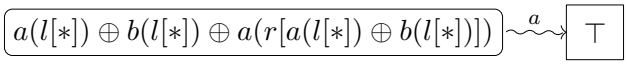


Let us next see a few examples of synthesis for partial automata expressions, where we will illustrate the role of $\bot$ and $\top$. In the graphical representation of a partial automaton (S, p), we will omit transitions for inputs a with $g(s)(a) = \kappa_1(*)$ and we will draw $s \overset{a}{\rightsquigarrow} \boxed{g(s)(a)}$ whenever $g(s)(a) \in \{\bot, \top\}$. Note however that $\bot \notin S$ and $\top \notin S$ and thus will have no defined transitions. As before, let us first present the corresponding automata for simple expressions – $\underline{\emptyset}$, $a(l[*])$, $a(\underline{\emptyset})$ and $a(l[*]) \oplus b(l[*])$.



Note how $\bot$ is used to encode underspecification, working as a kind of deadlock state. In the first three expressions the behaviour for one or both of the inputs is missing, whereas in the last expression the specification is complete.

The element $\top$ is used to deal with inconsistent specifications. For instance, consider the expression $a(l[*]) \oplus b(l[*]) \oplus a(r[a(l[*]) \oplus b(l[*])])$. All inputs are specified, but note that at the outermost level input a appears in two different sub-expressions – $a(l[*])$ and $a(r[a(l[*]) \oplus b(l[*])])$ – specifying at the same time that input a leads to successful termination and that it leads to a state where $a(l[*]) \oplus b(l[*])$ holds, which is contradictory, giving rise to the following automaton.



4. A sound and complete axiomatization

In the previous section, we have shown how to derive from the type of a system, given by a functor $\mathcal{G}$, a language $\mathsf{Exp}_{\mathcal{G}}$ that allows for specification of $\mathcal{G}$-behaviours. Analogously to Kleene's theorem, we have proved the correspondence between the behaviours denoted by $\mathsf{Exp}_{\mathcal{G}}$ and locally finite $\mathcal{G}$-coalgebras. In this section, we will show how to provide $\mathsf{Exp}_{\mathcal{G}}$ with a sound and complete axiomatization. Again, the functor $\mathcal{G}$ will serve as a main guide

for the definition. The defined axiomatization is closely related to Kleene algebra (the set of expressions has a join semilattice structure) and to the axiomatization provided by Milner for CCS (uniqueness of fixed points will be required). When instantiating the definition below to concrete functors one will recover known axiomatizations, such as the one for CCS mentioned above or the one for labelled transition systems (with explicit termination) presented in [1]. The latter will be discussed in detail in Section 5.

Next, we introduce an equational system for expressions of type $\mathcal{F} \lhd \mathcal{G}$. We define the relation $\equiv \subseteq \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}} \times \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}$, written infix, as the least equivalence relation containing the following identities:

(1) $(\mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}, \oplus, \underline{\emptyset})$ is a join-semilattice.

$$\begin{aligned}
&\varepsilon \oplus \varepsilon \equiv \varepsilon && (\textit{Idempotency})\\
&\varepsilon_1 \oplus \varepsilon_2 \equiv \varepsilon_2 \oplus \varepsilon_1 && (\textit{Commutativity})\\
&\varepsilon_1 \oplus (\varepsilon_2 \oplus \varepsilon_3) \equiv (\varepsilon_1 \oplus \varepsilon_2) \oplus \varepsilon_3 && (\textit{Associativity})\\
&\underline{\emptyset} \oplus \varepsilon \equiv \varepsilon && (\textit{Empty})
\end{aligned}$$

(2) μ is the unique fixed point.

$$\begin{aligned}
&\gamma[\mu x.\gamma/x] \equiv \mu x.\gamma && (FP)\\
&\gamma[\varepsilon/x] \equiv \varepsilon \Rightarrow \mu x.\gamma \equiv \varepsilon && (\textit{Unique})
\end{aligned}$$

(3) The join-semilattice structure propagates through the expressions.

$$\begin{array}{lcll|lcll}
\underline{\emptyset} & \equiv & \bot_{\mathsf{B}} & (\mathsf{B}-\underline{\emptyset}) & b_1 \oplus b_2 & \equiv & b_1 \vee_{\mathsf{B}} b_2 & (\mathsf{B}-\oplus)\\
l\langle\underline{\emptyset}\rangle & \equiv & \underline{\emptyset} & (\times-\underline{\emptyset}-L) & l\langle\varepsilon_1 \oplus \varepsilon_2\rangle & \equiv & l\langle\varepsilon_1\rangle \oplus l\langle\varepsilon_2\rangle & (\times-\oplus-L)\\
r\langle\underline{\emptyset}\rangle & \equiv & \underline{\emptyset} & (\times-\underline{\emptyset}-R) & r\langle\varepsilon_1 \oplus \varepsilon_2\rangle & \equiv & r\langle\varepsilon_1\rangle \oplus r\langle\varepsilon_2\rangle & (\times-\oplus-R)\\
a(\underline{\emptyset}) & \equiv & \underline{\emptyset} & (-^A-\underline{\emptyset}) & a(\varepsilon_1 \oplus \varepsilon_2) & \equiv & a(\varepsilon_1) \oplus a(\varepsilon_2) & (-^A-\oplus)\\
 & & & & l[\varepsilon_1 \oplus \varepsilon_2] & \equiv & l[\varepsilon_1] \oplus l[\varepsilon_2] & (+-\oplus-L)\\
 & & & & r[\varepsilon_1 \oplus \varepsilon_2] & \equiv & r[\varepsilon_1] \oplus r[\varepsilon_2] & (+-\oplus-R)\\
 & & & & l[\varepsilon_1] \oplus r[\varepsilon_2] & \equiv & l[\underline{\emptyset}] \oplus r[\underline{\emptyset}] & (+-\oplus-\top)
\end{array}$$

(4) $\equiv$ is a congruence.

$$\varepsilon_1 \equiv \varepsilon_2 \Rightarrow \varepsilon[\varepsilon_1/x] \equiv \varepsilon[\varepsilon_2/x] \quad \text{for } x \text{ free in } \varepsilon \qquad (\textit{Cong})$$

(5) α-equivalence

$$\mu x.\gamma \equiv \mu y.\gamma[y/x] \quad \text{for } y \text{ not free in } \gamma \qquad (\alpha - \textit{equiv})$$

It is important to remark that in the third group of rules there does not exist any rule applicable to expressions of type $\mathcal{P}_\omega\mathcal{F}$.

Example 4.1. Consider the non-deterministic automata over the alphabet $A = \{a\}$:

Applying $\langle\!\langle - \rangle\!\rangle$ (as defined in the proof of Theorem 3.12) one can easily compute the expressions corresponding to s_1 and s_2:

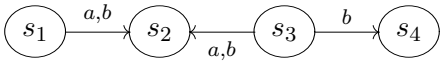
$$\begin{aligned}
\varepsilon_1 &= \langle\!\langle\, s_1 \,\rangle\!\rangle = \mu x_1.l\langle 0\rangle \oplus r\langle a(\{x_1\})\rangle\\
\varepsilon_2 &= \langle\!\langle\, s_2 \,\rangle\!\rangle = \mu y_1.l\langle 0\rangle \oplus r\langle a(\{\mu y_2.l\langle 0\rangle \oplus r\langle a(\{\mu y_1.l\langle 0\rangle \oplus r\langle a(\{y_2\})\rangle\})\rangle\})\rangle
\end{aligned}$$

We prove that $\varepsilon_2 \equiv \varepsilon_1$. In the following calculations let $\varepsilon = \mu x_1.r\langle a(\{x_1\})\rangle$.

$$\begin{array}{lll} & \varepsilon_2 \equiv \varepsilon_1 & \\ \Leftrightarrow & r\langle a(\{\mu y_2.r\langle a(\{r\langle a(\{y_2\})\rangle\})\rangle\})\rangle \equiv \varepsilon & ((\mathsf{B}-\underline{\emptyset}),\ (\times-\underline{\emptyset}-L),\ (FP) \text{ and } (Empty)) \\ \Leftrightarrow & \mu y_2.r\langle a(\{r\langle a(\{y_2\})\rangle\})\rangle \equiv \varepsilon & ((FP) \text{ on } \varepsilon \text{ and } (Cong) \text{ twice}) \\ \Leftarrow & r\langle a(\{r\langle a(\{\varepsilon\})\rangle\})\rangle \equiv \varepsilon & (\text{uniqueness of fixed points}) \\ \Leftrightarrow & r\langle a(\{\varepsilon\})\rangle \equiv \varepsilon & (\text{fixed point axiom}) \\ \Leftrightarrow & \varepsilon \equiv \varepsilon & (\text{fixed point axiom}) \end{array}$$

Note that the $(Cong)$ rule was used in almost every step.

For another example, consider the non-deterministic automaton over the alphabet $A = \{a, b\}$:



Using the definition of $\langle\!\langle - \rangle\!\rangle$ one can compute the following expressions for s_1, s_2, s_3 and s_4:

$$\begin{array}{l} \varepsilon_1 = \langle\!\langle\, s_1 \,\rangle\!\rangle = \mu x_1.l\langle 0\rangle \oplus r\langle a(\{\varepsilon_2\}) \oplus b(\{\varepsilon_2\})\rangle \\ \varepsilon_2 = \langle\!\langle\, s_2 \,\rangle\!\rangle = \mu x_2.l\langle 0\rangle \oplus \underline{\emptyset} \\ \varepsilon_3 = \langle\!\langle\, s_3 \,\rangle\!\rangle = \mu x_3.l\langle 0\rangle \oplus r\langle a(\{\varepsilon_2\}) \oplus b(\{\varepsilon_2\} \oplus \{\varepsilon_4\})\rangle \\ \varepsilon_4 = \langle\!\langle\, s_4 \,\rangle\!\rangle = \mu x_4.l\langle 0\rangle \oplus \underline{\emptyset} \end{array}$$

For ε_2 we calculate:

$$\begin{array}{lll} \varepsilon_2 & \equiv & l\langle 0\rangle \oplus \underline{\emptyset} \quad (FP) \\ & \equiv & l\langle \underline{\emptyset}\rangle \quad (Empty) \text{ and } (\mathsf{B}-\underline{\emptyset}) \\ & \equiv & \underline{\emptyset} \quad (\times-\underline{\emptyset}-L) \end{array}$$

Similarly, one has that $\varepsilon_4 \equiv \underline{\emptyset}$. Next, we prove $\varepsilon_1 \equiv \varepsilon_3$:

$$\begin{array}{lll} & \varepsilon_1 \equiv \varepsilon_3 & \\ \Leftrightarrow & l\langle 0\rangle \oplus r\langle a(\{\varepsilon_2\}) \oplus b(\{\varepsilon_2\})\rangle \equiv l\langle 0\rangle \oplus r\langle a(\{\varepsilon_2\}) \oplus b(\{\varepsilon_2\} \oplus \{\varepsilon_4\})\rangle & (FP) \\ \Leftrightarrow & l\langle 0\rangle \oplus r\langle a(\{\underline{\emptyset}\}) \oplus b(\{\underline{\emptyset}\})\rangle \equiv l\langle 0\rangle \oplus r\langle a(\{\underline{\emptyset}\}) \oplus b(\{\underline{\emptyset}\} \oplus \{\underline{\emptyset}\})\rangle & (\varepsilon_2 \equiv \underline{\emptyset} \equiv \varepsilon_4) \\ \Leftrightarrow & l\langle 0\rangle \oplus r\langle a(\{\underline{\emptyset}\}) \oplus b(\{\underline{\emptyset}\})\rangle \equiv l\langle 0\rangle \oplus r\langle a(\{\underline{\emptyset}\}) \oplus b(\{\underline{\emptyset}\})\rangle & (Idempotency) \end{array}$$

♠

The relation $\equiv$ gives rise to the (surjective) equivalence map $[-]\colon \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}} \to \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}/_{\equiv}$ defined by $[\varepsilon] = \{\varepsilon' \mid \varepsilon \equiv \varepsilon'\}$. The following diagram summarizes the maps we have defined so far:

$$\begin{array}{ccc} \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}} & \xrightarrow{[-]} & \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}/_{\equiv} \\ {\scriptstyle \delta_{\mathcal{F}\lhd\mathcal{G}}}\downarrow & & \\ \mathcal{F}(\mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}) & \xrightarrow[\mathcal{F}([-])]{} & \mathcal{F}(\mathsf{Exp}_{\mathcal{G}}/_{\equiv}) \end{array}$$

In order to complete the diagram, we next prove that $\equiv$ is contained in the kernel of $\mathcal{F}([-]) \circ \delta_{\mathcal{F}\lhd\mathcal{G}}$. This will guarantee the existence of a well-defined function

$$\partial_{\mathcal{F}\lhd\mathcal{G}}\colon \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}/_{\equiv} \to \mathcal{F}(\mathsf{Exp}_{\mathcal{G}}/_{\equiv})$$

which, when $\mathcal{F} = \mathcal{G}$, provides $\mathsf{Exp}_{\mathcal{G}}/_{\equiv}$ with a coalgebraic structure $\partial_{\mathcal{G}}\colon \mathsf{Exp}_{\mathcal{G}}/_{\equiv} \to \mathcal{G}(\mathsf{Exp}_{\mathcal{G}}/_{\equiv})$ (as before, we write $\partial_{\mathcal{G}}$ to abbreviate $\partial_{\mathcal{G}\lhd\mathcal{G}}$) and which makes $[-]$ a homomorphism of coalgebras.

Lemma 4.2. *Let $\mathcal{G}$ and $\mathcal{F}$ be non-deterministic functors, with $\mathcal{F} \lhd \mathcal{G}$. For all $\varepsilon_1, \varepsilon_2 \in \mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}$ with $\varepsilon_1 \equiv \varepsilon_2$,*

$$\mathcal{F}([-]) \circ \delta_{\mathcal{F}\lhd\mathcal{G}}(\varepsilon_1) = \mathcal{F}([-]) \circ \delta_{\mathcal{F}\lhd\mathcal{G}}(\varepsilon_2)$$

Proof. By induction on the length of derivations of $\equiv$.

First, let us consider derivations of length 1. We need to prove the result for all the axioms in items 1. and 3. plus the axioms *FP* and $(\alpha - equiv)$.

For the axioms in 1. the result follows by Lemma 3.13. The axiom *FP* follows trivially because of the definition of $\delta_{\mathcal{G}}$, since $\delta_{\mathcal{G}}(\mu x.\gamma) = \delta_{\mathcal{G}}(\gamma[\mu x.\gamma/x])$ and thus $\mathcal{G}([-]) \circ \delta_{\mathcal{G}}(\mu x.\gamma) = \mathcal{G}([-]) \circ \delta_{\mathcal{G}}(\gamma[\mu x.\gamma/x])$.

For the axiom $(\alpha - equiv)$ we use the $(Cong)$ rule, which is proved below:

$$\begin{array}{ll}
\mathcal{G}([-]) \circ \delta_{\mathcal{G}}(\mu x.\gamma) & \\
= \mathcal{G}([-]) \circ \delta_{\mathcal{G}}(\gamma[\mu x.\gamma/x]) & (\text{def. of } \delta_{\mathcal{G}}) \\
= \mathcal{G}([-]) \circ \delta_{\mathcal{G}}(\gamma[\mu y.\gamma[y/x]/x]) & (\text{by } (Cong)) \\
= \mathcal{G}([-]) \circ \delta_{\mathcal{G}}(\gamma[y/x][\mu y.\gamma[y/x]/y]) & (A[B[y/x]/x] = A[y/x][B[y/x]/y],\ y \text{ not free in } \gamma) \\
= \mathcal{G}([-]) \circ \delta_{\mathcal{G}}(\mu y.\gamma[y/x])) & (\text{def. of } \mathcal{G}([-]) \circ \delta_{\mathcal{G}})
\end{array}$$

Let us next show the proof for some of the axioms in 3.. The omitted cases are similar. We show for each axiom $\varepsilon_1 \equiv \varepsilon_2$ that $\delta_{\mathcal{F}\lhd\mathcal{G}}(\varepsilon_1) = \delta_{\mathcal{F}\lhd\mathcal{G}}(\varepsilon_2)$.

$\boxed{\bot_{\mathsf{B}} \equiv \underline{\emptyset}}$

$\delta_{\mathsf{B}\lhd\mathcal{G}}(\bot_{\mathsf{B}}) = \bot_{\mathsf{B}} = \delta_{\mathsf{B}\lhd\mathcal{G}}(\underline{\emptyset})$

$\boxed{b_1 \oplus b_2 \equiv b_1 \vee_{\mathsf{B}} b_2}$

$\delta_{\mathsf{B}\lhd\mathcal{G}}(b_1 \vee_{\mathsf{B}} b_2) = b_1 \vee_{\mathsf{B}} b_2 = \delta_{\mathsf{B}\lhd\mathcal{G}}(b_1 \oplus b_2)$

$\boxed{l(\underline{\emptyset}) \equiv \underline{\emptyset}}$

$\delta_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}(l(\underline{\emptyset})) = \langle \mathsf{Empty}_{\mathcal{F}_1\lhd\mathcal{G}}, \mathsf{Empty}_{\mathcal{F}_2\lhd\mathcal{G}}\rangle = \delta_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}(\underline{\emptyset})$

$\boxed{l(\varepsilon_1 \oplus \varepsilon_2) \equiv l(\varepsilon_1) \oplus l(\varepsilon_2)}$

$$\begin{array}{l}
\delta_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}(l(\varepsilon_1 \oplus \varepsilon_2)) \\
= \langle \delta_{\mathcal{F}_1\lhd\mathcal{G}}(\varepsilon_1 \oplus \varepsilon_2), \mathsf{Empty}_{\mathcal{F}_2\lhd\mathcal{G}}\rangle) \\
= \langle \mathsf{Plus}_{\mathcal{F}_1\lhd\mathcal{G}}(\delta_{\mathcal{F}_1\lhd\mathcal{G}}(\varepsilon_1), \delta_{\mathcal{F}_1\lhd\mathcal{G}}(\varepsilon_2)), \mathsf{Plus}_{\mathcal{F}_2\lhd\mathcal{G}}(\mathsf{Empty}_{\mathcal{F}_2\lhd\mathcal{G}}, \mathsf{Empty}_{\mathcal{F}_2\lhd\mathcal{G}})\rangle) \\
= \mathsf{Plus}_{\mathcal{F}_1\times\mathcal{F}_2}(\langle \delta_{\mathcal{F}_1\lhd\mathcal{G}}(\varepsilon_1), \mathsf{Empty}_{\mathcal{F}_2\lhd\mathcal{G}}\rangle, \langle \delta_{\mathcal{F}_1\lhd\mathcal{G}}(\varepsilon_2), \mathsf{Empty}_{\mathcal{F}_2\lhd\mathcal{G}}\rangle \\
= \delta_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}(l(\varepsilon_1) \oplus l(\varepsilon_2)))
\end{array}$$

$\boxed{l[\varepsilon_1 \oplus \varepsilon_2] \equiv l[\varepsilon_1] \oplus l[\varepsilon_2]}$

$$\begin{array}{l}
\delta_{\mathcal{F}_1\oplus\mathcal{F}_2\lhd\mathcal{G}}(l[\varepsilon_1 \oplus \varepsilon_2]) \\
= \kappa_1(\delta_{\mathcal{F}_1\lhd\mathcal{G}}(\varepsilon_1 \oplus \varepsilon_2)) \\
= \mathsf{Plus}_{\mathcal{F}_1\oplus\mathcal{F}_2}(\kappa_1(\delta_{\mathcal{F}_1\lhd\mathcal{G}}(\varepsilon_1)), \kappa_1(\delta_{\mathcal{F}_1\lhd\mathcal{G}}(\varepsilon_2)) \\
= \delta_{\mathcal{F}_1\oplus\mathcal{F}_2\lhd\mathcal{G}}(l[\varepsilon_1] \oplus l[\varepsilon_2])
\end{array}$$

$\boxed{l[\varepsilon_1] \oplus r[\varepsilon_2] \equiv l[\underline{\emptyset}] \oplus r[\underline{\emptyset}]}$

$$\begin{array}{l}
\delta_{\mathcal{F}_1\oplus\mathcal{F}_2\lhd\mathcal{G}}(l[\varepsilon_1] \oplus r[\varepsilon_2]) \\
= \mathsf{Plus}_{\mathcal{F}_1\oplus\mathcal{F}_2}(\kappa_1(\delta_{\mathcal{F}_1\lhd\mathcal{G}}(\varepsilon_1)), \kappa_2(\delta_{\mathcal{F}_2\lhd\mathcal{G}}(\varepsilon_2))) \\
= \top \\
= \mathsf{Plus}_{\mathcal{F}_1\oplus\mathcal{F}_2}(\kappa_1(\delta_{\mathcal{F}_1\lhd\mathcal{G}}(\underline{\emptyset})), \kappa_2(\delta_{\mathcal{F}_2\lhd\mathcal{G}}(\underline{\emptyset}))) \\
= \delta_{\mathcal{F}_1\oplus\mathcal{F}_2\lhd\mathcal{G}}(l[\underline{\emptyset}] \oplus r[\underline{\emptyset}])
\end{array}$$

Note that if we would have the axioms $l[\underline{\emptyset}] \equiv \underline{\emptyset}$ and $r[\underline{\emptyset}] \equiv \underline{\emptyset}$ in the axiomatization presented above, this theorem would not hold.

$$\delta_{\mathcal{F}_1 \oplus \mathcal{F}_2 \lhd \mathcal{G}}(l[\underline{\emptyset}]) = \kappa_1([\bot]) \neq \bot = \delta_{\mathcal{F}_1 \oplus \mathcal{F}_2 \lhd \mathcal{G}}(\underline{\emptyset})$$
$$\delta_{\mathcal{F}_1 \oplus \mathcal{F}_2 \lhd \mathcal{G}}(r[\underline{\emptyset}]) = \kappa_2([\bot]) \neq \bot = \delta_{\mathcal{F}_1 \oplus \mathcal{F}_2 \lhd \mathcal{G}}(\underline{\emptyset})$$

Derivations with length $k > 1$ can be obtained by two rules: (*Unique*) or (*Cong*). For the first (which uses the second), suppose that we have derived $\mu x.\gamma \equiv \varepsilon$ and that we have already proved $\gamma[\varepsilon/x] \equiv \varepsilon$. Then, we have:

$$\begin{array}{rcll} \mathcal{G}([-]) \circ \delta_{\mathcal{G}}(\mu x.\gamma) & = & \mathcal{G}([-]) \circ \delta_{\mathcal{G}}(\gamma[\mu x.\gamma/x]) & (\text{def. } \delta_{\mathcal{G}}) \\ & = & \mathcal{G}([-]) \circ \delta_{\mathcal{G}}(\gamma[\varepsilon/x]) & (\text{by } (\mathit{Cong})) \\ & = & \mathcal{G}([-]) \circ \delta_{\mathcal{G}}(\varepsilon) & (\text{induction hypothesis}) \end{array}$$

For (*Cong*), suppose that we have derived $\varepsilon[\varepsilon_1/x] \equiv \varepsilon[\varepsilon_2/x_2]$ and that we have already derived $\varepsilon_1 \equiv \varepsilon_2$, which gives us, as induction hypothesis, the equality

$$(\mathcal{F}[-])(\delta_{\mathcal{F} \lhd \mathcal{G}}(\varepsilon_1)) = (\mathcal{F}[-])(\delta_{\mathcal{F} \lhd \mathcal{G}}(\varepsilon_2)) \tag{4.1}$$

This equation is precisely what we need to prove the case $\varepsilon = x$ (and thus $\varepsilon_1, \varepsilon_2 : \mathcal{G} \lhd \mathcal{G}$):

$$\begin{array}{rcll} (\mathcal{G}[-])(\delta_{\mathcal{G}}(x[\varepsilon_1/x]) & = & (\mathcal{G}[-])(\delta_{\mathcal{G}}(\varepsilon_1)) & \\ & = & (\mathcal{G}[-])(\delta_{\mathcal{G}}(\varepsilon_2)) & (4.1) \\ & = & (\mathcal{G}[-])(\delta_{\mathcal{G}}(x[\varepsilon_2/x])) & \end{array}$$

For the cases $\varepsilon \neq x$, we prove that $\delta_{\mathcal{F} \lhd \mathcal{G}}(\varepsilon[\varepsilon_1/x]) = \delta_{\mathcal{F} \lhd \mathcal{G}}(\varepsilon[\varepsilon_2/x])$, by induction on the product of types of expressions and expressions (using the order defined in equation (3.1)). We show a few cases, the omitted ones are similar.

$$\begin{array}{rcl} \delta_{\mathcal{G} \lhd \mathcal{G}}((\mu y.\varepsilon)[\varepsilon_1/x]) & = & \delta_{\mathcal{G} \lhd \mathcal{G}}(\varepsilon[\varepsilon_1/x][\mu y.\varepsilon/y])) \\ & \overset{(IH)}{=} & \delta_{\mathcal{G} \lhd \mathcal{G}}(\varepsilon[\varepsilon_2/x][\mu y.\varepsilon/y])) = \delta_{\mathcal{G} \lhd \mathcal{G}}((\mu y.\varepsilon)[\varepsilon_2/x]) \\ \delta_{\mathcal{F}_1 \times \mathcal{F}_2 \lhd \mathcal{G}}(l\langle\varepsilon\rangle[\varepsilon_1/x]) & = & \langle \delta_{\mathcal{F}_1 \lhd \mathcal{G}}(\varepsilon[\varepsilon_1/x]), \mathsf{Empty}_{\mathcal{F}_2 \lhd \mathcal{G}} \rangle \\ & \overset{(IH)}{=} & \langle \delta_{\mathcal{F}_1 \lhd \mathcal{G}}(\varepsilon[\varepsilon_2/x]), \mathsf{Empty}_{\mathcal{F}_2 \lhd \mathcal{G}} \rangle = \delta_{\mathcal{F}_1 \times \mathcal{F}_2 \lhd \mathcal{G}}(l\langle\varepsilon\rangle[\varepsilon_2/x]) \\ \delta_{\mathcal{F}_1 \oplus \mathcal{F}_2 \lhd \mathcal{G}}(l[\varepsilon][\varepsilon_1/x]) & = & \kappa_1(\delta_{\mathcal{F}_1 \lhd \mathcal{G}}(\varepsilon[\varepsilon_1/x])) \\ & \overset{(IH)}{=} & \kappa_1(\delta_{\mathcal{F}_1 \lhd \mathcal{G}}(\varepsilon[\varepsilon_2/x])) = \delta_{\mathcal{F}_1 \oplus \mathcal{F}_2 \lhd \mathcal{G}}(l[\varepsilon][\varepsilon_2/x]) \end{array}$$

□

Thus, we have a well-defined function $\partial_{\mathcal{F} \lhd \mathcal{G}} \colon \mathsf{Exp}_{\mathcal{F} \lhd \mathcal{G}}/_{\equiv} \to \mathcal{F}(\mathsf{Exp}_{\mathcal{G}}/_{\equiv})$, which makes the diagram above commute, that is $\partial_{\mathcal{F} \lhd \mathcal{G}}([\varepsilon]) = (\mathcal{F}[-]) \circ \delta_{\mathcal{F} \lhd \mathcal{G}}(\varepsilon)$. This provides the set $\mathsf{Exp}_{\mathcal{G}}/_{\equiv}$ with a coalgebraic structure $\partial_{\mathcal{G}} \colon \mathsf{Exp}_{\mathcal{G}}/_{\equiv} \to \mathcal{G}(\mathsf{Exp}_{\mathcal{G}}/_{\equiv})$ which makes $[-]$ a homomorphism between the coalgebras $(\mathsf{Exp}_{\mathcal{G}}, \delta_{\mathcal{G}})$ and $(\mathsf{Exp}_{\mathcal{G}}/_{\equiv}, \partial_{\mathcal{G}})$.

4.1. **Soundness and Completeness.** Next we show that the axiomatization introduced in the previous section is sound and complete.

Soundness is a direct consequence of the fact that the equivalence map $[-]$ is a coalgebra homomorphism.

Theorem 4.3 (Soundness)**.** *Let $\mathcal{G}$ be a non-deterministic functor. For all $\varepsilon_1, \varepsilon_2 \in \mathsf{Exp}_{\mathcal{G}}$,*

$$\varepsilon_1 \equiv \varepsilon_2 \Rightarrow \varepsilon_1 \sim \varepsilon_2$$

Proof. Let $\mathcal{G}$ be a non-deterministic functor, let $\varepsilon_1, \varepsilon_2 \in \mathsf{Exp}_{\mathcal{G}}$ and suppose that $\varepsilon_1 \equiv \varepsilon_2$. Then, $[\varepsilon_1] = [\varepsilon_2]$ and, thus

$$\mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}/_{\equiv}}([\varepsilon_1]) = \mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}/_{\equiv}}([\varepsilon_2])$$

where $\mathbf{beh}_S$ denotes, for any $\mathcal{G}$-coalgebra (S, g), the unique map into the final coalgebra. The uniqueness of the map into the final coalgebra and the fact that $[-]$ is a coalgebra homomorphism implies that $\mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}/_{\equiv}} \circ [-] = \mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}}$ which then yields

$$\mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}}(\varepsilon_1) = \mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}}(\varepsilon_2)$$

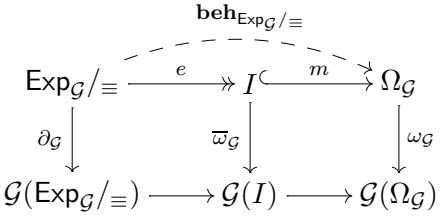
Since in the final coalgebra only the bisimilar elements are identified, $\varepsilon_1 \sim \varepsilon_2$ follows. □

For completeness a bit more of work is required. Let us explain upfront the key steps of the proof. The goal is to prove that $\varepsilon_1 \sim \varepsilon_2 \Rightarrow \varepsilon_1 \equiv \varepsilon_2$. First, note that we have

$$\varepsilon_1 \sim \varepsilon_2 \Leftrightarrow \mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}}(\varepsilon_1) = \mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}}(\varepsilon_2) \Leftrightarrow \mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}/_{\equiv}}([\varepsilon_1]) = \mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}/_{\equiv}}([\varepsilon_2]) \tag{4.2}$$

We then prove that $\mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}/_{\equiv}}$ is injective, which is a sufficient condition to guarantee that $\varepsilon_1 \equiv \varepsilon_2$ (since it implies, together with (4.2), that $[\varepsilon_1] = [\varepsilon_2]$).

We proceed as follows. First, we factorize the map $\mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}/_{\equiv}}$ into an epimorphism followed by a monomorphism [31, Theorem 7.1] as shown in the following diagram (where $I = \mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}/_{\equiv}}(\mathsf{Exp}_{\mathcal{G}}/_{\equiv})$):



Then, we prove that (1) $(\mathsf{Exp}_{\mathcal{G}}/_{\equiv}, \partial_{\mathcal{G}})$ is a locally finite coalgebra (Lemma 4.4) and (2) both coalgebras $(\mathsf{Exp}_{\mathcal{G}}/_{\equiv}, \partial_{\mathcal{G}})$ and $(I, \overline{\omega}_{\mathcal{G}})$ are final in the category of locally finite $\mathcal{G}$-coalgebras (Lemmas 4.7 and 4.8, respectively). Since final coalgebras are unique up to isomorphism, it follows that $e\colon \mathsf{Exp}_{\mathcal{G}}/_{\equiv} \to I$ is in fact an isomorphism and therefore $\mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}/_{\equiv}}$ is injective, which will give us completeness.

In the case of the deterministic automata functor $\mathcal{D} = 2 \times \mathsf{Id}^A$, the set I will be precisely the set of regular languages, the class of languages that can be denoted by regular expressions. This means that final locally finite coalgebras generalize regular languages (in the same way that final coalgebras generalize the set of all languages).

We proceed with presenting and proving the extra lemmas needed in order to prove completeness. We start by showing that the coalgebra $(\mathsf{Exp}_{\mathcal{G}}/_{\equiv}, \partial_{\mathcal{G}})$ is locally finite (note that this implies that $(I, \overline{\omega}_{\mathcal{G}})$ is also locally finite) and that $\partial_{\mathcal{G}}$ is an isomorphism.

Lemma 4.4. *The coalgebra $(\mathsf{Exp}_{\mathcal{G}}/_{\equiv}, \partial_{\mathcal{G}})$ is a locally finite coalgebra. Moreover, $\partial_{\mathcal{G}}$ is an isomorphism.*

Proof. Local finiteness is a direct consequence of the generalized Kleene's theorem (Theorem 3.14). In the proof of Theorem 3.14 we showed that, given $\varepsilon \in \mathsf{Exp}_{\mathcal{G}}$, the subcoalgebra $\langle [\varepsilon]_{ACIE} \rangle$ is finite. Thus, the subcoalgebra $\langle [\varepsilon] \rangle$ is also finite (since $\mathsf{Exp}_{\mathcal{G}}/_{\equiv}$ is a quotient of $\mathsf{Exp}_{\mathcal{G}}/_{\equiv_{ACIE}}$).

To see that $\partial_{\mathcal{G}}$ is an isomorphism, first define, for every $\mathcal{F} \lhd \mathcal{G}$,

$$\partial^{-1}_{\mathcal{F}\lhd\mathcal{G}}(c) = [\overline{\gamma}^{\mathcal{F}}_c] \tag{4.3}$$

where $\overline{\gamma}^{\mathcal{F}}_c$ is defined, for $\mathcal{F} \neq \mathsf{Id}$, as $\gamma^{\mathcal{F}}_c$ in the proof of Theorem 3.12, and for $\mathcal{F} = \mathsf{Id}$ as $\overline{\gamma}^{\mathsf{Id}}_{[\varepsilon]} = \varepsilon$. Then, we prove that $\partial^{-1}_{\mathcal{F}\lhd\mathcal{G}}$ has indeed the properties ① $\partial^{-1}_{\mathcal{F}\lhd\mathcal{G}} \circ \partial_{\mathcal{F}\lhd\mathcal{G}} = id_{\mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}/\equiv}$ and ② $\partial_{\mathcal{F}\lhd\mathcal{G}} \circ \partial^{-1}_{\mathcal{F}\lhd\mathcal{G}} = id_{\mathcal{F}(\mathsf{Exp}_{\mathcal{F}\lhd\mathcal{G}}/\equiv)}$. Instantiating $\mathcal{F} = \mathcal{G}$ one derives that $\delta_{\mathcal{G}}$ is an isomorphism. It is enough to prove for ① that $\overline{\gamma}^{\mathcal{F}}_{\partial_{\mathcal{F}\lhd\mathcal{G}}([\varepsilon])} \equiv \varepsilon$ and for ② that $\partial_{\mathcal{F}\lhd\mathcal{G}}([\overline{\gamma}^{\mathcal{F}}_c]) = c$. We illustrate a few cases. The omitted ones are similar.

① By induction on the product of types of expressions and expressions (using the order defined in equation (3.1)).

$$\overline{\gamma}^{\mathsf{Id}}_{\partial_{\mathsf{Id}\lhd\mathcal{G}}([\varepsilon])} = \varepsilon$$

$$\overline{\gamma}^{\mathcal{F}_1\times\mathcal{F}_2}_{\partial_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}([r\langle\varepsilon\rangle])} = l\langle\overline{\gamma}^{\mathcal{F}_1}_{\partial_{\mathcal{F}_1\lhd\mathcal{G}}(\underline{\emptyset})}\rangle \oplus r\langle\overline{\gamma}^{\mathcal{F}_2}_{\partial_{\mathcal{F}_2\lhd\mathcal{G}}(\varepsilon)}\rangle \overset{(IH)}{\equiv} l\langle\underline{\emptyset}\rangle \oplus r\langle\varepsilon\rangle \equiv r\langle\varepsilon\rangle$$

$$\overline{\gamma}^{\mathcal{G}}_{\partial_{\mathcal{G}}([\mu x.\varepsilon])} = \overline{\gamma}^{\mathcal{G}}_{\partial_{\mathcal{G}}([\varepsilon[\mu x.\varepsilon/x]])} \overset{(IH)}{\equiv} \varepsilon[\mu x.\varepsilon/x] \equiv \mu x.\varepsilon$$

Note that the cases $\varepsilon = \underline{\emptyset}$ and $\varepsilon = \varepsilon_1 \oplus \varepsilon_2$ require an extra proof (by induction on $\mathcal{F}$). More precisely, one needs to prove that

$$\text{ⓐ}\ \ \overline{\gamma}^{\mathcal{F}}_{\mathcal{F}[-](\mathsf{Empty}_{\mathcal{F}\lhd\mathcal{G}})} \equiv \underline{\emptyset} \text{ and ⓑ } \ \overline{\gamma}^{\mathcal{F}}_{\mathcal{F}[-](\mathsf{Plus}_{\mathcal{F}\lhd\mathcal{G}}(x_1,x_2))} \equiv \overline{\gamma}^{\mathcal{F}}_{\mathcal{F}[-](x_1)} \oplus \overline{\gamma}^{\mathcal{F}}_{\mathcal{F}[-](x_2)}$$

It is an easy proof by induction. We illustrate here only the cases $\mathcal{F} = \mathsf{Id}$, $\mathcal{F} = \mathsf{B}$ and $\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2$.

ⓐ
$$\overline{\gamma}^{\mathsf{Id}}_{[\underline{\emptyset}]} = \underline{\emptyset}$$
$$\overline{\gamma}^{\mathsf{B}}_{[\bot_{\mathsf{B}}]} = \bot_{\mathsf{B}} \equiv \underline{\emptyset}$$
$$\begin{aligned}\overline{\gamma}^{\mathcal{F}_1\times\mathcal{F}_2}_{\langle\mathcal{F}_1[-](\mathsf{Empty}_{\mathcal{F}_1\lhd\mathcal{G}}),\mathcal{F}_2[-](\mathsf{Empty}_{\mathcal{F}_2\lhd\mathcal{G}})\rangle} &= l\langle\overline{\gamma}^{\mathcal{F}_1}_{\mathcal{F}_1[-](\mathsf{Empty}_{\mathcal{F}_1\lhd\mathcal{G}})}\rangle \oplus r\langle\overline{\gamma}^{\mathcal{F}_2}_{\mathcal{F}_2[-](\mathsf{Empty}_{\mathcal{F}_2\lhd\mathcal{G}})}\rangle \\ &\overset{(IH)}{\equiv} l\langle\underline{\emptyset}\rangle \oplus r\langle\underline{\emptyset}\rangle \equiv \underline{\emptyset}\end{aligned}$$

ⓑ
$$\overline{\gamma}^{\mathsf{Id}}_{[x_1\oplus x_2]} = x_1 \oplus x_2 = \overline{\gamma}^{\mathsf{Id}}_{[x_1]} \oplus \overline{\gamma}^{\mathsf{Id}}_{[x_2]}$$
$$\begin{aligned}\overline{\gamma}^{\mathsf{B}}_{[x_1\vee_{\mathsf{B}}x_2]} = x_1 \vee_{\mathsf{B}} x_2 \equiv x_1 \oplus x_2 &= \overline{\gamma}^{\mathsf{B}}_{[x_1]} \oplus \overline{\gamma}^{\mathsf{B}}_{[x_2]} \\ \overline{\gamma}^{\mathcal{F}_1\times\mathcal{F}_2}_{\mathcal{F}_1\times\mathcal{F}_2[-](\mathsf{Plus}_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}(\langle u_1,v_1\rangle,\langle u_2,v_2\rangle))} &= \overline{\gamma}^{\mathcal{F}_1\times\mathcal{F}_2}_{\langle\mathsf{Plus}_{\mathcal{F}_1}(u_1,v_1),\mathsf{Plus}_{\mathcal{F}_2}(u_2,v_2)\rangle} \\ &= l\langle\overline{\gamma}^{\mathcal{F}_1}_{\mathsf{Plus}_{\mathcal{F}_1}(u_1,v_1)}\rangle \oplus r\langle\overline{\gamma}^{\mathcal{F}_2}_{\mathsf{Plus}_{\mathcal{F}_2}(u_2,v_2)}\rangle \\ &\overset{(IH)}{\equiv} l\langle\gamma^{\mathcal{F}_1}_{u_1} \oplus \gamma^{\mathcal{F}_1}_{v_1}\rangle \oplus r\langle\gamma^{\mathcal{F}_2}_{u_2} \oplus \gamma^{\mathcal{F}_2}_{v_2}\rangle \\ &\equiv (l\langle\gamma^{\mathcal{F}_1}_{u_1}\rangle \oplus r\langle\gamma^{\mathcal{F}_2}_{u_2}\rangle) \oplus (l\langle\gamma^{\mathcal{F}_1}_{v_1}\rangle \oplus r\langle\gamma^{\mathcal{F}_2}_{v_2}\rangle) \\ &= \overline{\gamma}^{\mathcal{F}_1\times\mathcal{F}_2}_{\langle u_1,u_2\rangle} \oplus \overline{\gamma}^{\mathcal{F}_1\times\mathcal{F}_2}_{\langle v_1,v_2\rangle}\end{aligned}$$

② By induction on the structure of $\mathcal{F}$.

$$\begin{aligned}\partial_{\mathcal{F}_1 \oplus\!\!\!\oplus \mathcal{F}_2 \triangleleft \mathcal{G}}([\overline{\gamma}_c^{\mathcal{F}_1 \oplus\!\!\!\oplus \mathcal{F}_2}]) \quad &= \quad \begin{cases} \partial_{\mathcal{F}_1 \oplus\!\!\!\oplus \mathcal{F}_2 \triangleleft \mathcal{G}}([l[\overline{\gamma}_{c'}^{\mathcal{F}_1}]]) = \kappa_1(\partial_{\mathcal{F}_1 \triangleleft \mathcal{G}}([\overline{\gamma}_{c'}^{\mathcal{F}_1}])) & c = \kappa_1(c') \\ \partial_{\mathcal{F}_1 \oplus\!\!\!\oplus \mathcal{F}_2 \triangleleft \mathcal{G}}([r[\overline{\gamma}_{c'}^{\mathcal{F}_2}]]) = \kappa_2(\partial_{\mathcal{F}_2 \triangleleft \mathcal{G}}([\overline{\gamma}_{c'}^{\mathcal{F}_2}])) & c = \kappa_2(c') \\ \partial_{\mathcal{F}_1 \oplus\!\!\!\oplus \mathcal{F}_2 \triangleleft \mathcal{G}}([\underline{\emptyset}]) = \bot & c = \bot \\ \partial_{\mathcal{F}_1 \oplus\!\!\!\oplus \mathcal{F}_2 \triangleleft \mathcal{G}}([l[\underline{\emptyset}] \oplus r[\underline{\emptyset}]]) = \top & c = \top \end{cases} \\ &\overset{(IH)}{=} \quad c \end{aligned}$$

$$\begin{aligned}\partial_{\mathcal{R}\mathcal{F} \triangleleft \mathcal{G}}([\overline{\gamma}_C^{\mathcal{R}\mathcal{F}}]) \quad &= \quad \begin{cases} \partial_{\mathcal{R}\mathcal{F} \triangleleft \mathcal{G}}([\underline{\emptyset}]) = \emptyset & C = \emptyset \\ \partial_{\mathcal{R}\mathcal{F} \triangleleft \mathcal{G}}([\bigoplus_{c \in C} \overline{\gamma}_c^{\mathcal{F}_1}]) = \{\partial_{\mathcal{F} \triangleleft \mathcal{G}}([\overline{\gamma}_c^{\mathcal{F}_1}]) \mid c \in C\} & \text{otherwise} \end{cases} \\ &\overset{(IH)}{=} \quad C \end{aligned}$$

□

Next, we prove the analogue of the following useful and intuitive equality on regular expressions. Given a deterministic automaton $\langle o, t \rangle \colon S \to 2 \times S^A$ and a state $s \in S$, the associated regular expression r_s can be written as

$$r_s = o(s) + \sum_{a \in A} a \cdot r_{t(s)(a)} \tag{4.4}$$

using the axioms of Kleene algebra [13, Theorem 4.4].

Lemma 4.5. *Let (S, g) be a locally finite $\mathcal{G}$-coalgebra, with $\mathcal{G} \neq \mathsf{Id}$, and let $s \in S$, with $\langle s \rangle = \{s_1, \ldots, s_n\}$ (where $s_1 = s$). Then:*

$$\langle\!\langle\, s_i \,\rangle\!\rangle \quad \equiv \quad \gamma^{\mathcal{G}}_{g(s_i)}\{\langle\!\langle\, s_1 \,\rangle\!\rangle / x_1\} \ldots \{\langle\!\langle\, s_n \,\rangle\!\rangle / x_n\} \tag{4.5}$$

Proof. Let A_i^k, where i and k range from 1 to n, be the terms defined as in the proof of Theorem 3.12. Recall that $\langle\!\langle\, s_i \,\rangle\!\rangle = A_i^n$. We calculate:

$$\begin{aligned} & \langle\!\langle\, s_i \,\rangle\!\rangle \\ &= A_i^n \\ &= (\mu x_i . \gamma^{\mathcal{G}}_{g(s_i)})\{A_1^0/x_1\} \ldots \{A_n^{n-1}/x_n\} \\ &= \mu x_i . (\gamma^{\mathcal{G}}_{g(s_i)}\{A_1^0/x_1\} \ldots \{A_{i-1}^{i-2}/x_{i-2}\}\{A_{i+1}^{i}/x_{i+1}\} \ldots \{A_n^{n-1}/x_n\}) \\ &\equiv \gamma^{\mathcal{G}}_{g(s_i)}\{A_1^0/x_1\} \ldots \{A_{i-1}^{i-2}/x_{i-2}\}\{A_{i+1}^{i}/x_{i+1}\} \ldots \{A_n^{n-1}/x_n\}\{A_i^n/x_i\} && \text{(fixed point axiom}^3\text{)} \\ &= \gamma^{\mathcal{G}}_{g(s_i)}\{A_1^0/x_1\} \ldots \{A_n^{n-1}/x_n\} && \text{(by 3.2)} \\ &= \gamma^{\mathcal{G}}_{g(s_i)}\{A_1^0\{A_2^1/x_2\} \ldots \{A_n^{n-1}/x_n\}/x_1\} \ldots \{A_n^{n-1}/x_n\} && \text{(by 3.3)} \\ &= \gamma^{\mathcal{G}}_{g(s_i)}\{A_1^n/x_1\}\{A_2^1/x_2\} \ldots \{A_n^{n-1}/x_n\} && \text{(def. } A_1^n\text{)} \\ &\vdots \quad \text{(repeat last 2 steps for } A_2^1, \ldots, A_{n-1}^{n-2}\text{)} \\ &= \gamma^{\mathcal{G}}_{g(s_i)}\{A_1^n/x_1\}\{A_2^n/x_2\} \ldots \{A_n^n/x_n\} && (A_{n-1}^n = A_n^n) \end{aligned}$$

□

[3]Note that the fixed point axiom can be formulated using syntactic replacement rather than substitution – $\gamma\{\mu x.\gamma/x\} \equiv \mu x.\gamma$ – since $\mu x.\gamma$ is a closed term.

Instantiating (4.5) for $\langle o, t\rangle \colon S \to 2 \times S^A$, one can easily spot the similarity with equation (4.4) above:

$$\langle\!\langle\, s \,\rangle\!\rangle \equiv l\langle o(s)\rangle \oplus r\Big\langle \bigoplus_{a\in A} a\big(\langle\!\langle\, t(s)(a) \,\rangle\!\rangle\big)\Big\rangle$$

Next, we prove that there exists a coalgebra homomorphism between any locally finite $\mathcal{G}$- coalgebra (S, g) and $(\mathsf{Exp}_{\mathcal{G}}/_{\equiv}, \partial_{\mathcal{G}})$.

Lemma 4.6. *Let (S, g) be a locally finite $\mathcal{G}$-coalgebra. There exists a coalgebra homomorphism $\lceil - \rceil \colon S \to \mathsf{Exp}_{\mathcal{G}}/_{\equiv}$.*

Proof. We define $\lceil - \rceil = [-] \circ \langle\!\langle - \rangle\!\rangle$, where $\langle\!\langle - \rangle\!\rangle$ is as in the proof of Theorem 3.12, associating to a state s of a locally finite coalgebra an expression $\langle\!\langle\, s \,\rangle\!\rangle$ with $s \sim \langle\!\langle\, s \,\rangle\!\rangle$. To prove that $\lceil - \rceil$ is a homomorphism we need to verify that $(\mathcal{G}\lceil - \rceil) \circ g = \partial_{\mathcal{G}} \circ \lceil - \rceil$.

If $\mathcal{G} = \mathsf{Id}$, then $(\mathcal{G}\lceil - \rceil) \circ g(s_i) = [\underline{\emptyset}] = \partial_{\mathcal{G}}(\lceil s_i \rceil)$. For $\mathcal{G} \neq \mathsf{Id}$ we calculate, using Lemma 4.5:

$$\partial_{\mathcal{G}}(\lceil s_i \rceil) = \partial_{\mathcal{G}}([\gamma^{\mathcal{G}}_{g(s_i)}[\langle\!\langle\, s_1 \,\rangle\!\rangle/x_1] \dots [\langle\!\langle\, s_n \,\rangle\!\rangle/x_n]])$$

and we then prove the more general equality, for $\mathcal{F} \lhd \mathcal{G}$ and $c \in \mathcal{F}\langle s\rangle$:

$$\partial_{\mathcal{F}\lhd\mathcal{G}}([\gamma^{\mathcal{F}}_{c}[\langle\!\langle\, s_1 \,\rangle\!\rangle/x_1] \dots [\langle\!\langle\, s_n \,\rangle\!\rangle/x_n]]) = \mathcal{F}\lceil - \rceil(c) \tag{4.6}$$

The intended equality then follows by taking $\mathcal{F} = \mathcal{G}$ and $c = g(s_i)$. Let us prove the equation (4.6) by induction on $\mathcal{F}$.

$\boxed{\mathcal{F} = \mathsf{Id}}$ $c = s_j \in \langle s\rangle$

$$\partial_{\mathsf{Id}\lhd\mathcal{G}}([\gamma^{\mathsf{Id}}_{s_j}[\langle\!\langle\, s_1 \,\rangle\!\rangle/x_1] \dots [\langle\!\langle\, s_n \,\rangle\!\rangle/x_n]]) = [\langle\!\langle\, s_j \,\rangle\!\rangle] = \lceil s_j \rceil$$

$\boxed{\mathcal{F} = \mathsf{B}}$ $c = b \in \mathsf{B}$

$$\partial_{\mathsf{B}\lhd\mathcal{G}}([\gamma^{\mathsf{B}}_{b}[\langle\!\langle\, s_1 \,\rangle\!\rangle/x_1] \dots [\langle\!\langle\, s_n \,\rangle\!\rangle/x_n]]) = [b] = \mathsf{B}\lceil - \rceil(b)$$

$\boxed{\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2}$ $c = \langle c_1, c_2\rangle \in (\mathcal{F}_1 \times \mathcal{F}_2)\langle s\rangle$

$$\begin{aligned}
& \partial_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}([\gamma^{\mathcal{F}_1\times\mathcal{F}_2}_{\langle c_1,c_2\rangle}[\langle\!\langle\, s_1 \,\rangle\!\rangle/x_1] \dots [\langle\!\langle\, s_n \,\rangle\!\rangle/x_n]]) \\
= \;& \partial_{\mathcal{F}_1\times\mathcal{F}_2\lhd\mathcal{G}}([l(\gamma^{\mathcal{F}_1}_{c_1}) \oplus r(\gamma^{\mathcal{F}_2}_{c_2})[\langle\!\langle\, s_1 \,\rangle\!\rangle/x_1] \dots [\langle\!\langle\, s_n \,\rangle\!\rangle/x_n]]) \\
= \;& \langle \partial_{\mathcal{F}_1\lhd\mathcal{G}}([\gamma^{\mathcal{F}_1}_{c_1}[\langle\!\langle\, s_1 \,\rangle\!\rangle/x_1] \dots [\langle\!\langle\, s_n \,\rangle\!\rangle/x_n]]), \partial_{\mathcal{F}_2\lhd\mathcal{G}}([\gamma^{\mathcal{F}_2}_{c_2}[\langle\!\langle\, s_1 \,\rangle\!\rangle/x_1] \dots [\langle\!\langle\, s_n \,\rangle\!\rangle/x_n]])\rangle \\
\overset{(IH)}{=} \;& \langle \mathcal{F}_1\lceil - \rceil(c_1), \mathcal{F}_2\lceil - \rceil(c_2)\rangle \\
= \;& (\mathcal{F}_1 \times \mathcal{F}_2\lceil - \rceil)(c)
\end{aligned}$$

$\boxed{\mathcal{F} = \mathcal{F}_1 \oplus \mathcal{F}_2}$, $\boxed{\mathcal{F} = \mathcal{F}_1^A}$ and $\boxed{\mathcal{F} = \mathcal{P}_{\!\mathcal{U}}\mathcal{F}_1}$: similar to $\mathcal{F}_1 \times \mathcal{F}_2$. □

We can now prove that the coalgebras $(\mathsf{Exp}_{\mathcal{G}}/_{\equiv}, \partial_{\mathcal{G}})$ and $(I, \overline{\omega}_{\mathcal{G}})$ are both final in the category of locally finite $\mathcal{G}$-coalgebras.

Lemma 4.7. *The coalgebra $(I, \overline{\omega}_{\mathcal{G}})$ is final in the category $Coalg(\mathcal{G})_{\mathbf{LF}}$.*

Proof. We want to show that for any locally finite $\mathcal{G}$-coalgebra (S, g), there exists a *unique* homomorphism $(S, g) \to (I, \overline{\omega}_{\mathcal{G}})$. The existence is guaranteed by Lemma 4.6, where the homomorphism $\lceil - \rceil \colon S \to \mathsf{Exp}_{\mathcal{G}}/_{\equiv}$ is defined. Post-composing this homomorphism with e (defined above) we get a coalgebra homomorphism $e \circ \lceil - \rceil \colon S \to I$. If there is another homomorphism $f \colon S \to I$, then by post-composition with the inclusion $m \colon I \hookrightarrow \Omega$ we get two homomorphisms ($m \circ f$ and $m \circ e \circ \lceil - \rceil$) into the final $\mathcal{G}$-coalgebra. Thus, f and $e \circ \lceil - \rceil$ must be equal. □

Lemma 4.8. *The coalgebra* $(\mathsf{Exp}_{\mathcal{G}}/_{\equiv}, \partial_{\mathcal{G}})$ *is final in the category* $Coalg(\mathcal{G})_{\mathbf{LF}}$.

Proof. We want to show that for any locally finite $\mathcal{G}$-coalgebra (S, g), there exists a *unique* homomorphism $(S, g) \to (\mathsf{Exp}_{\mathcal{G}}/_{\equiv}, \partial_{\mathcal{G}})$. We only need to prove uniqueness, since the existence is guaranteed by Lemma 4.6, where $\lceil - \rceil \colon S \to \mathsf{Exp}_{\mathcal{G}}/_{\equiv}$ is defined.

Suppose we have another homomorphism $f \colon S \to \mathsf{Exp}_{\mathcal{G}}/_{\equiv}$. Then, we shall prove that $f = \lceil - \rceil$. Let, for any $s \in S$, f_s denote any representative of $f(s)$ (that is, $f(s) = [f_s]$). First, observe that because f is a homomorphism the following holds for every $s \in S$:

$$f(s) = (\partial_{\mathcal{G}}^{-1} \circ \mathcal{G}(f) \circ g)(s) \Leftrightarrow f_s \equiv \gamma^{\mathcal{G}}_{g(s)}[f_{s_1}/x_1] \dots [f_{s_n}/x_n] \tag{4.7}$$

where $\langle s \rangle = \{s_1, \dots, s_n\}$, with $s_1 = s$ (recall that $\partial_{\mathcal{G}}^{-1}$ was defined in (4.3) and note that $\overline{\gamma}^{\mathcal{G}}_{(\mathcal{G}(f) \circ g)(s)} = \gamma^{\mathcal{G}}_{g(s)}[f_{s_i}/x_i]$).

Next, we prove that $f_{s_i} \equiv \langle\!\langle\, s_i \,\rangle\!\rangle$ (which is equivalent to $f(s_i) = \lceil\, s_i \,\rceil$), for all $i = 1, \dots n$. For simplicity we will here prove the case $n = 3$. The general case is identical but notationally heavier. First, we prove that $f_{s_1} \equiv A_1[f_{s_2}/x_2][f_{s_3}/x_3]$.

$$\begin{array}{lll}
 & f_{s_1} \equiv \gamma^{\mathcal{G}}_{g(s_1)}[f_{s_1}/x_1][f_{s_2}/x_2][f_{s_3}/x_3] & \text{(by (4.7))} \\
\Leftrightarrow & f_{s_1} \equiv \gamma^{\mathcal{G}}_{g(s_1)}[f_{s_2}/x_2][f_{s_3}/x_3][f_{s_1}/x_1] & \text{(all } f(s_i) \text{ are closed)} \\
\Rightarrow & f_{s_1} \equiv \mu x_1.\gamma^{\mathcal{G}}_{g(s_1)}[f_{s_2}/x_2][f_{s_3}/x_3] & \text{(by uniqueness of fixed points)} \\
\Leftrightarrow & f_{s_1} \equiv A_1[f_{s_2}/x_2][f_{s_3}/x_3] & \text{(def. of } A_1)
\end{array}$$

Now, using what we have computed for f_{s_1} we prove that $f_{s_2} \equiv A_2^1[f_{s_3}/x_3]$.

$$\begin{array}{lll}
 & f_{s_2} \equiv \gamma^{\mathcal{G}}_{g(s_2)}[f_{s_1}/x_1][f_{s_2}/x_2][f_{s_3}/x_3] & \text{(by (4.7))} \\
\Leftrightarrow & f_{s_2} \equiv \gamma^{\mathcal{G}}_{g(s_2)}[A_1/x_1][f_{s_2}/x_2][f_{s_3}/x_3] & \text{(expressions for } f_{s_1} \text{ and (3.3))} \\
\Leftrightarrow & f_{s_2} \equiv \gamma^{\mathcal{G}}_{g(s_2)}[A_1/x_1][f_{s_3}/x_3][f_{s_2}/x_2] & \text{(all } f(s_i) \text{ are closed)} \\
\Rightarrow & f_{s_2} \equiv \mu x_2.\gamma^{\mathcal{G}}_{g(s_2)}[A_1/x_1][f_{s_3}/x_3] & \text{(by uniqueness of fixed points)} \\
\Leftrightarrow & f_{s_2} \equiv A_2^1[f_{s_3}/x_3] & \text{(def. of } A_2^1)
\end{array}$$

At this point we substitute f_{s_2} in the expression for f_{s_1} by $A_2^1[f_{s_3}/x_3]$ which yields:

$$f_{s_1} \equiv A_1[A_2^1[f_{s_3}/x_3]/x_2][f_{s_3}/x_3] \equiv A_1[A_2^1/x_2][f_{s_3}/x_3]$$

Finally, we prove that $f_{s_3} \equiv A_3^2$:

$$\begin{array}{lll}
 & f_{s_3} \equiv \gamma^{\mathcal{G}}_{g(s_3)}[f_{s_1}/x_1][f_{s_2}/x_2][f_{s_3}/x_3] & \text{(by (4.7))} \\
\Leftrightarrow & f_{s_3} \equiv \gamma^{\mathcal{G}}_{g(s_3)}[A_1/x_1][A_2^1/x_2][f_{s_3}/x_3] & \text{(expr. for } f(s_i) \text{ and (3.3))} \\
\Rightarrow & f_{s_3} \equiv \mu x_3.\gamma^{\mathcal{G}}_{g(s_3)}[A_1/x_1][A_2^1/x_2] & \text{(by uniqueness of fixed points)} \\
\Leftrightarrow & f_{s_3} \equiv A_3^2 & \text{(def. of } A_3^2)
\end{array}$$

Thus, we have $f_{s_1} \equiv A_1[A_2^1/x_2][A_3^2/x_3]$, $f_{s_2} \equiv A_2^1[A_3^2/x_3]$ and $f_{s_3} \equiv A_3^2$. Note that $A_2^1[A_3^2/x_3] \equiv A_2^1\{A_3^2/x_3\}$ since x_2 is not free in A_3^2. Similarly, since x_1 is not free in A_2^1 and A_3^2, we have that $A_1[A_2^1/x_2][A_3^2/x_3] \equiv A_1\{A_2^1/x_2\}\{A_3^2/x_3\}$. Thus $f(s_i) = \lceil s_i \rceil$, for all $i = 1, 2, 3$. □

As a consequence of Lemma 4.8, we have that if $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic functors then $\mathsf{Exp}_{\mathcal{G}_1}/_{\equiv}$ and $\mathsf{Exp}_{\mathcal{G}_2}/_{\equiv}$ are also isomorphic (for instance, this would be true for $\mathcal{G}_1(X) = \mathsf{B} \times (X \times A)$ and $\mathcal{G}_2(X) = A \times (\mathsf{B} \times X)$).

We remark that Lemma 4.7 could have been proved as a consequence of Lemma 4.8, by observing that $(I, \overline{\omega}_{\mathcal{G}})$ is, by construction, a quotient of $(\mathsf{Exp}_{\mathcal{G}}/_{\equiv}, \partial_{\mathcal{G}})$.

At this point, because final objects are unique up-to isomorphism, we know that $e\colon \mathsf{Exp}_{\mathcal{G}}/_{\equiv} \to I$ is an isomorphism and hence we can conclude that the map $\mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}/_{\equiv}}$ is injective, since it factorizes into an isomorphism followed by a mono. This fact is the last thing we need to prove completeness.

Theorem 4.9 (Completeness)**.** *Let $\mathcal{G}$ be a non-deterministic functor. For all $\varepsilon_1, \varepsilon_2 \in \mathsf{Exp}_{\mathcal{G}}$,*

$$\varepsilon_1 \sim \varepsilon_2 \Rightarrow \varepsilon_1 \equiv \varepsilon_2$$

Proof. Let $\mathcal{G}$ be a non-deterministic functor, let $\varepsilon_1, \varepsilon_2 \in \mathsf{Exp}_{\mathcal{G}}$ and suppose that $\varepsilon_1 \sim \varepsilon_2$. Because only bisimilar elements are identified in the final coalgebra we know that it must be the case that $\mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}}(\varepsilon_1) = \mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}}(\varepsilon_2)$ and thus, since the equivalence class map $[-]$ is a homomorphism, $\mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}/_{\equiv}}([\varepsilon_1]) = \mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}/_{\equiv}}([\varepsilon_2])$. Because $\mathbf{beh}_{\mathsf{Exp}_{\mathcal{G}}/_{\equiv}}$ is injective we have that $[\varepsilon_1] = [\varepsilon_2]$. Hence, $\varepsilon_1 \equiv \varepsilon_2$. □

5. Two more examples

In this section we apply our framework to two other examples: labelled transition systems (with explicit termination) and automata on guarded strings. These two automata models are directly connected to, respectively, basic process algebra and Kleene algebra with tests. To improve readability we will present the corresponding languages using a more user-friendly syntax than the canonically derived one.

Labelled transition systems. Labelled transition systems (with explicit termination) are coalgebras for the functor $1 + (\mathcal{P}_\omega \mathsf{Id})^A$. As we will show below, instantiating our framework for this functor produces a language that is equivalent to the closed and guarded expressions generated by the following grammar, where $a \in A$ and $x \in X$ (X is a set of fixed point variables):

$$P ::= \mathbf{0} \mid P + P \mid a.P \mid \delta \mid \surd \mid \mu x.P \mid x$$

together with the equations (omitting the congruence and α-equivalence rules)

$$\begin{array}{lll}
P_1 + P_2 \equiv P_2 + P_1 & P_1 + (P_2 + P_3) \equiv (P_1 + P_2) + P_3 & \\
P + P \equiv P & P + \mathbf{0} \equiv P & \\
P + \delta \equiv P \ (\star) & \surd + \delta \equiv \surd + P \ (\star) & (\star) \text{ if } P \not\equiv \mathbf{0} \text{ and } P \not\equiv \surd \\
P[\mu x.P/x] \equiv \mu x.P & P[Q/x] \equiv Q \Rightarrow (\mu x.P) \equiv Q &
\end{array}$$

Note that, as expected, there is no law that allows us to prove $a.(P + Q) \equiv a.P + a.Q$. Moreover, observe that this syntax and axiomatization is very similar to the one presented

in [1]. In the syntax above, δ represents deadlock, $\surd$ successful termination and $\mathbf{0}$ the totally undefined process.

We will next show how the beautified syntax above was derived from the canonically derived syntax for the expressions $\varepsilon \in \mathsf{Exp}_{1+(\mathcal{P}\mathsf{Id})^A}$, which is given by the set of closed and guarded expressions defined by the following BNF:

$$\begin{array}{l}\varepsilon ::= \underline{\emptyset} \mid \varepsilon \oplus \varepsilon \mid x \mid \mu x.\varepsilon \mid l[\varepsilon_1] \mid r[\varepsilon_2] \\ \varepsilon_1 ::= \underline{\emptyset} \mid \varepsilon_1 \oplus \varepsilon_1 \mid * \\ \varepsilon_1 ::= \underline{\emptyset} \mid \varepsilon_2 \oplus \varepsilon_2 \mid a(\varepsilon') \\ \varepsilon' ::= \underline{\emptyset} \mid \varepsilon' \oplus \varepsilon' \mid \{\varepsilon\}\end{array}$$

We define two maps between this grammar and the grammar presented above. Let us start to show how to translate P's into ε's, by defining a map $(-)^\dagger$ by induction on the structure of P:

$$\begin{array}{lcllcl}(\mathbf{0})^\dagger & = & \underline{\emptyset} & (a.P)^\dagger & = & r[a(\{P^\dagger\})] \\ (P_1 + P_2)^\dagger & = & (P_1)^\dagger \oplus (P_2)^\dagger & (\surd)^\dagger & = & l[*] \\ (\mu x.P)^\dagger & = & \mu x.P^\dagger & (\delta)^\dagger & = & r[\underline{\emptyset}] \\ x^\dagger & = & x & & & \end{array}$$

And now the converse translation:

$$\begin{array}{lcllcl}(\underline{\emptyset})^\ddagger & = & \mathbf{0} & (l[*])^\ddagger & = & \surd \\ (\varepsilon_1 \oplus \varepsilon_2)^\ddagger & = & (\varepsilon_1)^\ddagger + (\varepsilon_2)^\ddagger & (r[\underline{\emptyset}])^\ddagger & = & \delta \\ (\mu x.\varepsilon)^\ddagger & = & \mu x.\varepsilon^\ddagger & (r[\varepsilon_2 \oplus \varepsilon_2'])^\ddagger & = & (r[\varepsilon_2])^\ddagger + (r[\varepsilon_2'])^\ddagger \\ x^\ddagger & = & x & (r[a(\underline{\emptyset})])^\ddagger & = & \delta \\ (l[\underline{\emptyset}])^\ddagger & = & \surd & (r[a(\varepsilon_1' \oplus \varepsilon_2')])^\ddagger & = & (r[a(\varepsilon_1')])^\ddagger + (r[a(\varepsilon_2')])^\ddagger \\ (l[\varepsilon_1 \oplus \varepsilon_1'])^\ddagger & = & (l[\varepsilon_1])^\ddagger + (l[\varepsilon_1'])^\ddagger & (r[a(\{\varepsilon\})])^\ddagger & = & a.\varepsilon^\ddagger \end{array}$$

One can prove that if $P_1 \equiv P_2$ (using the equations above) then $(P_1)^\dagger \equiv (P_2)^\dagger$ (using the automatically derived equations for the functor) and also that $\varepsilon_1 \equiv \varepsilon_2$ implies $(\varepsilon_1)^\ddagger \equiv (\varepsilon_2)^\ddagger$.

Automata on guarded strings. It has recently been shown [23] that automata on guarded strings (acceptors of the join irreducible elements of the free Kleene algebra with tests on generators Σ, T) are coalgebras for the functor $\mathsf{B} \times \mathsf{Id}^{At \times \Sigma}$, where At is the set of atoms, *i.e.* minimal nonzero elements of the free Boolean algebra B generated by T and Σ is a set of actions. Applying our framework to this functor yields a language that is equivalent to the closed and guarded expressions generated by the following grammar, where $b \in \mathsf{B}$ and $a \in \Sigma$:

$$P ::= \mathbf{0} \mid \langle b \rangle \mid P + P \mid b \to a.P \mid \mu x.P \mid x$$

accompanied by the equations (omitting the congruence and α-equivalence rules)

$$\begin{array}{ll} P_1 + P_2 \equiv P_2 + P_1 & P_1 + (P_2 + P_3) \equiv (P_1 + P_2) + P_3 \\ P + P \equiv P & P + \mathbf{0} \equiv P \\ \langle b_1 \rangle + \langle b_2 \rangle \equiv \langle b_1 \vee_{\mathsf{B}} b_2 \rangle & \mathbf{0} \equiv \langle \bot_{\mathsf{B}} \rangle \\ (b \to a.\mathbf{0}) \equiv \mathbf{0} & (\bot_{\mathsf{B}} \to a.P) \equiv \mathbf{0} \\ (b \to a.P_2) + (b \to a.P_2) \equiv b \to a.(P_1 + P_2) & (b_1 \to a.P) + (b_2 \to a.P) \equiv (b_1 \vee_{\mathsf{B}} b_2) \to a.P \\ P[\mu x.P/x] \equiv \mu x.P & P[Q/x] \equiv Q \Rightarrow (\mu x.P) \equiv Q \end{array}$$

We will not present a full comparison of this syntax to the one of Kleene algebra with tests [23] (and propositional Hoare triples). The differences between our syntax and that

of KAT are similar to the ones between regular expressions and the language $\mathsf{Exp}_{\mathcal{D}}$ for the functor representing deterministic automata (see Definition 3.5). Similarly to the LTS example one can define maps between the beautified syntax and the automatically generated one and prove its correctness.

6. Polynomial and finitary coalgebras

The functors we considered above allowed us to modularly derive languages and axiomatizations for a large class of coalgebras. If we consider the subset of *NDF* without the $\mathcal{P}_\omega$ functor, the class of coalgebras for these functors almost coincides with polynomial coalgebras (that is, coalgebras for a polynomial functor). The only difference comes from the use of join-semilattices for constant functors and $\dot{+}$ instead of the ordinary coproduct, which played an important role in order for us to be able to have underspecification and overspecification. We will next show how to derive expressions and axiomatizations directly for polynomial coalgebras, where no underspecification or overspecification is allowed.

Before we show the formal definition, let us provide some intuition. The main changes[4], compared to the previous sections, would be not to have $\underline{\emptyset}$ and $\oplus$ and consider an expression $\langle -, - \rangle$ for the product instead of the two expressions $l\langle - \rangle$ and $r\langle - \rangle$ which we considered and an expression $\langle a_1(-), a_2(-), \ldots, a_n(-) \rangle$ for the exponential (with $A = \{a_1, \ldots a_n\}$). As an example, take the functor $\mathcal{D}(X) = 2 \times X^A$ of deterministic automata. The expressions corresponding to this functor would then be the set of closed and guarded expressions given by the following BNF:

$$\varepsilon ::= x \mid \mu x.\varepsilon \mid \langle 0, \langle a_1(\varepsilon), a_2(\varepsilon), \ldots, a_n(\varepsilon) \rangle \rangle \mid \langle 1, \langle a_1(\varepsilon), a_2(\varepsilon), \ldots, a_n(\varepsilon) \rangle \rangle$$

This syntax can be perceived as an explicit and complete description of the automaton. This means that underspecification is nonexistent and the compactness of regular expressions is lost. As an example of the verbosity present in this new language, take $A = \{a, b, c\}$ and consider the language that accepts words with only a's and has at last one a (described by aa^* in Kleene's regular expressions). In the language $\mathsf{Exp}_{\mathcal{D}}$ it would be written as $\mu x.a(l\langle 1 \rangle \oplus x)$. Using the approach described above it would be encoded as the expression

$$\mu x.\langle 0, \langle a(\langle 1, \langle a(x), b(\mathit{empty}), c(\mathit{empty}) \rangle \rangle), b(\mathit{empty}), c(\mathit{empty}) \rangle \rangle$$

where $\mathit{empty} = \mu y.\langle 0, \langle a(y), b(y), c(y) \rangle$ is the expression denoting the empty language. The approach we presented before, by allowing underspecification, provides a more user-friendly syntax and stays close to the know syntaxes for deterministic automata and LTSs.

In what follows we will formally present a language for polynomial coalgebras. We start by introducing the definition of polynomial functor, which we take from [2].

Definition 6.1 (Polynomial Functor)**.** Sums of the Cartesian power functors are called polynomial functors:

$$\mathbf{P}_\Sigma(X) = \coprod_{\sigma \in \Sigma} X^{ar(\sigma)}$$

Here, $\coprod$ stands for ordinary coproduct and the indexing set Σ is a signature, that is a possibly infinite collection of symbols σ, each of which is equipped with a finite cardinal $ar(\sigma)$, called the arity of σ. ♣

[4]This syntax was suggested to us by B. Klin, during CONCUR'09.

Definition 6.2 (Expressions and axioms for polynomial functors)**.** Let $\mathbf{P}_\Sigma$ be a polynomial functor. The set $\mathsf{Exp}_{\mathbf{P}_\Sigma}$ of expressions for $\mathbf{P}_\Sigma$ is given by the closed and guarded expressions generated by the following BNF, where $\sigma \in \Sigma$ and $x \in V$, for V a set of fixed point variables:

$$\varepsilon_i ::= x \mid \mu x.\varepsilon \mid \sigma(\varepsilon_1, \ldots, \varepsilon_{ar(\sigma)})$$

accompanied by the equations:

$$\begin{aligned}
&\gamma[\mu x.\gamma/x] \equiv \mu x.\gamma && (FP)\\
&\gamma[\varepsilon/x] \equiv \varepsilon \Rightarrow \mu x.\gamma \equiv \varepsilon && (Unique)\\
&\varepsilon_1 \equiv \varepsilon_2 \Rightarrow \varepsilon[\varepsilon_1/x] \equiv \varepsilon[\varepsilon_2/x], \quad \text{if } x \text{ is free in } \varepsilon && (Cong)\\
&\mu x.\gamma \equiv \mu y.\gamma[y/x], \quad \text{if } y \text{ is not free in } \gamma && (\alpha - equiv)
\end{aligned}$$

♣

Providing the set $\mathsf{Exp}_{\mathbf{P}_\Sigma}$ with a coalgebraic structure is achieved using induction on the number of unguarded occurrences of nested fixed points:

$$\begin{aligned}
&\delta\colon \mathsf{Exp}_{\mathbf{P}_\Sigma} \to \coprod_{\sigma\in\Sigma} (\mathsf{Exp}_{\mathbf{P}_\Sigma})^{ar(\sigma)}\\
&\delta(\mu x.\varepsilon) = \delta(\varepsilon[\mu x.\varepsilon/x])\\
&\delta(\sigma(\varepsilon_1, \ldots, \varepsilon_{ar(\sigma)})) = \kappa_\sigma(\langle \varepsilon_1, \ldots, \varepsilon_{ar(\sigma)}\rangle)
\end{aligned}$$

We are now ready to state and prove Kleene's theorem.

Theorem 6.3 (Kleene's theorem for polynomial functors)**.** *Let $\mathbf{P}_\Sigma$ be a polynomial functor.*

(1) *For every locally finite coalgebra $(S, g\colon S \to \mathbf{P}_\Sigma(S))$ and for every $s \in S$ there exists an expression $\varepsilon \in \mathsf{Exp}_{\mathbf{P}_\Sigma}$ such that $\varepsilon \sim s$.*
(2) *For every expression $\varepsilon \in \mathsf{Exp}_{\mathbf{P}_\Sigma}$ there is a finite coalgebra $(S, g\colon S \to \mathbf{P}_\Sigma(S))$ with $s \in S$ such that $s \sim \varepsilon$.*

Proof. Point 1. amounts to solve a system of equations. Let $\langle s\rangle = \{s_1, \ldots, s_n\}$. We associate with each $s_i \in \langle s\rangle$ an expression $\langle\!\langle\, s_i \,\rangle\!\rangle = A_i^n$, where A_i^n is defined inductively as in the proof of 3.12, with $A_i^{k+1} = A_i^k\{A_{k+1}^k/x_{k+1}\}$ and $A_i^0 = A_i$ given by

$$A_i = \mu x_{s_i}.\sigma(x_{s'_1}, \ldots, x_{s'_{ar(\sigma)}}), \quad g(s_i) = \kappa_\sigma(s'_1, \ldots, s'_{ar(\sigma)})$$

It remains to prove that $s_i \sim \langle\!\langle\, s_i \,\rangle\!\rangle$, for all $s_i \in \langle s\rangle$. We observe that

$$R = \{\langle s_i, \langle\!\langle\, s_i \,\rangle\!\rangle\rangle \mid s_i \in \langle s\rangle\}$$

is a bisimulation, since, for $g(s_i) = \kappa_\sigma(s'_1, \ldots, s'_{ar(\sigma)})$, we have

$$\begin{aligned}
&\delta(\langle\!\langle\, s_i \,\rangle\!\rangle)\\
&= \delta((\mu x_i.\sigma(x_{s'_1}, \ldots, x_{s'_{ar(\sigma)}}))\{A_1^0/x_1\}\ldots\{A_n^{n-1}/x_n\})\\
&= \delta(\mu x_i.\sigma(x_{s'_1}, \ldots, x_{s'_{ar(\sigma)}})\{A_1^0/x_1\}\ldots\{A_{i-1}^{i-2}/x_{i-1}\}\{A_{i+1}^i/x_{i+1}\}\ldots\{A_n^{n-1}/x_n\})\\
&= \delta(\sigma(x_{s'_1}, \ldots, x_{s'_{ar(\sigma)}})\{A_1^0/x_1\}\ldots\{A_{i-1}^{i-2}/x_{i-1}\}\{A_{i+1}^i/x_{i+1}\}\ldots\{A_n^{n-1}/x_n\}[A_i^n/x_i])\\
&= \delta(\sigma(x_{s'_1}, \ldots, x_{s'_{ar(\sigma)}})\{A_1^0/x_1\}\ldots\{A_{i-1}^{i-2}/x_{i-1}\}\{A_{i+1}^i/x_{i+1}\}\ldots\{A_n^{n-1}/x_n\}\{A_i^n/x_i\})\\
&= \delta(\sigma(x_{s'_1}, \ldots, x_{s'_{ar(\sigma)}})\{A_1^0/x_1\}\ldots\{A_{i-1}^{i-2}/x_{i-1}\}\{A_i^n/x_i\}\{A_{i+1}^i/x_{i+1}\}\ldots\{A_n^{n-1}/x_n\})\\
&= \kappa_\sigma(\langle\!\langle\, s'_1 \,\rangle\!\rangle, \ldots, \langle\!\langle\, s'_{ar(\sigma)} \,\rangle\!\rangle)
\end{aligned}$$

For point 2, we observe that the subcoalgebra $\langle\varepsilon\rangle$, for any $\varepsilon \in \mathsf{Exp}_{\mathbf{P}_\Sigma}$ is finite, since the set $cl(\varepsilon)$ containing all sub-formulas and unfoldings of fixed points of ε, which is finite, is a subcoalgebra of $(\mathsf{Exp}_{\mathbf{P}_\Sigma}, \delta)$. The fact that in this point, contrary to what happened in Theorem 3.14, we do not need to quotient the set of expressions is a direct consequence of the absence of underspecification or, more concretely, of the expressions $\underline{\emptyset}$ and $\oplus$. □

The proof of soundness and completeness would follow a similar strategy as in the previous section and we will omit it here.

In order to be able to compare the language introduced in this section with the language obtained in our previous approach, we have to define an infinitary version of the operator $\mathbin{\Diamond\!\!\!\!+}$ and extend the framework accordingly. We start by defining the aforementioned operator on sets: $\bigdiamondplus_{i\in I} X_i = \left(\coprod_{i\in I} X_i\right) \cup \{\bot, \top\}$ and the corresponding functor, for which we shall use the same symbol, is defined pointwise in the same way as for $\mathbin{\Diamond\!\!\!\!+}$. Note that $\mathbin{\Diamond\!\!\!\!+}$ is a special case of this operator (resp. functor) for I a two element set. In fact, for simplicity, we shall only consider this operator for index sets I with two or more elements.

There is a natural transformation between polynomial functors and the class of non-deterministic functors extended with $\bigdiamondplus$: every polynomial functor $\mathbf{P}_\Sigma$ is mapped to

$$\overline{\mathbf{P}}_\Sigma(X) = \bigdiamondplus_{\sigma\in\Sigma} X^{ar(\sigma)}$$

Next, we slightly alter the definition of expressions. Instead of the expressions $l[-]$ and $r[-]$ we had before for $\mathbin{\Diamond\!\!\!\!+}$ we add an expression $i[-]$ for each $i \in I$ and the expected typing rule:

$$\frac{\vdash \varepsilon \colon \mathcal{F}_j \lhd \mathcal{G} \qquad j \in I}{\vdash j[\varepsilon] \colon \bigdiamondplus_{i\in I} \mathcal{F}_i \lhd \mathcal{G}}$$

All the other elements in our story are adjusted in the expected way. We show what happens in the axiomatization. For $\mathbin{\Diamond\!\!\!\!+}$ we had the rules

$$l[\varepsilon_1 \oplus \varepsilon_2] \equiv l[\varepsilon_1] \oplus l[\varepsilon_2] \qquad r[\varepsilon_1 \oplus \varepsilon_2] \equiv r[\varepsilon_1] \oplus r[\varepsilon_2] \qquad l[\varepsilon_1] \oplus r[\varepsilon_2] \equiv l[\underline{\emptyset}] \oplus r[\underline{\emptyset}]$$

which are now replaced by

$$i[\varepsilon_1] \oplus i[\varepsilon_2] \equiv i[\varepsilon_1 \oplus \varepsilon_2] \qquad i[\varepsilon_1] \oplus j[\varepsilon_2] \equiv k[\underline{\emptyset}] \oplus l[\underline{\emptyset}], \quad i \neq j, k \neq l$$

It is natural to ask what is the relation between the sets of expressions $\mathsf{Exp}_{\mathbf{P}_\Sigma}$ and $\mathsf{Exp}_{\overline{\mathbf{P}}_\Sigma}$. The set $\mathsf{Exp}_{\mathbf{P}_\Sigma}$ is bijective to the subset of $\mathsf{Exp}_{\overline{\mathbf{P}}_\Sigma}$ containing only fully specified expressions, that is expressions ε for which the subcoalgebra $\langle\varepsilon\rangle$ does not contain any state for which $\delta_{\overline{\mathbf{P}}_\Sigma}$ evaluates to $\bot$ and $\top$. This condition is purely semantical and we were not able to find a purely syntactic restriction that would capture it.

We next repeat the exercise above for finitary functors. A finitary functor $\mathbf{F}$ is a functor that is a quotient of a polynomial functor, *i.e.* there exists a natural transformation $\eta \colon \mathbf{P}_\Sigma \to \mathbf{F}$, whose components $\eta_X \colon \mathbf{P}_\Sigma(X) \to \mathbf{F}(X)$ are epimorphisms. We define $\mathsf{Exp}_{\mathbf{F}} = \mathsf{Exp}_{\mathbf{P}_\Sigma}$.

Theorem 6.4 (Kleene's theorem for finitary functors)**.** *Let $\mathbf{F}$ be a finitary functor.*

(1) *Let (S, f) be a locally-finite $\mathbf{F}$-coalgebra. Then, for any $s \in S$, there exists an expression $\langle\!\langle\, s \,\rangle\!\rangle \in \mathsf{Exp}_{\mathbf{F}}$ such that $s \sim \langle\!\langle\, s \,\rangle\!\rangle$.*
(2) *Let $\varepsilon \in \mathsf{Exp}_{\mathbf{F}}$. Then, there exists a finite $\mathbf{F}$-coalgebra (S, f) with $s \in S$ such that $s \sim \varepsilon$.*

Proof. Let $\mathbf{F}$ be a finitary functor (quotient of a polynomial functor $\mathbf{P}_\Sigma$).

① Let (S, f) be a locally finite $\mathbf{F}$-coalgebra and let $s \in S$. We denote by $T = \{s_1, \ldots, s_n\}$ the state space of the subcoalgebra $\langle s \rangle$ (with $s_1 = s$). We then have that there exists an $f^\sharp$ making the following diagram commute:

$$\begin{array}{ccccc} T & \xrightarrow{id} & T & \longrightarrow & S \\ {\scriptstyle f^\sharp}\downarrow & & \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle f} \\ \mathbf{P}_\Sigma(T) & \xrightarrow[\eta_S]{} & \mathbf{F}(T) & \longrightarrow & \mathbf{F}(S) \end{array}$$

We then build $\langle\!\langle\, s \,\rangle\!\rangle$ w.r.t $f^\sharp$ just as in Theorem 3.12 (note that $(T, f^\sharp)$ is finite) and the result follows because $\langle\!\langle\, s \,\rangle\!\rangle \sim_{\mathbf{F}} s \Leftarrow \langle\!\langle\, s \,\rangle\!\rangle \sim_{\mathbf{P}_\Sigma} s$ (consequence of naturality).

② Let $\varepsilon \in \mathsf{Exp}_{\mathbf{F}}$. By Theorem 3.14, there exists a finite $\mathbf{P}_\Sigma$-coalgebra (S, f) with $s \in S$ such that $s \sim_{\mathbf{P}_\Sigma} \varepsilon$. Thus, we take $(S, \eta_S \circ f)$ and we have a finite $\mathbf{F}$-coalgebra with $s \in S$ such that $\varepsilon \sim_{\mathbf{F}} s$. □

For the axiomatization a bit more ingenuity is required. One needs to derive which extra axioms are induced by the epimorphism and then prove that they are sound and complete.

For instance, the finite powerset functor (which we included in the syntax of non-deterministic functors) is the classical example of a finitary functor. It is the quotient of the polynomial functor $\mathbf{P}_\Sigma(X) = 1 + X + X^2 + \ldots$ (this represents lists of length n) by identifying lists that contain precisely the same elements (that is, eliminating repeated elements and abstracting from the ordering).

The syntax for $\mathsf{Exp}_{\mathbf{P}_\Sigma}$ is the set of closed and guarded expressions given by the following BNF:

$$\varepsilon ::= x \mid \mu x.\varepsilon \mid i(\varepsilon_1, \ldots, \varepsilon_i), \quad i \in \mathbb{N}$$

together with the axioms for the fixed point, $(\alpha - equiv)$ and $(Cong)$.

Taking into account the restriction mentioned we would have to include the extra axioms:

$$i(\varepsilon_1, \ldots, \varepsilon_i) \equiv i(\varepsilon'_1, \ldots, \varepsilon'_i) \text{ if } \{\varepsilon_1, \ldots \varepsilon_i\} = \{\varepsilon'_1, \ldots \varepsilon'_i\}$$
$$i(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_i) \equiv (i-1)(\varepsilon_1, \varepsilon_3, \ldots, \varepsilon_i) \text{ if } \varepsilon_1 \equiv \varepsilon_2$$

In this case, one can see that this set of axioms is sound and complete, by simply proving, for $\mathbf{P}_\Sigma(X) = 1 + X + X^2 + \ldots$, $\mathsf{Exp}_{\mathbf{P}_\Sigma}/_\equiv \cong \mathsf{Exp}_{\mathcal{P}_\omega}/_\equiv$ (since we already had a language and sound and complete axiomatization for the $\mathcal{P}_\omega$ functor). The restricted syntax and axioms needs to be derived for each concrete finitary functor. Finding a uniform way of defining such restricted syntax/axioms and also uniformly proving soundness and completeness is a challenging problem and it is left as future work.

7. Discussion

We presented a systematic way of deriving, from the type of a system, a language of (generalized) regular expressions and a sound and complete axiomatization thereof. We showed the analogue of Kleene's theorem, proving the correspondence of the behaviours captured by the expressions and the systems under consideration. The whole approach was illustrated with five examples: deterministic finite automata, partial deterministic automata, non-deterministic automata, labelled transition systems and automata on guarded strings. Moreover, all the results presented in [9] for Mealy machines can be recovered as a particular instance of the present framework.

Iterative theories have been introduced by Elgot [15] as a model of computation and they formalize potentially infinite computations as solutions of recursive equations. The main example of an iterative theory is the theory of regular trees, that is trees which have on finitely many distinct subtrees. Adámek, Milius and Velebil have presented Elgot's work from a coalgebraic perspective [3, 4], simplified some of his original proofs, and generalized the notion of free iterative theory to any finitary endofunctor of every locally presentable category. The language modulo the axioms we will associate with each functor is closely related to the work above: it is an initial iterative algebra. This also shows the connection of our work with the work by Bloom and Ésik on iterative algebras/theories [5]. It would be interesting to investigate the connections with iterative algebras further.

In [20], a bialgebraic review of deterministic automata and regular expressions was presented. One of the main results of [20] was a description of the free algebra and Brzozowski coalgebra structure on regular expressions as a bialgebra with respect to a GSOS law. We expect that this extends to our framework, but fully working this out is left as future work.

In this paper we studied coalgebras for **Set** functors. It is an important and challenging question to extend our results to other categories. Following our work, S. Milius [27] has showed how to derive a language and sound and complete axiomatization for the functor $\mathbb{R} \times \mathsf{Id}$ in the category of vector spaces and linear maps. It would also be interesting to study functors over metric spaces [36, 12].

In his seminal paper [21], S. Kleene introduced an algebraic description of regular languages: regular expressions. This was the precursor of many papers, including this one. Salomaa [33] presented a sound and complete axiomatization for proving the equivalence of regular expressions. This was later refined by Kozen in [22]: he showed that Salomaa's axiomatization is non-algebraic, in the sense that it is unsound under substitution of alphabet symbols by arbitrary regular expressions, and presented an algebraic axiomatization. In [28], Milner introduced a set of expressions for finite LTS's and proved an analogue of Kleene's theorem: each expression denotes the behaviour of a finite LTS and, conversely, the behaviour of a finite LTS can be specified by an expression. He also provided an axiomatization for his expressions, with the property that two expressions are provably equivalent if and only if they are bisimilar.

Our approach is inspired by the work of Kleene, Kozen and Milner. For that reason, we have $\underline{\emptyset}$ and $\oplus$ in the syntax of our expressions, which allow to have underspecification and overspecification. These features had to be reflected in the type of the coalgebras we are able to deal with: the class of functors considered include join-semilattices as constant functors and $\dot{+}$ instead of the ordinary coproduct, which has allowed us to remain in the category **Set**. The fact that underspecification and overspecification can be captured by a semilattice structure, plus the fact that the axiomatization provides the set of expressions with a join semilattice structure, hint (as one of the reviewers pointed out) that the whole framework could have been studied directly in the category of join-semilattices. This is indeed true, but, for simplicity, we decided to remain in the category **Set**. It is not clear how much could be gained by directly working on join semi-lattices.

The connection between regular expressions and coalgebras was first explored in [32]. There deterministic automata, the set of formal languages and regular expressions are all presented as coalgebras of the functor $2 \times \mathsf{Id}^A$ (where A is the alphabet, and 2 is the two element set). It is then shown that the standard semantics of language acceptance of automata and the assignment of languages to regular expressions both arise as the unique homomorphism into the final coalgebra of formal languages. The coalgebra structure on the

set of regular expressions is determined by their so-called *Brzozowski* derivatives [13]. In the present paper, the set of expressions for the functor $\mathcal{D}(S) = 2 \times S^A$ differs from the classical definition in that we do not have Kleene star and full concatenation (sequential composition) but, instead, the least fixed point operator and action prefixing. Modulo that difference, the definition of a coalgebra structure on the set of expressions in both [32] and the present paper is essentially the same. All in all, one can therefore say that standard regular expressions and their treatment in [32] can be viewed as a special instance of the present approach. This is also the case for the generalization of the results in [32] to automata on guarded strings [23]. Finally, the present paper extends the results in our FoSSaCS'08 paper [9], where a sound and complete specification language and a synthesis algorithm for Mealy machines is given. Mealy machines are coalgebras of the functor $(\mathsf{B} \times \mathsf{Id})^A$, where A is a finite input alphabet and B is a finite semilattice for the output alphabet. Part of the material of the present paper is based on two conference papers: our FoSSaCS'09 paper [11] and our LICS'09 paper [10].

In the last few years, several proposals of specification languages for coalgebras appeared [29, 30, 19, 17, 14, 7, 8, 34, 24]. Our approach is similar in spirit to that of [17, 30, 19, 34] in that we use the ingredients of a functor for typing expressions, and differs from [30, 19] because we do not need an explicit "next-state" operator, as we can deduce it from the type information. The modal operators associated to a functor in [30, 19, 34] can easily be related with the expressions considered in our language. As an example, consider the expression $\langle \pi_2 \rangle [\kappa_1] \langle \alpha \rangle \bot$, written in the syntax of [30], which belongs to the language associated with the functor $2 \times (\mathsf{Id} + 1)$ (the modal operator $\langle \alpha \rangle$ is next operator associated with the identity functor). In our language, this would be represented by $r\langle l[\underline{\emptyset}] \rangle$.

Apart from [24], the languages mentioned above do not include fixed point operators. Our language of regular expressions can be seen as an extension of the coalgebraic logic of [7] with fixed point operators, as well as the multi-sorted logics of [34], and it is similar to a fragment of the logic presented in [24]. However, our goal is rather different: we want (1) a finitary language that characterizes exactly all *locally finite* coalgebras; (2) a Kleene like theorem for the language or, in other words, a map (and not a relation) from expressions to coalgebras and vice-versa. Similar to many of the works above, we also derive a modular axiomatization, sound and complete with respect to observational equivalence. From the perspective of modal logic, the second half of Kleene's theorem, where we show how to construct a coalgebra from an expression, is the same as constructing a canonical model. In [34], the models presented for the multi-sorted logics are multi-sorted coalgebras, whereas here we remain in the world of coalgebras in the category **Set**, constructing, from an expression in $\mathsf{Exp}_{\mathcal{G}}$, for a given functor $\mathcal{G}$, a $\mathcal{G}$-coalgebra. Further exploring the connections with the approach presented in [34] is a promising research path, opening the door to extending our framework for more general classes of functors.

In conclusion, we mention a recent generalization of the present approach: all the results presented in this paper can be extended in order to accommodate systems with *quantities*, such as probability or costs [6]. The main technical challenge is that quantitative systems have an inherently non-idempotent behaviour and thus the proof of Kleene's theorem and the axiomatization require extra care. This extension allows for the derivation of specification languages and axiomatizations for a wide variety of systems, which include weighted automata, simple probabilistic systems (also known as Markov chains) and systems with mixed probability and non-determinism (such as Segala systems). For instance, we have

derived a language and an axiomatization for the so-called stratified systems. The language is equivalent to the one presented in [16], but no axiomatization was known.

The derivation of the syntax and axioms associated with each non-deterministic functor has been implemented in the coinductive prover CIRC [26]. This allows for automatic reasoning about the equivalence of expressions specifying systems.

Acknowledgements. The authors are grateful for useful comments from several people: Filippo Bonchi, Helle Hansen, Bartek Klin, Dexter Kozen, Clemens Kupke, Stefan Milius, Prakash Panagaden, Ana Sokolova, Yde Venema and Erik de Vink. The title of this paper was inspired by the title of a section of a paper of Dexter Kozen [23]. The proof of soundness and completeness was simplified (when compared with the one presented in our LICS paper [10]) inspired by recent work of Stefan Milius on expressions for linear systems (personal communication). Finally, we would like to thank the three anonymous reviewers for their very detailed reports, which greatly improved the presentation of the paper.

References

[1] Luca Aceto and Matthew Hennessy. Termination, deadlock, and divergence. *J. ACM*, 39(1):147–187, 1992. pages 21, 31

[2] Jirí Adámek, Dominik Lücke, and Stefan Milius. Recursive coalgebras of finitary functors. *ITA*, 41(4):447–462, 2007. pages 32

[3] Jirí Adámek, Stefan Milius, and Jiri Velebil. Free iterative theories: A coalgebraic view. *Mathematical Structures in Computer Science*, 13(2):259–320, 2003. pages 36

[4] Jirí Adámek, Stefan Milius, and Jiri Velebil. Iterative algebras at work. *Mathematical Structures in Computer Science*, 16(6):1085–1131, 2006. pages 36

[5] S.L. Bloom and Z. Ésik. *Iteration theories: the equational logic of iterative processes*. EATCS Monographs on Theoretical Computer Science. Springer, 1993. pages 36

[6] Filippo Bonchi, Marcello M. Bonsangue, Jan J. M. M. Rutten, and Alexandra Silva. Deriving syntax and axioms for quantitative regular behaviours. In Mario Bravetti and Gianluigi Zavattaro, editors, *CONCUR*, volume 5710 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2009. pages 37

[7] Marcello M. Bonsangue and Alexander Kurz. Duality for logics of transition systems. In Vladimiro Sassone, editor, *FoSSaCS*, volume 3441 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 2005. pages 37

[8] Marcello M. Bonsangue and Alexander Kurz. Presenting functors by operations and equations. In Luca Aceto and Anna Ingólfsdóttir, editors, *FoSSaCS*, volume 3921 of *LNCS*, pages 172–186. Springer, 2006. pages 37

[9] Marcello M. Bonsangue, Jan J. M. M. Rutten, and Alexandra Silva. Coalgebraic logic and synthesis of Mealy machines. In Roberto M. Amadio, editor, *FoSSaCS*, volume 4962 of *Lecture Notes in Computer Science*, pages 231–245. Springer, 2008. pages 1, 4, 35, 37

[10] Marcello M. Bonsangue, Jan J. M. M. Rutten, and Alexandra Silva. An algebra for Kripke polynomial coalgebras. In *LICS*, pages 49–58. IEEE Computer Society, 2009. pages 2, 37, 38

[11] Marcello M. Bonsangue, Jan J. M. M. Rutten, and Alexandra Silva. A Kleene theorem for polynomial coalgebras. In Luca de Alfaro, editor, *FOSSACS*, volume 5504 of *Lecture Notes in Computer Science*, pages 122–136. Springer, 2009. pages 2, 37

[12] Franck van Breugel and James Worrell. Approximating and computing behavioural distances in probabilistic transition systems. *Theor. Comput. Sci.*, 360(1-3):373–385, 2006. pages 36

[13] Janusz A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964. pages 8, 10, 15, 27, 37

[14] Corina Cîrstea and Dirk Pattinson. Modular construction of modal logics. In Philippa Gardner and Nobuko Yoshida, editors, *CONCUR*, volume 3170 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2004. pages 37

[15] C.C. Elgot. Monadic computation and iterative algebraic theories. In H.E. Rose and J.C. Shepherdson, editors, *Logic Colloquium '73*. North-Holland Publishers, 1975. pages 36

[16] Rob J. van Glabbeek, Scott A. Smolka, and Bernhard Steffen. Reactive, generative and stratified models of probabilistic processes. *Inf. Comput.*, 121(1):59–80, 1995. pages 38

[17] Robert Goldblatt. Equational logic of polynomial coalgebras. In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyaschev, editors, *Advances in Modal Logic 4*, pages 149–184. King's College Publications, 2002. pages 37

[18] Claudio Hermida and Bart Jacobs. Structural induction and coinduction in a fibrational setting. *Inf. Comput.*, 145(2):107–152, 1998. pages 5

[19] Bart Jacobs. Many-sorted coalgebraic modal logic: a model-theoretic study. *ITA*, 35(1):31–59, 2001. pages 37

[20] Bart Jacobs. A bialgebraic review of deterministic automata, regular expressions and languages. In Kokichi Futatsugi, Jean-Pierre Jouannaud, and José Meseguer, editors, *Essays Dedicated to Joseph A. Goguen*, volume 4060 of *Lecture Notes in Computer Science*, pages 375–404. Springer, 2006. pages 36

[21] Stephen Kleene. Representation of events in nerve nets and finite automata. *Automata Studies*, pages 3–42, 1956. pages 1, 36

[22] Dexter Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In *Logic in Computer Science*, pages 214–225, 1991. pages 1, 36

[23] Dexter Kozen. On the coalgebraic theory of Kleene algebra with tests. Technical Report http://hdl.handle.net/1813/10173, Computing and Information Science, Cornell University, March 2008. pages 31, 37, 38

[24] Clemens Kupke and Yde Venema. Coalgebraic automata theory: Basic results. *Logical Methods in Computer Science*, 4(4), 2008. pages 37

[25] Alexander Kurz, Marina Lenisa, and Andrzej Tarlecki, editors. *Algebra and Coalgebra in Computer Science, Third International Conference, CALCO 2009, Udine, Italy, September 7-10, 2009. Proceedings*, volume 5728 of *Lecture Notes in Computer Science*. Springer, 2009. pages 39

[26] Dorel Lucanu, Eugen-Ioan Goriac, Georgiana Caltais, and Grigore Rosu. Circ: A behavioral verification tool based on circular coinduction. In Kurz et al. [25], pages 433–442. pages 38

[27] Stefan Milius. A sound and complete calculus for finite stream circuits. In *LICS*, 2010. To appear. pages 36

[28] Robin Milner. A complete inference system for a class of regular behaviours. *J. Comput. Syst. Sci.*, 28(3):439–466, 1984. pages 1, 7, 36

[29] Larry Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96, 1999. pages 37

[30] Martin Rößiger. Coalgebras and modal logic. *Electronic Notes in Theoretical Computer Science*, 33, 2000. pages 37

[31] Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000. pages 2, 4, 5, 6, 25

[32] Jan 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 *Lecture Notes in Computer Science*, pages 194–218, 1998. pages 8, 36, 37

[33] Arto Salomaa. Two complete axiom systems for the algebra of regular events. *J. ACM*, 13(1):158–169, 1966. pages 36

[34] Lutz Schröder and Dirk Pattinson. Modular algorithms for heterogeneous modal logics. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP*, volume 4596 of *Lecture Notes in Computer Science*, pages 459–471. Springer, 2007. pages 37

[35] Sam Staton. Relating coalgebraic notions of bisimulation. In Kurz et al. [25], pages 191–205. pages 5

[36] Daniele Turi and Jan J. M. M. Rutten. On the foundations of final coalgebra semantics: non-well-founded sets, partial orders, metric spaces. *Mathematical Structures in Computer Science*, 8(5):481–540, 1998. pages 36