Context-Free Languages, Coalgebraically

Joost Winter[1]*, Marcello M. Bonsangue[2,1], and Jan Rutten[1,3]

[1] Centrum Wiskunde & Informatica (CWI)
[2] LIACS – Leiden University
[3] Radboud University Nijmegen

Abstract. We give a coalgebraic account of context-free languages using the functor $\mathcal{D}(X) = 2 \times X^A$ for deterministic automata over an alphabet A, in three different but equivalent ways: (i) by viewing context-free grammars as $\mathcal{D}$-coalgebras; (ii) by defining a format for behavioural differential equations (w.r.t. $\mathcal{D}$) for which the unique solutions are precisely the context-free languages; and (iii) as the $\mathcal{D}$-coalgebra of generalized regular expressions in which the Kleene star is replaced by a unique fixed point operator. In all cases, semantics is defined by the unique homomorphism into the final coalgebra of all languages, thus paving the way for coinductive proofs of context-free language equivalence. Furthermore, the three characterizations are elementary to the extent that they can serve as the basis for the definition of a general coalgebraic notion of context-freeness, which we see as the ultimate long-term goal of the present study.

1 Introduction

The set $\mathcal{P}(A^*)$ of all formal languages over an alphabet A is a final coalgebra of the functor $\mathcal{D}(X) = 2 \times X^A$. Deterministic automata are $\mathcal{D}$-coalgebras and their behaviour, in terms of language acceptance is given by the final homomorphism into $\mathcal{P}(A^*)$. A language is *regular* if it is in the image of the final homomorphism from a *finite* $\mathcal{D}$-coalgebra to $\mathcal{P}(A^*)$. Or, equivalently by Kleene's theorem, if it is in the image of the final homomorphism from the set of *regular expressions*, which constitute a $\mathcal{D}$-coalgebra by means of the so-called Brzozowski derivatives.

Thus the coalgebraic picture of regular languages and regular expressions is well-understood (cf. [9] for details). Moreover the picture is so elementary that it has recently been possible [12] to generalize it to a large class of other systems, including Mealy machines, labelled transition systems, and various probabilistic automata.

In the present paper, we will develop in part a similar such coalgebraic picture for *context-free* languages, which form another well-known class, extending regular languages. Our focus will be on *context-free grammars*, which constitute one of the common definition schemes for context-free languages. (Another well-known characterization is through pushdown automata, which will not be treated here.)

* Supported by the NWO project CoRE.

Because the set of all languages is a final coalgebra of the functor $\mathcal{D}(X) = 2 \times X^A$, as was mentioned above, it seems natural to try and use the same functor for a coalgebraic treatment of context-free grammars and languages. We will do so in three different but equivalent ways. (i) We will, in Sect. 3, view context-free grammars (in essentially Greibach normal form) as $\mathcal{D}$-coalgebras, which we shall call grammar coalgebras; their elements will correspond to partial derivations. (ii) Next we will, in Sect. 4, define a format of behavioural differential equations with respect to the functor $\mathcal{D}$, for which the unique solutions are precisely the context-free languages. (iii) In Sect. 5, we will define context-free languages by means of generalized regular expressions, in which the Kleene star is replaced by a unique fixed point operator, and which are given a $\mathcal{D}$-coalgebra structure using a variation of Brzozowski derivatives. In all three cases, semantics is defined by the final homomorphism into $\mathcal{P}(A^*)$.

We show that the above three coalgebraic characterizations are equivalent in the following sense: a language is context-free if and only if it is in the image of the final homomorphism starting in either a grammar $\mathcal{D}$-coalgebra; or a $\mathcal{D}$-coalgebra corresponding to a (finite) system of behavioural differential equations; or the $\mathcal{D}$-coalgebra of generalized expressions with fixed point operator.

The proofs of these equivalences are not trivial, but contain few surprises, consisting of ingredients that are already present at various places in the literature. What we do see as the contribution of this paper are the three characterizations as such, together with the fact that their equivalence could be established in such an elementary fashion. We expect that this will lead to various further results, as follows.

Grammar coalgebras establish a direct correspondence between context-free grammars and context-free languages, by finality, and thus pave the way for coinductive proofs of context-free language equivalence. Furthermore, we will argue that our way of defining context-free languages through behavioural differential equations will lead to a generalization of the very notion of context-freeness to other types of systems, much in the same way as regular expressions and regular languages were generalized in [12]. A sketch of an elementary but interesting first instance hereof is given at the end of the present paper, in Sect. 6, where we will introduce the new notion of *context-free streams*. Finally, expressions with a fixed point operator are well-suited for the formulation of algebraic characterizations, which we see as yet another direction for future research.

Related work. In contrast to regular languages, equality of context-free languages is known to be an undecidable property [4]. This may explain why not so much algebraic or coalgebraic work has been devoted to study the theory of context-free languages. The first, and only, coalgebraic treatment of context-free languages we are aware of, is presented in [3]. In this paper context-free languages are described indirectly, as the result of flattening finite skeletal parsed trees. The authors study context-free grammars as coalgebras for a functor different form ours, i.e. the functor $\mathcal{P}((A + (-))^*)$.

Algebraically, the starting point is Kozen's complete characterization of regular languages in terms of Kleene algebras, idempotent semirings equipped with a star-operation satisfying some fixed point equations [6]. In [7, 1], Kleene algebras have been extended with a least fixed point operator to axiomatize fragments of the theory of context-free languages. We take a similar approach, but coalgebraic in nature and substituting the Kleene star with a unique fixed point operator. Whereas [7, 1] are interested in providing solutions to systems of equations of the form $x = t$ using least fixed points, we look at systems of behavioural differential equations and give a semantic solution in terms of context-free languages (in Sect. 4) and syntactic solutions in terms of regular expressions with unique fixed points (in Sect. 5).

Regular expressions with the Kleene star replaced by a unique right-recursive fixed point operator have been studied in [14, 12] for a large variety of coalgebras, including the one for deterministic automata.

Acknowledgements. We would like to thank Alexandra Silva, for valuable suggestions and discussions.

2 Coalgebras and Deterministic Automata

In this section we give the basic definitions of coalgebras, deterministic automata and context-free grammars. A more extensive coalgebraic treatment of languages, automata and regular expressions can be found, for example, in [10, 9, 5].

A *coalgebra* for an endofunctor $\mathcal{F}{:}\mathbf{Set} \to \mathbf{Set}$ consists of a carrier set C together with a map $c{:}C \to \mathcal{F}C$. The functor $\mathcal{F}$ is usually called the *type* of the coalgebra. In this paper we will be concerned with coalgebras for structured automata [13], i.e. of type $\mathcal{D}\mathcal{F}$, for the functor $\mathcal{D} = 2 \times (-)^A$ and endofunctors $\mathcal{F}{:}\mathbf{Set} \to \mathbf{Set}$. Here and in the rest of this paper, A is a finite set (in this context also called *alphabet*), 2 is the two-element set $\{0, 1\}$ and $\times$ is the Cartesian product. Sometimes we see 2 as a complete lattice with $0 \leq 1$ and join $\vee$ and meet $\wedge$ as expected. A coalgebra $(C, c{:}C \to \mathcal{D}\mathcal{F}X)$ can be interpreted as an automaton that for a given state $t \in C$ returns a pair $c(t) = \langle o(t), \delta(t)\rangle$, determining whether the state t is final (i.e. $o(t) = 1$) or not ($o(t) = 0$), and offering a structured state $\delta(t)(a) \in \mathcal{F}X$ for each input $a \in A$. Typically we will write t_a for $\delta(t)(a)$, call $o(t)$ the *output of* t and t_a the *a-derivative of* t . When confusion may arise about the coalgebra we are currently referring to, we will superscribe $o(t)$ and t_a with the coalgebra map c. We can extend the notion of a-derivative to *word derivatives* t_w, for $w \in A^*$, by setting $t_\lambda = t$ for the empty word λ and $t_{aw} = (t_a)_w$ for $a \in A$ and $w \in A^*$.

A *homomorphism* from a $\mathcal{D}$-coalgebra (C, c) to a $\mathcal{D}$-coalgebra (D, d) is a function $f{:}\,C \to D$ preserving outputs and next states, that is, for all $t \in C$,

$$o(f(t)) = o(t) \quad \text{and} \quad f(t_a) = f(t)_a$$

(which is equivalent to the condition that $d \circ f = \mathcal{D}(f) \circ c$, where the action of the functor $\mathcal{D}$ on functions is as expected).

For example, the set $\mathcal{P}(A^*)$ of all *languages* on the alphabet A can be equipped with a $\mathcal{D}$-coalgebra map by setting for every $L \subseteq A^*$, $o(L) = 1$ if and only if $\lambda \in L$, and $L_a = \{w \in A^* \mid aw \in L\}$. This coalgebra is called *final* because for every $\mathcal{D}$-coalgebra (C, c), there is a unique homomorphism $[\![\ \ ]\!]_c{:}C \to \mathcal{P}(A^*)$ given by

$$\lambda \in [\![t]\!]_c \quad \text{iff} \quad o(t) = 1\,, \quad \text{and} \quad aw \in [\![t]\!]_c \quad \text{iff} \quad w \in [\![t_a]\!]_c\,.$$

A relation $R \subseteq C \times D$ between the carriers of two $\mathcal{D}$-coalgebras (C, c) and (D, d) is called a *bisimulation* if, whenever $(s, t) \in R$, we have $o(s) = o(t)$, and $(s_a, t_a) \in R$ for all $a \in A$. Whenever there exists a bisimulation R such that $(s, t) \in R$, we say that s and t are *bisimilar* and write $s \sim t$. It holds that $s \sim t$ if and only if $[\![s]\!]_c = [\![t]\!]_d$, or, in other words, s and t are bisimilar exactly when they are mapped onto the same language.

A relation $R \subseteq C \times D$ is a *bisimulation up-to* if, whenever $(s, t) \in R$, we have $o(s) = o(t)$, and for all $a \in A$, there are $s' \in C$ and $t' \in D$ such that $s_a \sim s'$, $t_a \sim t'$, and $(s', t') \in R$. Clearly $\sim$ is a bisimulation up-to. Conversely, for every bisimulation up-to R, if $(s, t) \in R$, then $s \sim t$.

3 Context-Free Languages via Grammars

We assume the reader to be familiar with the standard notions on context-free grammars and languages, and give only the definitions and results we need in the rest of this paper. For a more comprehensive study of context-free grammars and languages, see e.g. [8].

A *context-free grammar*, or *CFG*, on a finite alphabet A is a pair (X, p), where X is a finite set of nonterminals, or variables, and $p{:}X \to \mathcal{P}_\omega((A + X)^*)$ is a function describing the production rules. We use the standard notation to describe the production rules:

$$x \to t \quad \text{iff} \quad t \in p(x)\,,$$

where $x \in X$ and $t \in (A{+}X)^*$. Here $+$ denotes the coproduct (or disjoint union), $\mathcal{P}_\omega$ the finite power set, and $(A + X)^*$ is the set of all the strings of finite length over A and X. According to the above definition, CFGs are coalgebras for the functor $\mathcal{P}_\omega((A + (-))^*)$, and indeed a coalgebraic account of context-free grammars and context-free languages using the above functor (without the finiteness condition on the power set) is presented in [3]. There, the focus is mainly on finite skeletal parsed trees (i.e. finite strings with additional tree structure), and context-free languages are obtained after applying a flattening function. In the present paper we will depart from the above work in order to describe uniquely context-free languages in three different (but equivalent) coalgebraic forms.

In order to define the language associated to a context-free grammar, next we define the notion of derivation. Given a CFG $G = (X, p)$, for any string $s, s' \in (A + X)^*$, we write $s \Rightarrow s'$, and say s' is derivable from s in a single *derivation* step, whenever $s = s_1 x s_2$ and $s' = s_1 t s_2$ for a production rule $x \to t$ of G, and $s_2, s_2 \in (A{+}X)^*$. We say that s' is derivable from s in a single *leftmost* *derivation* step whenever s_1 is a (possibly empty) string of terminals in A^*. As usual, $\Rightarrow^*$ denotes the reflexive and transitive closure of $\Rightarrow$. In general, if $s \Rightarrow^* s'$, then s' is derivable from s using only leftmost derivation steps. Therefore we can restrict our attention to leftmost derivations only.

For a CFG (X, p) and any variable $x \in X$, called the starting symbol, we define the language $L(x) \subseteq A^*$ generated by (X, p) from x by

$$L(x) = \{w \in A^* | x \Rightarrow^* w\}.$$

A language $L \subseteq A^*$ is called *context-free* if there exists a CFG (X, p) and a variable $x \in X$, such that $L = L(x)$.

For our coalgebraic treatment of context-free languages it will be convenent to work with CFGs with production rules of a specific form. We say that a CFG is in *weak Greibach normal form* if all of its production rules are of the form

$$x \to at \quad \text{or} \quad x \to \lambda\,,$$

where $a \in A$ is an alphabet symbol, and $t \in (A + X)^*$ is a (possibly empty) sequence of terminal and alphabet symbols. The main difference between weak Greibach normal form and the usual notion of Greibach normal form [2] is that here, t is not a string over X but over $(A + X)$, and hence may contain both terminal and alphabet symbols. Clearly, every CFG in Greibach normal form is also in weak Greibach normal form.

For every terminal symbol x, the language $L(x)$ generated by a CFG (X, p) in weak Greibach normal form is a context-free language. Conversely, every context free language L can be generated by a CFG in Greibach normal form from some terminal symbol [2]. Therefore CFGs in weak Greibach normal form characterize precisely context-free languages.

3.1 A Coalgebraic Treatment of Context-Free Grammars

In this section we look at CFGs in weak Greibach normal form, and, for each such grammar, we define a corresponding $\mathcal{D}$-coalgebra, in the sense that the unique coalgebra homomorphism from the grammar (seen as a $\mathcal{D}$-coalgebra) to the final $\mathcal{D}$-coalgebra of all languages, maps nonterminal symbols precisely to the context-free language they generate. The key observation is that every CFG (X, p) with productions in weak Greibach normal form can be seen as a coalgebra for the functor $\mathcal{D}\mathcal{P}_\omega((A+(-))^*)$ (rather than as a $\mathcal{P}_\omega((A+(-))^*)$-coalgebra as we have seen in the previous section). More precisely, we represent the production rules by a map $p{:}X \to \mathcal{D}\mathcal{P}_\omega((A + (-))^*)$, with $p(x) = \langle o(x), \delta(x)\rangle$ defined, for all terminal symbols $x \in X$, by

$$o(x) = 1 \quad \text{iff} \quad x \to \lambda\,, \quad \text{and} \quad t \in x_a \quad \text{iff} \quad x \to at$$

(writing as before x_a for $f(a)$). Consider for example the grammar (in weak Greibach normal form) over the alphabet $A = \{a, b\}$ with nonterminal symbols $X = \{x, y\}$ and productions

$$x \to axa\,, \quad x \to ayb\,, \quad x \to aa\,, \quad y \to ayb\,, \quad y \to \lambda\,.$$

The language generated from x is $L(x) = \{a^n b^m a^k | n = m + k, n \geq 1\}$, while the language generated from y is $L(y) = \{a^n b^n | n \geq 0\}$. In coalgebraic form, the above productions read as follows:

$$\begin{array}{lll} o(x) = 0 & x_a = \{xa, yb, a\} & x_b = \emptyset \\ o(y) = 1 & y_a = \{yb\} & y_b = \emptyset\,. \end{array}$$

The coalgebra associated to each CFG in weak Greibach normal form is not a proper deterministic automaton (i.e. a $\mathcal{D}$-coalgebra) because its type is of the form $\mathcal{D}\mathcal{P}_\omega((A + (-))^*)$. However, using a technique analogous to determinization [13], we can turn it into a deterministic automaton by embedding the nonterminal symbols X into $\mathcal{P}_\omega((A+X)^*)$ using the assignment $\eta_X{:}X \to \mathcal{P}_\omega((A+X)^*)$, mapping each $x \in X$ into the singleton set $\{x\}$ (in which x is seen as a string). In fact, we extend in a canonical manner each coalgebra $p{:}X \to \mathcal{D}\mathcal{P}_\omega((A+X)^*)$ for a CFG to what we call a *grammar coalgebra* $p^\#{:}\mathcal{P}_\omega((A+X)^*) \to \mathcal{D}\mathcal{P}_\omega((A+X)^*)$ as follows: for each finite subset $S \subseteq (A+X)^*$ we define its output value and its a-derivative by

S	$o(S)$	S_a	
$\emptyset$	0	$\emptyset$	
$\{\lambda\}$	1	$\emptyset$	
$\{as\}$	0	$\{s\}$	where $t \in (A+X)^*$
$\{bs\}$	0	$\emptyset$	if $b \neq a, b \in A$
$\{xs\}$	$o(s)$	$\{ts \,\vert\, t \in x_a\} \cup \{s_a\}$	if $o(x) = 1, x \in X, t \in (A+X)^*$
$\{xs\}$	0	$\{ts \,\vert\, t \in x_a\}$	if $o(x) = 0, t \in (A+X)^*$
$T \cup U$	$o(T) \vee o(U)$	$T_a \cup U_a$	where $T, U \subseteq (A+X)^*$.

The idea of this definition is to view subsets of $(A+X)^*$ as languages. In fact, the above definition coincides with the coalgebra structure map of the final $\mathcal{D}$-coalgebra $\mathcal{P}(A^*)$ if we take subsets of strings in A^* (not containing nonterminal symbols). This is combined with the coalgebra map of the grammar which gives the output value and a-derivative for each nonterminal symbol, as can be seen from the fact that $p^\# \circ \eta_X = p$.

$$\begin{tikzcd} X \arrow[r, "\eta_X"'] \arrow[d, "p"] & \mathcal{P}_\omega((A+X)^*) \arrow[r, "{[\![\ ]\!]}"'] \arrow[dl, "p^\#"] & \mathcal{P}(A^*) \arrow[d] \\ \mathcal{D}\mathcal{P}_\omega((A+X)^*) \arrow[rr] & & \mathcal{D}\mathcal{P}(A^*) \end{tikzcd}$$

We are now ready to state our main result for this section, namely the correspondence between context-free languages and the languages associated by the final homomorphism $[\![-]\!]$ above to each nonterminal symbol of a CFG.

Theorem 1. *Let (X, p) be a context-free grammar in weak Greibach normal form over a finite alphabet A, and S a finite subset of $(A+X)^*$. For every word $w \in A^*$, we have $w \in [\![S]\!]_{p^\#}$ if and only if there exists some $s \in S$ such that $s \Rightarrow^* w$.*

(For a sketch of the proof of this theorem, and some of the later theorems, see the Appendix.)

It follows that a language L is context-free iff $L = [\![\eta_X(x)]\!]_{p^\#}$, for some grammar coalgebra generated by a CFG (X, p) and $x \in X$.

4 Context-Free Languages via Equations

Next we will look at a characterization of context-free languages in terms of systems of behavioural differential equations, analogous to those introduced in [11]. The idea is to define context-free languages by means of equations that involve output values and derivatives for each alphabet symbol, using a simple language with only variables, choice and sequential composition. Each system of behavioural differential equations in our format has as its unique solution a language, which we will prove to be context-free whenever the number of equations is finite. Conversely, for each context-free language L we will construct a finite system of behavioural differential equations with L as unique solution.

To illustrate our approach, consider the example from the previous section. A formal definition of this context-free language could be given by the following system of behavioural differential equations:

output value	a-derivative	b-derivative
$o(x) = 0$	$x_a = (x \cdot a) + (y \cdot b) + a$	$y_a = y \cdot b$
$o(y) = 1$	$x_b = 0$	$y_b = 0$

Next we present a syntax describing the format of behavioural differential equations that will be considered: let A be a finite set of *alphabet* symbols, X be a (possibly infinite) set of *variables*, and $\{o(x) | x \in X\}$ and $\{x_a | x \in X\}$ for each $a \in A$ be (syntactic) sets of symbols, representing notational variants of the variables. The variables $x \in X$ will play the role of placeholders for languages $L \subseteq A^*$, while their notational variants x_a will be placeholders for the corresponding language L_a, for each $a \in A$, and the $o(x)$ will correspond to the information whether L contains the empty string λ or not. We call a behavioural differential equation for context-free languages *well-formed* if it consists of an equation $o(x) = v$ and an equation $x_a = t$ for each $a \in A$, where $v \in \{0, 1\}$ and t is a term defined by

$$t ::= 0 \,|\, 1 \,|\, x \,|\, a \,|\, t + t \,|\, t \cdot t$$

where $x \in X$ and $a \in A$. (In the remainder of this paper, we will often simply write 'ab' rather than '$a \cdot b$'.) We let TX denote the set of all terms, as defined above, over the set X. A *well-formed system of equations* for X consists of one well-formed equation for each $x \in X$. Equivalently, a well-formed system of equations over X (for a fixed A) can be seen as a mapping $f{:}X \to \mathcal{D}(TX)$ where, writing $f(x) = \langle o(x), \delta(x) \rangle$, we define $o(x)$ and, for each $a \in A$, $x_a = \delta(x)(a)$ by the values specified by the system of behavioural differential equations.

Before defining what a solution of a system of equations is, we need to interpret the above operations on terms as functions on languages. To this end, we transform a system of equations $f{:}X \to \mathcal{D}(TX)$ into a deterministic automaton $\bar{f}{:}TX \to \mathcal{D}(TX)$ inductively as follows:

t	$o(t)$	t_a	
0	0	0	
1	1	0	
x	$o(x)$	x_a	
a	0	1	
b	0	0	if $b \neq a, b \in A$
$u+v$	$o(u) \vee o(v)$	$u_a + v_a$	$u, v \in TX$
$u \cdot v$	0	$u_a \cdot v$	if $o(u) = 0, u, v \in TX$
$u \cdot v$	$o(v)$	$u_a \cdot v + v_a$	if $o(u) = 1, u, v \in TX$

We can now combine a system of equations $f{:}X \to \mathcal{D}(TX)$, its extension $\bar{f}$, and the unique homomorphism into the final coalgebra of all languages in the following diagram:

$$\begin{tikzcd}
X \arrow[r, hook, "i"'] \arrow[d, "f"] & TX \arrow[r, "{[\![\ ]\!]}"'] \arrow[dl, "\bar{f}"] & \mathcal{P}(A^*) \arrow[d] \\
\mathcal{D}(TX) \arrow[rr] & & \mathcal{D}\mathcal{P}(A^*)
\end{tikzcd}$$

A *solution* for such a system of equations consists of a mapping of variables x to languages $L_x \subseteq A^*$ such that $L_x = [\![x]\!]$ for all $x \in X$. The above diagram basically shows that every well-formed system of equations has a unique solution. Our next goal is to prove that such a solution is a context-free language. Before we do this, however, we introduce the notion of term equivalence:

Definition 2. *We say that two terms $t, u \in TX$ are* equivalent*, denoted by $t \equiv u$, when for every well-formed system of equations (X, f), $t \sim u$ with respect to the coalgebra $(TX, \bar{f})$ generated from (X, f).*

One can easily show that the relation $\equiv$ is a congruence with respect to the sum $+$ and multiplication $\cdot$ of terms in TX, for any set X. Furthermore, TX modulo $\equiv$ forms an idempotent semiring:

Proposition 3. *For any set X and terms $t, u, v \in TX$, the following hold:*

$$\begin{array}{ll}
t + u \equiv u + t & u \equiv v \Rightarrow t[u/x] \equiv t[v/x] \\
t + t \equiv t & 0 \cdot t \equiv 0 \equiv t \cdot 0 \\
0 + t \equiv t \equiv t + 0 & 1 \cdot t \equiv t \equiv t \cdot 1 \\
t + (u + v) \equiv (t + u) + v & t \cdot (u \cdot v) \equiv (t \cdot u) \cdot v \\
(t + u) \cdot v \equiv t \cdot v + u \cdot v & t \cdot (u + v) \equiv t \cdot u + t \cdot v
\end{array}$$

Equivalence between terms in TX can be extended to bisimilarity between different systems of equations on the same set of variables. This result will be convenient when proving that every context-free language is the solution of a well-formed system of equations.

Proposition 4. *If (X, f) and (X, g) are two systems of equations such that, for every $x \in X$, and every symbol a, $o^f(x) = o^g(x)$, and $x^f_a \equiv x^g_a$, then the identity relation on TX is a bisimulation up-to between the generated coalgebras $(TX, \bar{f})$ and $(TX, \bar{g})$.*

We can now establish the first main result of this section, relating states of a grammar coalgebra to terms in TX.

Theorem 5. *Let A and X be finite sets. For every context-free grammar (X, p) in weak Greibach normal form and finite subset S of $(A + X)^*$ there exists a well-formed system of equations $f{:}X \to (TX)^A$ and a term $t \in TX$ such that $S \sim t$ with respect to the generated coalgebras $p^\#$ and $\bar{f}$, respectively.*

Proof. (Sketch) The system of equations f can be constructed from the grammar p as follows: $o(x) = 1$ if $\lambda \in p(x)$, and 0 otherwise; and $x_a = \sum\{\bar{s} \mid as \in p(x)\}$, where $\bar{s}$ is the obvious translation of $s \in (A+X)^*$ into a term of TX. The proof now proceeds by showing that $\{(S, t) \mid S \subseteq (A + X)^*, t = \sum\{\bar{s} \mid s \in S\}\}$ is a bisimulation up-to with respect to the generated coalgebras. □

Because for every context-free language L there exists a CFG (X, p) and $x \in X$ such that $L = [\![\eta_X(x)]\!]_{p^\#}$, it follows that *every context-free language is the solution of a well formed system of equations.*

It remains to show that every solution of a system of equations is a context-free language. Our approach will be to construct, for every system of equations (X, f), a context-free grammar (X, p), such that $x \sim \eta_X(x)$ for all x with respect to the generated coalgebras $\bar{f}$ and $p^\#$. To this end, we first transform our system of equations so that terms at the right hand side of all equations are in disjunctive normal form. This is possible because of the laws proven in Propositions 3 and because $\equiv$ is a congruence.

We say that a term $t \in TX$ is *conjunctive* when either $t = 1$; or when $t = a \cdot u$, where $a \in A$ and u a conjunctive term; or when $t = x \cdot u$, where x is a variable, and u is a conjunctive term. We say that a term $t \in TX$ is in *disjunctive normal form*, when either $t = 0$; or when $t = u + v$, where u is conjunctive, and v is in disjunctive normal form. Using Proposition 3, it is easy to see that for every term t, there is an equivalent (and thus bisimilar for all $(TX, \bar{f})$-coalgebras) term t' in disjunctive normal form. This implies that for every system of equations we can construct a new system of equations with the same variables and having the same solution but such that all terms on the right-hand side of the a-differential equations are in disjunctive normal form. We call a system of equations with the latter property a *system of equations in disjunctive normal form.*

We are finally ready for the other main result of this section, stating that *every solution of a system of equations is a context-free language.*

Theorem 6. *For a finite set X, if (X, f) is a system of equations in disjunctive normal form there exists a context-free grammar (X, p), such that $x \sim \eta_X(x)$ with respect to the generated coalgebras $(TX, \bar{f})$ and $(\mathcal{P}((A + X)^*), p^\#)$, respectively.*

Proof. Given a system of equations (X, f), we construct the grammar (X, p), such that

$$p(x) = \bigcup\{at \,|\, t \text{ is a disjunct of } x_a\} \cup \{\lambda \,|\, \text{if } o(x) = 1\}\,.$$

It is easy to see that f is a translation of p in the sense of Theorem 5. Hence, it follows that $x \sim \{x\} = \eta_X(x)$ with respect to $\bar{f}$ and $p^{\#}$. $\square$

Combining the two theorems in this section we have obtained that context-free languages are precisely the solutions of well-formed systems of equations.

5 Context-Free Expressions

In this section, we will introduce context-free expressions as an extension of regular expressions, where the Kleene star is replaced by a (unique) fixed point operator μ. We then will define a notion of Brzozowski-like derivatives for these expressions, and prove that the languages characterizable by such expressions are precisely the context-free languages. In contrast to the previous coalgebraic formalisms, this formalism gives us a *single* coalgebra of which the elements correspond exactly to the context-free languages.

Our usage of fixed point expressions with a coinductive semantics has a very similar flavour to that in [12], in which fixed point expressions are used as a characterization of regular expressions over a variety of functors. The additional expressive power obtained by the context-free expressions presented here is due to an explicit inclusion of a concatenation operator.[4] This provides an additional perspective on the treatment given here, in which 'context-freeness' is obtained by the addition of a new operator to a calculus of regular experssions[5], and may pave the way for an investigation of (1) extending this approach to other coinductively defined operators, and (2) extending this approach to a generalized notion of context-freeness for other functors.

We define the set of terms t (henceforth to be called *context-free expressions*) and guarded terms g over an alphabet A and a set of variables X as follows:

$$\begin{aligned} t &::= 0 \,|\, 1 \,|\, x \in X \,|\, a \in A \,|\, t + t \,|\, t \cdot t \,|\, \mu x.g \\ g &::= a \cdot t \, (a \in A) \,|\, 1 \,|\, g + g \end{aligned}$$

For all closed terms t, we can describe the behaviour by defining the output value $o(t)$, and the derivative t_a for each alphabet symbol a. We do this as follows:

[4] In [12], a translation from the familiar format of regular expressions (with concatenation) into μ-style expressions is given by means of substitution. However, this translation does not work for expressions of the type $x \cdot t$.

[5] Although this calculus does not explicitly contain the Kleene star, it can easily be expressed by means of the equality $t^* = \mu x.((t \cdot x) + 1)$

t	$o(t)$	t_a	
0	0	0	
1	1	0	
a	0	1	
b	0	0	if $b \neq a, b \in A$
$u+v$	$o(u) \vee o(v)$	$u_a + v_a$	
$u \cdot v$	0	$u_a \cdot v$	if $o(u) = 0$
$u \cdot v$	$o(v)$	$u_a \cdot v + v_a$	if $o(u) = 1$
$\mu x.u$	$o(u[\mu x.u/x])$	$(u[\mu x.u/x])_a$	

Here $t[u/x]$, as usual, denotes the term obtained from t by replacing all *free* occurrences of x by u. Because of the guardedness conditions of terms occurring directly inside the μ operator, it is easy to see that the above definition is well-defined.

Note furthermore that we have just defined a $\mathcal{D}$-coalgebra – the term coalgebra – with the set of all context-free expressions as objects, and the behaviour defined above as its transition function.

Again, the relation $\sim$ is a congruence with respect to the sum $+$ and multiplication $\cdot$ of context-free expressions. Furthermore, the set of context-free expressions modulo $\sim$ forms an idempotent semiring:

Proposition 7. *For all context-free expressions t, u, v, the following hold:*

$$\begin{array}{ll}
t+u \sim u+t & u \sim v \Rightarrow t[u/x] \sim t[v/x] \\
t+t \sim t & 0 \cdot t \sim 0 \sim t \cdot 0 \\
0+t \sim t \sim t+0 & 1 \cdot t \sim t \sim t \cdot 1 \\
t+(u+v) \sim (t+u)+v & t \cdot (u \cdot v) \sim (t \cdot u) \cdot v \\
(t+u) \cdot v \sim t \cdot v + u \cdot v & t \cdot (u+v) \sim t \cdot u + t \cdot v \\
t[u/x] \sim u \Rightarrow \mu x.t \sim u & \mu x.t \sim \mu y.(t[y/x]) \textit{ if } y \textit{ is not free in } t \\
\mu x.t \sim t[\mu x.t/x] &
\end{array}$$

As an illustration of context-free expressions, it is easy to see that the expression $\mu x.(axb+1)$ will be mapped onto the language $\{a^n b^n\}$. As another example, consider the expression

$$\mu x.(axa + aa + a\mu y.(ayb+1)b).$$

In the next subsection, it will become clear that this expression corresponds to the language $\{a^n b^m a^k \mid n = m+k, n \geq 1\}$ from the earlier examples.

5.1 From Systems of Equations to Context-Free Expressions

Assume we have a coalgebra generated by a system of equations, and an term in this coalgebra. From Sect. 4, we know that this term is mapped by the final homomorphism onto a context-free language. In this section, we will embark on the task of finding a context-free expression corresponding to this term, in the sense that it is mapped onto the same language. We will do this using a process of repeated substitution.

To start with, given a system of equations (X, f), to we will associate with every variable x the μ-expression

$$s_x := \mu x.(\sum_{a\in A} a \cdot x_a + o(x)),$$

and call it the *corresponding* or *associated* μ-expression. (As before, this notation strictly speaking does not denote a single expression, but rather a set of expressions which, by associativity of addition, are all bisimilar.)

We now go on by defining the notions of *single syntactic substitutions* and *chains of syntactic substitutions*: a term t' is a single syntactic substitution of t, if t' is obtained by replacing (syntactically) a *single* occurrence of a single variable by its corresponding μ-expression. A chain of syntactic substitutions is a list of terms $t_1, \dots, t_n$ such that, for each $1 \leq i < n$, t_{i+1} is a single syntactic substitution of t_i.

We are especially interested in chains of syntactic substitutions, where the resulting term does not contain any free variables, or only a limited set of free variables. We will call such terms closures and pseudoclosures of the original term:

Definition 8. *We say a term t' is a Z-*pseudoclosure *of t for a set $Z \subseteq X$ of variables, if there exists a a chain of syntactic substitutions $t_1, \dots, t_n$ such that $t_1 = t$, $t_n = t'$ and t' only contains free variables from Z. We call a $\emptyset$-pseudoclosure simply a* closure*.*

As a continuation of our running example, recall the system of equations corresponding to the language $\{a^n b^m a^k \mid n = m + k, n \geq 1\}$. From this system of equations, we obtain the following assignment of μ-expressions to variables:

$$s_x = \mu x.(axa + ayb + aa + 0)$$
$$s_y = \mu y.(ayb + 1)$$

From x, we obtain $\mu x.(axa + ayb + aa + 0)$ by means of a single syntactic substitutions, and another single syntatctic substitution then gives us $\mu x.(axa + a\mu y.(ayb + 1)b + aa + 0)$. This expression does not contain any free variables anymore, and therefore is a closure of x.

Some general laws about closures and psueudoclosures are easily established:

Proposition 9. *1. If u' is a W-pseudoclosure of u and v' a W-pseudoclosure of v, then $u' + v'$ is a W-pseudoclosure of $u + v$, $u' \cdot v'$ is a W-pseudoclosure of $u \cdot v$, and $\mu x.u'$ is a $W - \{x\}$-pseudoclosure of $\mu x.u$.*

2. If $t = u + v$, and t' is a W-pseudoclosure of t, then t' is of the form $u' + v'$, where u' is a W-pseudoclosure of u, and v' is a W-pseudoclosure of v'. The same fact holds if we replace $+$ by $\cdot$.

Using the previous proposition, we can establish that, for every term t, a closure t' exists. It should be noted, though, that this t' generally is not unique: for a term t, in general, many closures exist.

Proposition 10. *Given a term $t \in TX$ (that is, a μ-free term), a set of variables $Z \subseteq X$, and an assignment of a term t_x to each variable $x \in X$, there exists a Z-pseudoclosure t' of t.*

With the next proposition we construct a bisimulation up-to between a coalgebra generated by a system of equations and the term coalgebra, relating every term t to every t' such that t' is a closure of t.

Proposition 11. *Given a system of equations (X, f) (yielding an corresponding expression s_x for each variable $x \in X$), a coalgebra generated by it $(TX, \bar{f})$, and a assignment of a term u_x to each variable $x \in X$ such that $u_x \equiv s_x$ for all $x \in X$, the relation*

$$R = \{(t, t') \mid t \in TX \text{ and } t' \text{ is a closure of } t \text{ (w.r.t. the assignment } u_x)\}$$

is a bisimulation up-to between $(TX, \bar{f})$ and the term coalgebra.

Going back once again to our example, this proposition directly establishes that the expression

$$\mu x.(axa + a\mu y.(ayb + 1)b + aa + 0)$$

corresponds to the language $\{a^n b^m a^k \mid n = m + k, n \geq 1\}$. The earlier expression $\mu x.(axa + a\mu y.(ayb + 1)b + aa)$ is easily seen to be bisimilar to this expression.

The two previous propositions directly imply that, for any term in a coalgebra generated by a system of equations (and, hence, for every context-free language), we can use any closure of it as a bisimilar context-free expression. Hence, and because every term has a closure, for every context-free language we can find a context-free expression that is mapped to it by the final homomorphism:

Theorem 12. *Let L be a context-free language. There exists a context-free expression t such that $[\![t]\!] = L$.*

5.2 From Context-Free Expressions to Systems of Equations

Going in the other direction, the recipe is as follows: given a context-free expression t' in which every variable is bound by a μ-operator just once, we 'deconstruct' this expression into a system of equations, and a term t, of which t' is a closure. Then Proposition 11 directly gives us the result, that there is a system of equations (X, f), a variable $x \in X$, such that $t' \sim t$ with respect to the term coalgebra and the coalgebra $(TX, \bar{f})$ generated by (X, f). Hence, it follows that t is mapped by the final homomorphism onto a context-free language.

By applying a process of α-renaming, we can obtain an expression t' from any expression t such that, in t', no variable is bound twice, or, in other words, such that there are no two distinct subexpressions of t' that bind the same variable. It is easy to see that the resulting term t' will always be bisimilar to t.

Now we are able to move on to the main proposition of this section, in which for every context-free expression t' a system of equations is constructed, such that in the coalgebra generated by it, there is a term t with $t \sim t'$.

Proposition 13. *Given a context-free expression t', such that no two distinct sub-expressions of t' are μ-expressions binding the same variable, there exists a system of equations (X, f) generating an assignment of expressions s_x to variables $x \in X$, and a term t, such that (with respect to (X, f)) t' is a closure of t, based on an assignment of expressions t_x to variables $x \in X$ with $t_x \equiv s_x$.*

As every context-free expression is bisimilar to a term in a coalgebra generated by some system of equations, it follows directly that the final homomorphism maps every context-free expression to a context-free language:

Theorem 14. *For every context-free expression t, $[\![t]\!]$ is a context-free language.*

6 Discussion

Our coalgebraic account of context-free languages in terms of grammars, systems of behavioural equations, and context-free expressions can be taken as a starting point for a generalization in at least two different and orthogonal directions. On the one hand, we can consider other languages of expressions for the functor $\mathcal{D}$ to obtain different classes of languages, and on the other hand we can generalize the notion of context-freeness to coalgebras for other functors .

As an interesting example of the first type, one could consider systems of behavioural differential equations for which the term at the right of each equation stems from the language of expressions

$$t ::= 0 \,|\, 1 \,|\, x \in X \;|\, t + t \,|\, a \cdot t\,.$$

The semantic solution of such a system is given by regular languages, while the syntactic one is given by a language of expressions as studied in [14]. The corresponding notion of grammars for regular languages is then given by considering productions of the form $p{:}X \to \mathcal{D}(\mathcal{P}_\omega(A^* \times X))$, i.e. *right-linear grammars* [4].

Examples of the second type of generalization will depend on the structure of the functor. Here we briefly sketch only an elementary but interesting first example. The functor $\mathcal{S}(X) = \mathbb{N} \times X$ has the set of all *streams* $\mathbb{N}^\omega$ as its final coalgebra. We note that, similar to the set of all languages, also $\mathbb{N}^\omega$ is a semiring, with elementwise addition of streams as sum, and convolution product as product. Streams can be defined by behavioural differential equations, which specify the head $\sigma(0) \in \mathbb{N}$ and tail σ' of a stream $\sigma \in \mathbb{N}^\omega$. Now let us call a stream context-free whenever it can be specified by a stream differential equation $\sigma' = t$, where t is a term of the following form:

$$t ::= n \,|\, \mathcal{X} \,|\, t + t \,|\, t \times t$$

and where n is any natural number, $\mathcal{X}$ denotes the constant stream $(0, 1, 0, 0, \ldots)$, $+$ denotes elementwise addition of streams, and $\times$ denotes convolution product. We note that the above syntax (and its semantics) for t is a straightforward variation on the syntax (and its semantics) that we used for the behavioural differential equations for context-free languages, in Section 4.

As an example of a context-free stream, let γ be defined by the well-formed differential equation given by

$$\gamma' = \gamma \times \gamma \qquad \gamma(0) = 1$$

It has as its unique solution the stream of the Catalan numbers $(1, 1, 2, 5, 14, \ldots)$, which occurs in numerous counting problems (such as the number of well-bracketed words consisting of matching pairs of an opening and a closing bracket). The stream γ is known not to be rational, so this example nicely illustrates how the class of context-free streams extends the class of rational streams, in the same way as with languages.

Further research directions include a coalgebraic characterization of context-free languages in terms of pushdown automata [4, 8], and the study of coinductive decision procedures for bisimilarity of deterministic context-free languages, a problem that is known to be decidable [15].

References

1. Ésik, Z., Leiß, H.: Algebraically complete semirings and Greibach normal form. Annals of Pure and Applied Logic 133(1-3), 173–203 (2005)
2. Greibach, S.A.: A new normal-form theorem for context-free, phrase structure grammars. Journal of the Association for Computing Machinery 12, 42–52 (1965)
3. Hasuo, I., Jacobs, B.: Context-free languages via coalgebraic trace semantics. In: CALCO. LNCS, vol. 3629, pp. 213–231. Springer (2005)
4. Hopcroft, J.E., Ullman, J.D.: Formal languages and their relation to automata. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1969)
5. Jacobs, B.: A bialgebraic review of deterministic automata, regular expressions and languages. In: Essays Dedicated to Joseph A. Goguen. LNCS, vol. 4060, pp. 375–404. Springer (2006)
6. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. Inf. Comput. 110(2), 366–390 (1994)
7. Leiß, H.: Towards Kleene algebra with recursion. In: CSL. LNCS, vol. 626, pp. 242–256. Springer (1991)
8. Linz, P.: An Introduction to Formal Languages and Automata. Jones and Bartlett (1997)
9. Rutten, J.J.M.M.: Automata and coinduction (an exercise in coalgebra). In: CONCUR. LNCS, vol. 1466, pp. 194–218. Springer (1998)
10. Rutten, J.J.M.M.: Universal coalgebra: a theory of systems. Theoretical Computer Science 249(1), 3–80 (2000)
11. Rutten, J.J.M.M.: Behavioural differential equations: a coinductive calculus of streams, automata, and power series. Theoretical Computer Science 308(1-3), 1–53 (2003)
12. Silva, A.: Kleene Coalgebra. Ph.D. thesis, Radboud Universiteit Nijmegen (2010)
13. Silva, A., Bonchi, F., Bonsangue, M.M., Rutten, J.J.M.M.: Generalizing the powerset construction, coalgebraically. In: FSTTCS. LIPIcs, vol. 8, pp. 272–283. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2010)
14. Silva, A., Bonsangue, M.M., Rutten, J.J.M.M.: Non-deterministic Kleene coalgebras. Logical Methods in Computer Science 6(3) (2010)
15. Stirling, C.: Decidability of DPDA equivalence. Theoretical Computer Science 255(1-2), 1–31 (2001)

A Appendix: Proofs

Proof (of Proposition 1). In this proof, two facts that are easily verified by induction will be used: 1) $o(S_w) = 1$ iff $w \in [\![S]\!]$ for all words w and all S; and 2) for all $S \subseteq (A + X)^*$, if $w \in [\![S]\!]$, then there is a $s \in S$ such that $w \in [\![\{s\}]\!]$.

We proceed by induction on the length of words w. For the empty word λ, it is easy to see that $s \Rightarrow^* \lambda$ if and only if s only consists of nonterminal symbols x that have a production rule $x \to \lambda$. Conversely, we have $\lambda \in [\![S]\!]$ iff $o(S_\lambda) = 1$, that is, if $o(S) = 1$: but it follows easily from the definition that this is the case iff there is a $s \in S$ consisting only of nonterminal symbols x that have a production rule $x \to \lambda$.

Assume that the inductive hypothesis holds for w, and consider the word aw. Assume $s \Rightarrow^* aw$, and consider the first term in the leftmost derivation of aw that is of the form at. Thus $s \Rightarrow^* at \Rightarrow^* aw$. By inspecting the definitions, it is easily seen that if $s \in S$, then $t \in S_a$. Furthermore, it also follows that $t \Rightarrow^* w$, and the inductive hypothesis then gives $w \in [\![S_a]\!]$, and hence that $o((S_a)_w) = 1$. But clearly $S_{aw} = (S_a)_w$, so $aw \in [\![S]\!]$.

For the other direction, assume that $aw \in [\![S]\!]$. Then there must be some $s \in S$, such that $aw \in [\![\{s\}]\!]$, or that $o(\{s\}_{aw}) = o((\{s\}_a)_w) = 1$. We now get that $w \in [\![\{s\}_a]\!]$, and the inductive hypothesis now gives some $t \in \{s\}_a$ such that $t \Rightarrow^* w$. From inspecting the definitions, it is also easy to see that $s \Rightarrow^* at$. Hence, we get $s \Rightarrow^* at \Rightarrow^* aw$, which is what needed to be shown. □

Proof (of Proposition 3). As two useful illustrations of coinductive proofs, we will show the proofs that $t \cdot (u + v) \equiv t \cdot u + t \cdot v$ and $t \cdot (u \cdot v) \equiv (t \cdot u) \cdot v$:

To see that $t \cdot (u + v) \equiv t \cdot u + t \cdot v$, consider the relation

$$R = \{(t \cdot (u + v), t \cdot u + t \cdot v)\} \cup \{((t \cdot (u + v)) + w, (t \cdot u + t \cdot v) + w)\}.$$

We will show that this is a bisimulation up-to. Assume $r \in R$. If r is of the form $(t \cdot (u + v), t \cdot u + t \cdot v)$, we have $o(t \cdot (u + v)) = o(t) \wedge (o(u) \vee o(v)) = (o(t) \wedge o(u)) \vee (o(t) \wedge o(v)) = o(t \cdot u + t \cdot v)$. Furthermore, whenever $o(t) = 0$, we have:

$$\begin{aligned} (t \cdot (u + v))_a &= t_a \cdot (u + v) \\ (t \cdot u + t \cdot v)_a &= t_a \cdot u + t_a \cdot v \end{aligned}$$

and clearly $(t_a \cdot (u + v), t_a \cdot u + t_a \cdot v) \in R$. When $o(t) = 1$, we have:

$$\begin{aligned} (t \cdot (u + v))_a &= t_a \cdot (u + v) + (u + v)_a \\ &= t_a \cdot (u + v) + (u_a + v_a) \end{aligned}$$

and

$$\begin{aligned} (t \cdot u + t \cdot v)_a &= (t \cdot u)_a + (t \cdot v)_a \\ &= (t_a \cdot u + u_a) + (t_a \cdot v + v_a). \end{aligned}$$

But clearly, we have, by earlier laws

$$(t_a \cdot u + u_a) + (t_a \cdot v + v_a) \sim (t_a \cdot u + t_a \cdot v) + (u_a + v_a).$$

As we know that $(t_a \cdot (u+v) + (u_a + v_a), (t_a \cdot u + t_a \cdot v) + (u_a + v_a)) \in R$, we know that the bisimulation up-to condition is fulfilled. The case where r is of the form $((t \cdot (u+v)) + w, (t \cdot u + t \cdot v) + w)$ goes very similarly.

To see that $t \cdot (u \cdot v) \equiv (t \cdot u) \cdot v$, consider the relation

$$R = \{t \cdot (u \cdot v), (t \cdot u) \cdot v\} \cup \{t \cdot (u \cdot v) + w, (t \cdot u) \cdot v + w\}.$$

Again, we will show that this is a bisimulation up-to. Again, we will treat the case where we have a $r \in R$ of the form $(t \cdot (u \cdot v), (t \cdot u) \cdot v)$ – the other case goes very similarly, and now we distinguish three cases.

If $o(t) = 0$, we will necessarily also have $o(t \cdot u) = 0$, and $(t \cdot (u \cdot v))_a = t_a \cdot (u \cdot v)$, as well as $((t \cdot u) \cdot v)_a = (t \cdot u)_a \cdot v = (t_a \cdot u) \cdot v$. Clearly $(t_a \cdot (u \cdot v), (t_a \cdot u) \cdot v) \in R$.

If $o(t) = 1$ and $o(u) = 0$, we have

$$\begin{aligned}(t \cdot (u \cdot v))_a &= t_a \cdot (u \cdot v) + (u \cdot v)_a \\ &= t_a \cdot (u \cdot v) + u_a \cdot v.\end{aligned}$$

On the other side,

$$\begin{aligned}((t \cdot u) \cdot v)_a &= (t \cdot u)_a \cdot v \\ &= (t_a \cdot u + u_a) \cdot v\end{aligned}$$

However, we already know that $(t_a \cdot u + u_a) \cdot v \sim (t_a \cdot u) \cdot v + u_a \cdot v$, and now because $(t_a \cdot (u \cdot v) + u_a \cdot v, (t_a \cdot u) \cdot v + u_a \cdot v) \in R$ the bisimulation up-to condition is satisfied.

Finally, if $o(t) = o(u) = 1$, we have

$$\begin{aligned}(t \cdot (u \cdot v))_a &= t_a \cdot (u \cdot v) + (u \cdot v)_a \\ &= t_a \cdot (u \cdot v) + (u_a \cdot v + v_a)\end{aligned}$$

and

$$\begin{aligned}((t \cdot u) \cdot v)_a &= (t \cdot u)_a \cdot v + v_a \\ &= (t_a \cdot u + u_a) \cdot v + v_a.\end{aligned}$$

But again,

$$(t_a \cdot u + u_a) \cdot v + v_a \sim ((t_a \cdot u) \cdot v + u_a \cdot v) + v_a$$

and

$$((t_a \cdot u) \cdot v + u_a \cdot v) + v_a \sim (t_a \cdot u) \cdot v + (u_a \cdot v + v_a).$$

As $(t_a \cdot (u \cdot v) + (u_a \cdot v + v_a), (t_a \cdot u) \cdot v + (u_a \cdot v + v_a)) \in R$, this completes the proof that R is a bisimulation up-to. □

Proof (of Proposition 4). Let t be a term.

First, we have to show that $o^{\bar{f}}(x) = o^{\bar{g}}(x)$. When t is a variable, an alphabet symbol, 0 or 1, this is trivial, and when t is a compound term, this is proven by induction.

Secondly, we have to show that for all alphabet symbols a, there are terms t', t'', such that $t_a^{\bar{f}} \sim t'$, $t_a^{\bar{g}} \sim t''$, and $(t', t'') \in R$ (or, in other words, $t' = t''$).

This, too, will be proven by induction, and is trivial for the base cases where t is an alphabet symbol, 0, or 1.

When t is a variable x, we have $x_a^{\bar{f}} = x_a^f \sim x_a^g = x_a^{\bar{g}}$. Taking $t' = t'' = x_a^{\bar{f}}$ then suffices.

When t is of the form $u + v$ or of the form $u \cdot v$, we will make use of the inductive assumption that the bisimilarity condition holds for u and v. But then it is easy to see that it holds for t too, making use of the fact that $\sim$ is a congruence with respect to $+$ and $\cdot$. □

Proof (of Proposition 10). By (reverse) induction on the size of Z. If $Z = X$, the result is trivial, because every term is its own X-pseudoclosure.

Now we assume the theorem holds for $W \subseteq X$, and need to prove that, for any $x \in X$, the theorem also holds for $W - \{x\}$. We do this by induction on (μ-free) terms.

1. For terms 0, 1, and a, the result is trivial as these terms do not contain any free variables.
2. For terms $t = u + v$ or $t = u \cdot v$, the result follows from Proposition 9 and the inductive hypothesis.
3. For the variable x, we know that there must be a W-pseudoclosure u of t_x. Then $\mu x.u$ is a $(W - \{x\})$-pseudoclosure of x.
4. For variables $y \neq x$, assume that u is a W-pseudoclosure of t_y, and v is a W-pseudoclosure of t_x. Then $u[\mu x.v/x]$ is a $W - \{x\}$-pseudoclosure of t_y (it is easy to see that $u[\mu x.v/x]$ can be obtained from u by a chain of single syntactic substitutions), and hence $\mu y.u[\mu x.v/x]$ is a $W - \{x\}$-pseudoclosure of y. □

Proof (of Proposition 11). Say $(t, t') \in R$. It suffices to show that $o(t) = o(t')$, and that for each alphabet symbol a, there is a s_a, such that $t'_a \sim s_a$, and $(t_a, s_a) \in R$. We will do both by induction.

Showing that $o(t) = o(t')$ is immediate when $t = 0$, $t = 1$, or $t = a$, because in these cases we have $t = t'$. When $t = x$, it must be the case that t' is obtainable by a chain of single syntactic substitutions from $\mu x.(\sum_{a \in A} a \cdot x_a + o(x))$. But, as every chain of single syntactic substitutions from $\mu x.(\sum_{a \in A} a \cdot x_a + o(x))$ will be of the form $\mu x.(\sum_{a \in A} a \cdot s(a) + o(x))$, and $o(a \cdot s(a)) = 0$ for all $s(a)$, it follows easily that $o(t') = o(x)$. When $t = u + v$, it follows that t' must be of the form $u' + v'$, where u' is a closure of u, and v' is a closure of v. We then get $o(t) = o(u) \vee o(v) = o(u') \vee o(v') = o(t')$, using the inductive assumption that $o(u) = o(u')$ and $o(v) = o(v')$. The case where $t = u \cdot v$ goes analogously.

Showing that $(t_a, t'_a) \in R$, again, is immediate when $t = 0$, $t = 1$, or $t = a$. When $t = x$, the first single syntactic substitution to obtain t' must, again, be replacing x by $\mu x.(\sum_{a \in A} a \cdot x_a + o(x))$. As a result, t' must be of the shape $\mu x.(\sum_{a \in A} a \cdot s(a) + o(x))$, where each $s(a)$ is a $\{x\}$-pseudoclosure of x_a. It follows that $u_a := s(a)[\mu x.(\sum_{a \in A} a \cdot s(a) + o(x))/x]$ is a closure of x_a. As $(b \cdot t_b)_a = 0$ when $a \neq b$, as $(a \cdot t_a)_a = t_a$, and as $1_a = 0_a = 0$, it follows that $u_a \sim t'_a$: as u_a is a closure of t_a, the condition holds. The cases where $t = u + v$ and $t = u \cdot v$ are immediate from the inductive hypothesis and Proposition 9. □

Proof (of Proposition 13). In this proof, we define the *μ-pruning* of a term as the μ-free term obtained by replacing the outermost μ-expressions in it by the variables bound by these expressions.

For every variable x, let t_x be equal to the μ-pruning of u_x, where $\mu x.u_x$ is the unique sub-expression of t binding x. Let s be equal to the μ-pruning of t.

To see that there exists a declaration system (X, f) such that $s_x \equiv t_x$ for all $x \in X$, we note that for each t_x there exists a s_x with $s_x \equiv t_x$ of the form

$$s_x = \sum_{a \in A} a \cdot s(a, x) + o \quad (o \in \{0, 1\})$$

(The reason why this is so, roughly is the follows: because u_x occurs directly within the scope of a μ-operator, it has to be a guarded term. We can easily show, by induction on guarded terms g, that the μ-pruning of each such term has a normal form of this type.)

But we can easily construct a declaration system (X, f) that yields back these s_x, by setting, for each $x \in X$, $x_a = s(a, x)$, and $o(x) = o$.

To see that t is a closure of s, consider the following sublemma:

Lemma. *Given the above association of terms t_x to each variable x, if t is a μ-pruning of a term t' (that is an ingredient of the original term in consideration) with only free variables in $W \subseteq X$, then t' is a W-pseudoclosure of t.*

Proof (of Lemma). By induction on terms.

For terms t of the form 0, 1, a, if t is a μ-pruning of t', then $t = t'$, and hence t' is a closure (and, hence, a W-pseudoclosure for any $W \subseteq X$) of t as 0, 1, and a are closed terms.

For terms t' of the form $u' + v'$, if t is a μ-pruning of t', then t is of the form $u + v$, where u is a μ-pruning of u' and v is a μ-pruning of v'. But then, by the inductive hypothesis, u' is a W-pseudoclosure of u and v' is a W-pseudoclosure of v, and then, by Proposition 9, t' is a W-pseudoclosure of t. The case where $t = u \cdot v$ goes analogously.

For terms t' of the form $x \in W$, x is its own μ-pruning, and clearly x is a W-pseudoclosure of itself, as $x \in W$.

For terms t' of the form $\mu x.u$, if t is a μ-pruning of t', then $t = x$. A single syntactic substitution yields $\mu x.t_x$ from x, where t_x is the μ-pruning of the unique sub-expression binding x, that is u. As u can contain free variables from W as well as x itself, the inductive assumption gives us that u is a $W \cup \{x\}$-pseudoclosure of t_x, and now Proposition 9 gives the result that t' is a $(W \cup \{x\}) - \{x\}$-pseudoclosure (and, hence, also a W-pseudoclosure) of t. $\square$

The sublemma directly implies that t is a closure of s, and completes the proof of Proposition 13. $\square$