# Coalgebra, Coinduction, and Languages

Joost Winter

March 2, 2011

# Table of contents

# Overview (1)

In this talk, I will present a general overview of coalgebra and coinduction, and also present a bit of my own research. Some issues that will be treated are:

- First, we will start with an introduction, sketching the contrasts between the algebraic and the coalgebraic view on things, using the examples of finite lists and inductive definitions/proofs, and streams and coinductive definitions/proofs as an illustration.
- Then, we show how a coalgebraic approach towards languages has been made by Rutten, making use of Brzozowski derivatives, and give a nice characterization of regular languages in this framework.

# Overview (2)

- Finally, we show a novel approach, extending the earlier approach towards regular languages to context-free grammars and languages, and present three different coalgebraic characterizations of these. One of these characterizations builds on Rutten's earlier work on behavioural differential equations; another characterization makes use of fixed point-expressions.

# What is coalgebra?

Coalgebra is often seen as a notion that is *dual* to algebra, but what does this really mean?

- Algebra can be seen as being essentially about *construction*, whereas coalgebra is about *deconstruction* or *observation*.
- Likewise, the notion of an *initial algebra* in the algebraic world has its dual notion of a *final coalgebra* in the coalgebraic world.
- Looking from a perspective of category theory, many coalgebraic notions can be seen as formally dual to the algebraic notions.

# The algebraic picture: finite lists and induction (1)

We can view the following, familiar, definition of lists of elements from a carrier set $A$ as an *algebra*:

$$l ::= \mathsf{nil} \mid \mathsf{cons}(a \in A, l)$$

This algebra, a term algebra, is considered *initial* because no non-trivial equalities hold.
We can view this algebra as a mapping

$$f : 1 + (A \times X) \to X$$

where $+$ denotes the disjoint union. In general, an algebra over a signature $\Sigma$ can be seen as a mapping from $\Sigma X$ to $X$.

# The algebraic picture: finite lists and induction (2)

This definition enables us to use inductive definitions and proofs. As a familiar example, we can define the length of a list in the following manner, inductively:

$$\begin{aligned} \mathsf{len}(\mathsf{nil}) &= 0 \\ \mathsf{len}(\mathsf{cons}(a, l)) &= \mathsf{len}(l) + 1 \end{aligned}$$

# The coalgebraic picture: streams and coinduction (1)

Consider a machine with two buttons, *value* and *next*. Pressing the *value*-button gives a representation of the current state of the machine (let's say, an element of a set $A$), whereas pressing the *next*-button triggers a transition to a new state of the machine. This machine can be represented as a *coalgebra*:

$$\langle \text{value}, \text{next} \rangle : X \to A \times X$$

We should treat the carrier set $X$ as a black box: when doing this, the observable behaviour that remains is the *value* after pressing $n$ times on the *next*-button:

$$(a_0, a_1, a_2, \ldots) \in A^{\mathbb{N}}$$

# The coalgebraic picture: streams and coinduction (2)

Similarly, we can add a coalgebra structure to the set $A^{\mathbb{N}}$ of infinite sequences (or, as the coalgebra folks generally call them, *streams*) over $A$ as follows:

$$\langle \mathsf{head}, \mathsf{tail} \rangle : A^{\mathbb{N}} \to A \times A^{\mathbb{N}}$$

by, for $\alpha = (a_0, a_1, a_2, \ldots) \in A^{\mathbb{N}}$, declaring

$$\mathsf{head}(\alpha) = a_0 \qquad \mathsf{tail}(\alpha) = (a_1, a_2, a_3, \ldots)$$

In this case, the observable behaviour of an element of the carrier set – as before, a set $(a_0, a_1, a_2, \ldots)$ – will be *identical* to its content.

# The coalgebraic picture: streams and coinduction (3)

As an example of a coinductive definition, consider the following definition of a stream consisting of just the even elements $(a_0, a_2, a_4, \ldots)$ of another stream:

$$\begin{aligned} \mathrm{head}(\mathrm{even}(\sigma)) &= \mathrm{head}(\sigma) \\ \mathrm{tail}(\mathrm{even}(\sigma)) &= \mathrm{even}(\mathrm{tail}(\mathrm{tail}(\sigma))) \end{aligned}$$

Whereas, in an inductive definition of a function $f$, we define the value of $f$ on all constructors, in a coinductive definition we define the values of all destructors for each $f(x)$ (thereby providing a complete definition of $f$).

# Functors (1)

The notion of a *functor* originates from category theory. I will not give a fully general definition here, but illustrate the notion of a functor with a simple example.

## Example

Consider the function $F$, mapping any set $X$ to the set $X^*$ of finite lists over $X$. We can use $F$ to *lift* functions $f : X \to Y$ to functions $F(f) : F(X) \to F(Y)$, mapping the list

$$\langle x_1, \ldots, x_n \rangle$$

to the list

$$\langle f(x_1), \ldots, f(x_n) \rangle$$

# Functors (2)

In general, a functor $F$ is a function on both sets $X$ and functions $f : X \to Y$, satisfying the following conditions:

1. $F(1) = 1$
2. $F(f \circ g) = F(f) \circ F(g)$

# Functors (3)

As another example, consider the function $F(X) = A \times X$. Given any (fixed) set $A$, this function

1. Maps any set $X$ to the cartesian product $A \times X$.
2. Maps a function $f : X \to Y$ to the function $F(f) : A \times X \to A \times Y$, such that, for any $a \in A$ and $x \in X$, $F(f)(a, x) = (a, f(x))$. (In other words, $F(f)$ is the function $1 \times f$, applying the identity function to the first component, and $f$ to the second component).

It is easy to see that this mapping satisfies the two conditions on functorality.

# Coalgebra: a formal definition

We can now proceed with the following, formal, definition of a coalgebra:

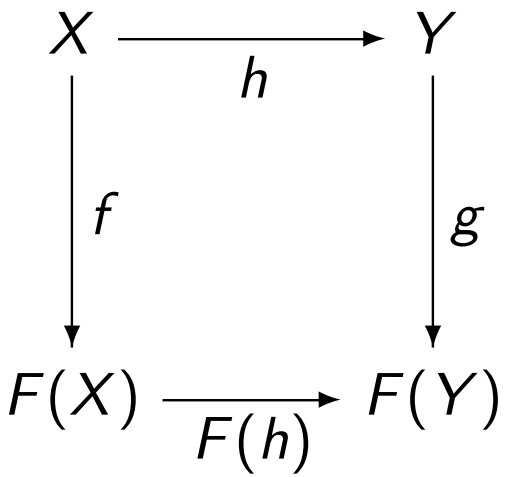
## Definition

A coalgebra over a functor $F$ is a tuple $(X, f)$, where $X$ is called the carrier set, and $f$ is a mapping $X \to F(X)$.

We can, for example, view the earlier example of streams as coalgebras over the functor $F(X) = A \times X$. (More later!)

# Homomorphisms, finality, and bisimulation (1)

Given two $F$-coalgebras $(X, f)$ and $(Y, g)$, we say a mapping $h : X \to Y$ is a *homomorphism* whenever $F(h) \circ f = g \circ h$.

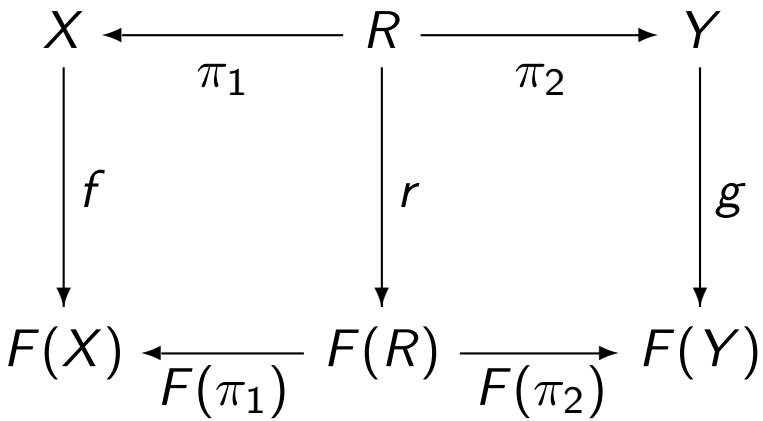


# Homomorphisms, finality, and bisimulation (2)

- We call a $F$-coalgebra $(Y, g)$ *final* if, for every $F$-coalgebra $(X, f)$ there is a *unique* homomorphism $h : X \to Y$. (For some functors, a final coalgebra exists, but not for all!)
- Given any $F$-coalgebra $(X, f)$, for any $x \in X$, we let $[\![x]\!]^f$ denote the value of $x$ under the unique homomorphism of $(X, f)$ to the final coalgebra (if it exists).

# Homomorphisms, finality, and bisimulation (3)

Given two $F$-coalgebras $(X, f)$ and $(Y, g)$, we say a relation $R \subseteq X \times Y$ is a *bisimulation* between $(X, f)$ and $(Y, g)$, whenever there is a function $r : R \to F(R)$ such that the projection functions $\pi_1 : R \to X$ and $\pi_2 : R \to Y$ to the first and second components of $R$ are homomorphisms:




A bisimulation can be seen as a relational homomorphism. In fact, the graph $\{(x, f(x)) \mid x \in X\}$ of any homomorphism $f : X \to Y$ always is a bisimulation between $X$ and $Y$.

# Homomorphisms, finality, and bisimulation (4)

- Given a coalgebra $(X, f)$, $x, y \in X$, and a bisimulation $R$ such that $(x, y) \in R$, we say that $x$ and $y$ are bisimilar and write $x \sim y$.
- Given two coalgebras $(X, f)$ and $(Y, g)$, any $x \in X$, $y \in Y$, and a bisimulation $R$ such that $(x, y) \in R$, we have $[\![x]\!]^f = [\![y]\!]^g$.
- This fact witnesses the notion that the notion of bisimilarity corresponds to behavioural equivalence: in the final coalgebra (for any functor that has one), there are no distinct states that are behaviourally equivalent.
- From this fact, it furthermore follows directly that bisimilarity is an equivalence relation.

# Coalgebra: an example (1)

Consider again the functor $F(X) = A \times X$:

- A *homomorphism* for two $F$-coalgebras $(X, f)$ and $(Y, g)$ is a function $h : X \to Y$, such that:

$$F(h) \circ f = g \circ h$$

$$\text{that is} \qquad (1 \times h) \circ f = g \circ h$$

  This is equivalent to: for all $x$, there are $y$ and $z$ such that $f(x) = (a, y)$, $g(h(x)) = (a, z)$, and $h(y) = z$.
- It turns out that the *final* coalgebra for the functor $F$ is the set $A^{\mathbb{N}}$ of infinite sequences over $A$, or as we will call them, streams over $A$.

# Coalgebra: an example (2)

- A *bisimulation* between two $F$-coalgebras $(X, f)$ and $(Y, g)$, is a relation $R$, such that whenever $(x, y) \in R$:
  1. $\pi_1(f(x)) = \pi_1(g(y))$
  2. $(\pi_2(f(x)), \pi_2(g(y))) \in R$

  It may not be trivial to see that earlier, general, definition of bisimulation instantiates to this one in case of the functor $F(X) = A \times X$, but this fact is easy to work out!

# Definitions – familiar and coalgebraic (1)

The familiar definition of a *deterministic automaton* is as follows: a *deterministic automaton* is a tuple $(Q, A, F, \delta)$, where

- $Q$ is a set of states (when $Q$ is finite, we talk about a *finite automaton*);
- $A$ is a finite set, called the alphabet;
- $F$ is a subset of $Q$ representing the *accepting* states; and
- $\delta$ is a transition function $Q \times A \to Q$, or equivalently, $Q \to Q^A$.

(Commonly, we also include an *initial state* $q_0 \in Q$, but right now, this is not needed.)

# Definitions – familiar and coalgebraic (2)

Alternatively, we can view a determinstic automaton as a function $f : Q \to 2 \times Q^A$ (where 2 is the set $\{0, 1\}$), which can be understood as follows:

- For every $q \in Q$, the first component $f(q)$ tells us whether $q$ is accepting or not.
- For every $q \in Q$, the second component of $f(q)$ gives us the partial application of the transition function $\delta$ to $q$.

# We have a coalgebra! (1)

- However, with this last definition, we have in fact defined a coalgebra for the functor $L(X) = 2 \times X^A$. Let's see what the notions of homomorphism, finality, and bisimulation will be here.
- Convention: for coalgebras $f : X \to 2 \times X^A$, we will represent $\pi_1(f(x))$ as $o(x)$, and $\pi_2(f(x))(a)$ as $x_a$. We will call $o(x)$ the *output value* of $x$, and $x_a$ the *a-derivative* of $x$.
  When confusion may arise about the coalgebra we are dealing with, we superscribe $o(x)$ and $x_a$ with the name of the transition function of the coalgebra. E.g. $o^f(x)$ and $x_a^f$ for a coalgebra $(X, f)$.

# We have a coalgebra! (2)

- We can extend the notions of derivatives w.r.t. alphabet symbols to word derivatives: for any $x \in X$,
  1. $x_\lambda = x$; and
  2. $x_{a \cdot w} = (x_a)_w$.

# We have a coalgebra! (3)

- A homomorphism between $L$-coalgebras $(X, f)$ and $(Y, g)$, is a function $f : X \to Y$, such that for all $x \in X$:
  1. $o(x) = o(f(x))$
  2. $f(x_a) = f(x)_a$
- The final coalgebra of the functor $L$ consists of the set $\mathcal{P}(A^*)$ of all languages over $A$. For any $\mathcal{L} \in \mathcal{P}(A^*)$, we have $o(\mathcal{L}) = 1$ iff $\lambda \in \mathcal{L}$, and $\mathcal{L}_a = \{s \in A^* \mid a \cdot s \in \mathcal{L}\}$.
- A bisimulation between $L$-coalgebras $(X, f)$ and $(Y, g)$, is a relation $R \subseteq X \times Y$, such that for any $(x, y) \in R$:
  1. $o(x) = o(y)$
  2. For all alphabet symbols $a$, $(x_a, y_a) \in R$

# Regular expressions and Brzozowski-derivatives (1)

The set $\mathcal{E}$ of *regular expressions* over an alphabet $A$ can be defined as follows:

$$t ::= a \in A \mid 0 \mid 1 \mid t + t \mid t \cdot t \mid t^*$$

We can assign a $L$-coalgebra structure to this set of regular expressions by specifying the output values and derivatives for each expression, giving us a $L$-coalgebra $(\mathcal{E}, e)$:

| $t$ | $o(t)$ | $t_a$ |
|---|---|---|
| $0$ | $0$ | $0$ |
| $1$ | $1$ | $0$ |
| $b$ | $0$ | if $b = a$ then $1$ else $0$ |
| $u + v$ | $o(u) \vee o(v)$ | $u_a + v_a$ |
| $u \cdot v$ | $o(u) \wedge o(v)$ | $u_a \cdot v + o(u) \cdot v_a$ |
| $u^*$ | $1$ | $u_a \cdot u^*$ |

# Regular expressions and Brzozowski-derivatives (2)

For any regular expression $t$, $[\![t]\!]^e$ is precisely the language denoted by $t$ according to the traditional semantics of regular expressions.

# Kleene's theorem, coalgebraically

- For any language $\mathcal{L} \in \mathcal{P}(A^*)$, we can define the subcoalgebra generated by $\mathcal{L}$ as

$$\langle \mathcal{L} \rangle := \{ \mathcal{L}_w \mid w \in A^* \}$$

  It is easy to see that this indeed generates a subcoalgebra: given any $\mathcal{K} \in \langle \mathcal{L} \rangle$, it is easy to see that for every $a \in A$, also $\mathcal{K}_a \in \langle \mathcal{L} \rangle$. In other words, $\langle \mathcal{L} \rangle$ is closed under taking derivatives to alphabet symbols.
- Kleene's theorem, coalgebraically: *For any $\mathcal{L} \in \mathcal{P}(A^*)$, $\langle L \rangle$ is finite iff there is a regular expression $t$ such that $\mathcal{L} = [\![t]\!]^e$.*

# Introduction: context-free grammars and languages

- The 'next step up' from regular expressions and languages, and finite automata, in the Chomsky hierarchy, are the context-free languages and grammars, and pushdown automata.
- We represent context-free grammars over a finite alphabet $A$, somewhat unusually, in a coalgebraic way as mappings $p : X \to \mathcal{P}_\omega((X + A)^*)$.
- We say a context-free grammar is in weak Greibach normal form, if for every $x \in X$, every $t \in p(x)$ is either equal to the empty word $\lambda$, or of the form $a \cdot t'$.
- As the name implies, this is a weakening of the more familiar Greibach normal form. As a direct result, every CFG can be represented in weak Greibach normal form.

# Grammar coalgebras (1)

- We start with a CFG in weak Greibach normal form

$$p : X \to \mathcal{P}_\omega((X + A)^*)$$

- We can transform this CFG into the form

$$p' : X \to 2 \times \mathcal{P}_\omega((X + A)^*)^A$$

by setting $o(x) = 1$ iff $\lambda \in p(x)$, and $x_a = \{t \mid a \cdot t \in p(x)\}$.

# Grammar coalgebras (2)

- We can now extend this mapping $p'$ into a $L$-coalgebra

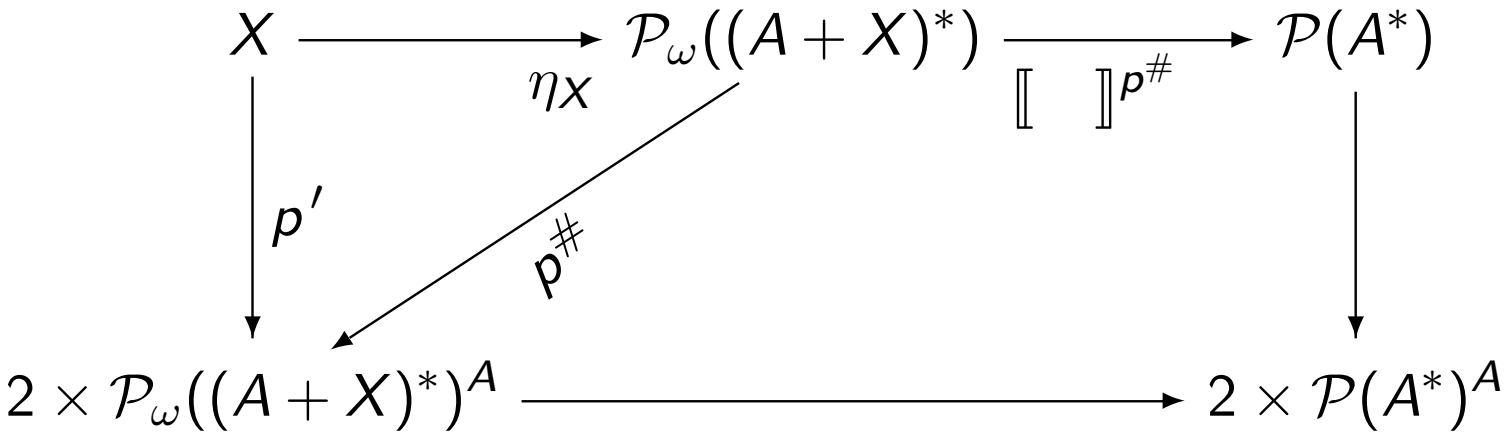
$$p^{\#} : \mathcal{P}_\omega((X + A)^*) \to 2 \times \mathcal{P}_\omega((X + A)^*)^A$$

called the *grammar coalgebra* of $p$, by defining $p^{\#}$ as follows:

| $S$ | $o(S)$ | $S_a$ |
|---|---|---|
| $\emptyset$ | $0$ | $\emptyset$ |
| $\{\lambda\}$ | $1$ | $\emptyset$ |
| $\{bs\}$ | $0$ | if $b = a$ then $\{s\}$ else $\emptyset$ |
| $\{xs\}$ | $o(s)$ | $\{ts \mid t \in x_a\} \cup \{s\}_a$ if $o(x) = 1$ |
| $\{xs\}$ | $0$ | $\{ts \mid t \in x_a\}$ if $o(x) = 0$ |
| $T \cup U$ | $o(T) \vee o(U)$ | $T_a \cup U_a$ |

# Grammar coalgebras (3)

- What we just did can be summarized in the following diagram:




- Proposition: *Given any context-free grammar $p$ in weak Greibach normal form, and any variable $x \in X$, $[\![\{x\}]\!]^{p^\#}$ is equal to the language generated by $x$. As a result, any context-free grammar occurs as the image under the final homomorphism of some grammar coalgebra.*

# Behavioural differential equations (1)

- In Rutten (2001), so-called *behavioural differential equations* were introduced as a specification format for streams. Here, a stream is specified by two expressions, one corresponding to its direct output value, and one to its derivative: under certain conditions, it turns out that such behavioural differential equations uniquely specify streams.
- Here, we will introduce a similar format of behavioural differential equations for the functor $L$, this time to specify languages.
- It should be noted that it is a well-known fact that context-free languages occur as solutions over equation systems for regular expressions – but a coalgebraic treatment of this fact has not yet been given until now.

# Behavioural differential equations (2)

We will be concerned with terms $t$ that can be specified as follows:

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

where $X$ is a finite set of variables. Given $X$, we let $TX$ denote the set of terms over $X$.
A well-formed system of equations, for a set of variables $X$, consists of:

1. For every $x \in X$, exactly one equation of the form $o(x) = v$, where $v \in \{0, 1\}$.
2. For every $x \in X$ and $a \in A$, exactly one equation of the form $x_a = t$, where $t \in TX$.

# Behavioural differential equations (3)

Alternatively, we can consider a well-formed system of equations as a mapping

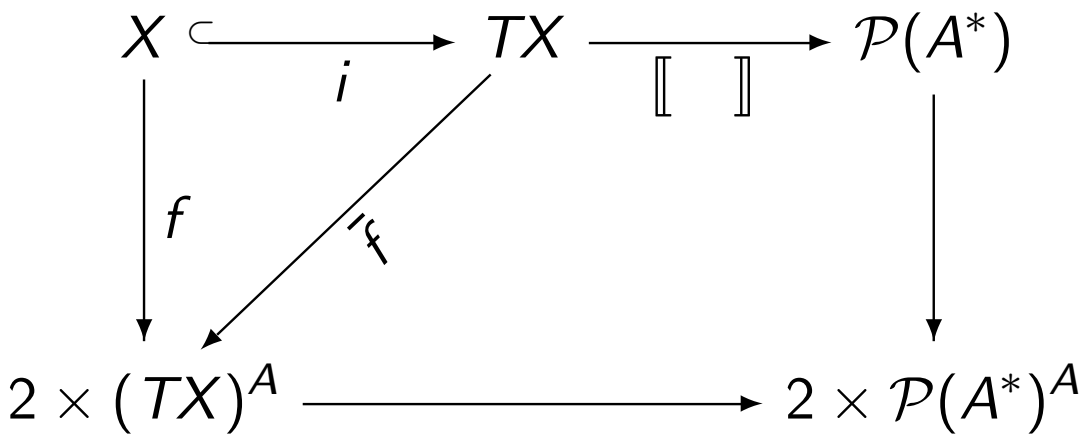
$$f : X \to 2 \times TX^A$$

We can extend such a mapping $f$ to the $L$-coalgebra $\bar{f} : TX \to 2 \times TX^A$ *generated by* $(X, f)$ as follows:

| $t$ | $o(t)$ | $t_a$ |
|---|---|---|
| $x$ | $o(x)$ | $x_a$ (as specified by $f$) |
| $0$ | $0$ | $0$ |
| $1$ | $1$ | $0$ |
| $b$ | $0$ | if $b = a$ then $1$ else $0$ |
| $u + v$ | $o(u) \vee o(v)$ | $u_a + v_a$ |
| $u \cdot v$ | $o(u) \wedge o(v)$ | $u_a \cdot v + o(u) \cdot v_a$ |

# Behavioural differential equations (4)

- This construction can again be summarized diagrammatically:




- Proposition: *A language $\mathcal{L}$ is context-free iff there is a well-formed system of equations $(X, f)$ and an $x \in X$, such that $[\![x]\!]^{\bar{f}} = \mathcal{L}$ w.r.t. the coalgebra $(TX, \bar{f})$ generated by it.*

# Context-free expressions (1)

- Our final coalgebraic representation of context-free languages will be *context-free expressions*, a generalization of regular expressions, where the Kleene-star is replaced by a (unique) fixed point operator $\mu$.
- Given an alphabet $A$ and a (countably infinite) set of variables $X$, we define context-free expressions $t$ and guarded expressions $g$ as follows:

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

# Context-free expressions (2)

We can define a $L$-coalgebra structure over these expressions as follows:

| $t$ | $o(t)$ | $t_a$ |
|---|---|---|
| $0$ | $0$ | $0$ |
| $1$ | $1$ | $0$ |
| $b$ | $0$ | if $b = a$ then $1$ else $0$ |
| $u + v$ | $o(u) \vee o(v)$ | $u_a + v_a$ |
| $u \cdot v$ | $o(u) \wedge o(v)$ | $u_a \cdot v + o(u) \cdot v_a$ |
| $\mu x.u$ | $o(u[\mu x.u/x])$ | $(u[\mu x.u/x])_a$ |

Due to the guardedness conditions, this is a well-defined definition.

# Context-free expressions (3)

Proposition: *A language $\mathcal{L}$ is context-free iff it occurs as the image of a context-free expression under the final homomorphism.*

# Wrap-up

- I hope you all now know a bit better what coalgebra is!
- There is a very neat coalgebraic representation of regular expressions, and Kleene's theorem can be expressed succinctly in a coalgebraic fashion.
- I have been trying to extend this work towards context-free languages and grammars, and provided three different, equivalent, coalgebraic treatments of these.
- Future work: extend the above work to other functors. This is likely to be a successful enterprise: Jan Rutten already provided a very nice example of a context-free stream.
- More future work: a coalgebraic account of pushdown-automata.

# Bibliography



[Jacobs/Rutten, 1997] Bart Jacobs, Jan Rutten, *A Tutorial on (Co)Algebras and (Co)Induction*


[Rutten, 1998] Jan Rutten, *Automata and Coinduction (An Exercise in Coalgebra)*


[Rutten, 2005] Jan Rutten, *A Coinductive Calculus of Streams*


[Silva, 2010] Alexandra Silva, *Kleene Coalgebra*


[Winter/Bonsangue/Rutten, 2011?] Joost Winter, Marcello Bonsangue, Jan Rutten, *Context-free Languages, Coalgebraically*