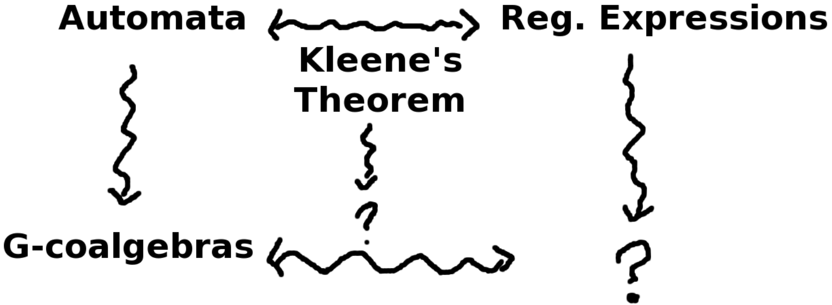
A Kleene theorem for polynomial coalgebras

Marcello Bonsangue[1,2] Jan Rutten[1,3] Alexandra Silva[1]

[1]Centrum voor Wiskunde en Informatica
[2]LIACS - Leiden University
[3]Vrije Universiteit Amsterdam

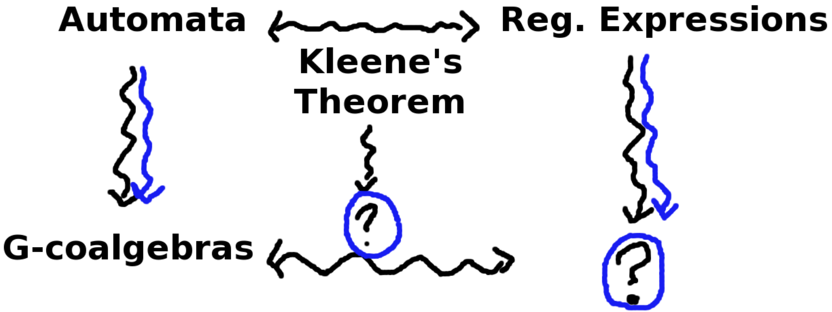
ACG, September 2008

Motivation



Motivation

Today :



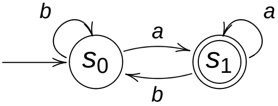
Deterministic automata. . .

- Deterministic finite automata (DFA) are a widely used model in Computer Science.
- They are acceptors of languages $A \subseteq \Sigma^*$.

- Formally: $\mathcal{A} = (Q, \Sigma, i, F \subseteq Q, \delta\ :\ Q \times \Sigma \to Q)$

Deterministic automata. . .

- Deterministic finite automata (DFA) are a widely used model in Computer Science.
- They are acceptors of languages $A \subseteq \Sigma^*$.

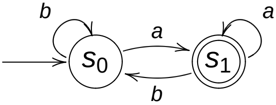


$\mathcal{L} = \{b^n a^m \mid n \geq 0, m > 0\} \cup \{b^n a^m b^k a^j \mid n \geq 0, m, k, j > 0\}$

- Formally: $\mathcal{A} = (Q, \Sigma, i, F \subseteq Q, \delta : Q \times \Sigma \to Q)$

Deterministic automata. . .

- Deterministic finite automata (DFA) are a widely used model in Computer Science.
- They are acceptors of languages $A \subseteq \Sigma^*$.

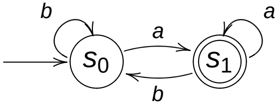


$\mathcal{L} = \{b^n a^m \mid n \geq 0, m > 0\} \cup \{b^n a^m b^k a^j \mid n \geq 0, m, k, j > 0\}$

- Formally: $\mathcal{A} = (Q, \Sigma, i, F \subseteq Q, \delta \ : \ Q \times \Sigma \to Q)$

Deterministic automata. . .

- Deterministic finite automata (DFA) are a widely used model in Computer Science.
- They are acceptors of languages $A \subseteq \Sigma^*$.



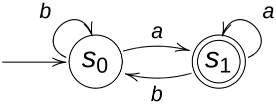
$\mathcal{L} = \{b^n a^m \mid n \geq 0, m > 0\} \cup \{b^n a^m b^k a^j \mid n \geq 0, m, k, j > 0\}$

- Formally: $\mathcal{A} = (Q, \Sigma, i, F \subseteq Q, \delta \ : \ Q \times \Sigma \to Q)$

$F : Q \to 2$

Deterministic automata...

- Deterministic finite automata (DFA) are a widely used model in Computer Science.
- They are acceptors of languages $A \subseteq \Sigma^*$.



$\mathcal{L} = \{b^n a^m \mid n \geq 0, m > 0\} \cup \{b^n a^m b^k a^j \mid n \geq 0, m, k, j > 0\}$

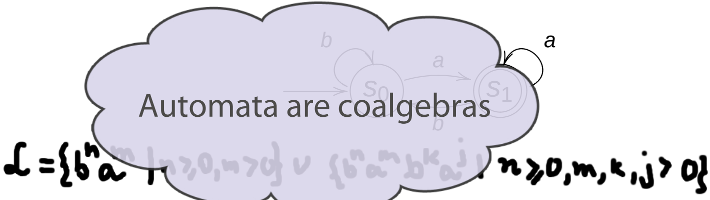
- Formally: $\mathcal{A} = (Q, \Sigma, i, F \subseteq Q, \delta : Q \times \Sigma \to Q)$

$F : Q \to 2$ $\qquad$ $t : Q \longrightarrow 2 \times Q^\Sigma$

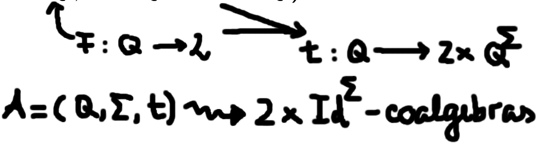
$A = (Q, \Sigma, t) \rightsquigarrow 2 \times Id^\Sigma$ - coalgebras

Deterministic automata...

- Deterministic finite automata (DFA) are a widely used model in Computer Science.
- They are acceptors of languages $A \subseteq \Sigma^*$.



- Formally: $\mathcal{A} = (Q, \Sigma, i, F \subseteq Q, \delta : Q \times \Sigma \to Q)$



...and regular expressions

- *User-friendly* alternative to DFA notation.
- Simple syntax:

$$E ::= a \in \Sigma \mid \epsilon \mid \emptyset \mid E + E \mid E \cdot E \mid E^*$$

- They also represent languages:

$$L(a) = \{a\}, \quad L(\epsilon) = \{\epsilon\}$$
$$L(E_1 + E_2) = L(E_1) \cup L(E_2)$$
$$L(E_1 \cdot E_2) = L(E_1) \cdot L(E_2)$$
$$L(E^\star) = \bigcup_{n \in \mathbb{N}} L(E)^n$$

. . . and regular expressions

- *User-friendly* alternative to DFA notation.
- Simple syntax:

$$E ::= a \in \Sigma \mid \epsilon \mid \emptyset \mid E + E \mid E \cdot E \mid E^*$$

- They also represent languages:

$$L(a) = \{a\}, \quad L(\epsilon) = \{\epsilon\}$$
$$L(E_1 + E_2) = L(E_1) \cup L(E_2)$$
$$L(E_1 \cdot E_2) = L(E_1) \cdot L(E_2)$$
$$L(E^\star) = \bigcup_{n \in \mathbb{N}} L(E)^n$$

. . . and regular expressions

- *User-friendly* alternative to DFA notation.
- Simple syntax:

$$E ::= a \in \Sigma \mid \epsilon \mid \emptyset \mid E + E \mid E \cdot E \mid E^*$$

- They also represent languages:

$$\begin{array}{l} L(a) = \{a\}, \quad L(\epsilon) = \{\epsilon\} \\ L(E_1 + E_2) = L(E_1) \cup L(E_2) \\ L(E_1 \cdot E_2) = L(E_1) \cdot L(E_2) \\ L(E^\star) = \bigcup_{n \in \mathbb{N}} L(E)^n \end{array}$$

. . . and regular expressions

- *User-friendly* alternative to DFA notation.
- Simple syntax:

$$E ::= a \in \Sigma \mid \epsilon \mid \emptyset \mid E + E \mid E \cdot E \mid E^*$$

- They also represent languages:

$$\begin{array}{l} L(a) = \{a\}, \quad L(\epsilon) = \{\epsilon\} \\ L(E_1 + E_2) = L(E_1) \cup L(E_2) \\ L(E_1 \cdot E_2) = L(E_1) \cdot L(E_2) \\ L(E^\star) = \bigcup_{n \in \mathbb{N}} L(E)^n \end{array}$$

$$\mathcal{L} = \{b^n a^m \mid n \geq 0, m > 0\} \cup \{b^n a^m b^k a^j \mid n \geq 0, m, k, j > 0\}$$

…and regular expressions

- *User-friendly* alternative to DFA notation.
- Simple syntax:

$$E ::= a \in \Sigma \mid \epsilon \mid \emptyset \mid E + E \mid E \cdot E \mid E^*$$

- They also represent languages:

$$\begin{aligned} &L(a) = \{a\}, \quad L(\epsilon) = \{\epsilon\} \\ &L(E_1 + E_2) = L(E_1) \cup L(E_2) \\ &L(E_1 \cdot E_2) = L(E_1) \cdot L(E_2) \\ &L(E^\star) = \bigcup_{n \in \mathbb{N}} L(E)^n \end{aligned}$$

$\mathcal{L} = \{b^n a^m \mid n \geq 0, m > 0\} \cup \{b^n a^m b^k a^j \mid n \geq 0, m, k, j > 0\}$
$= b^* a (b^* a)^*$

Kleene's Theorem

DFA = RE

Let $A \subseteq \Sigma^*$. The following are equivalent.

1. $A = L(\mathcal{A})$, for some finite automaton $\mathcal{A}$.
2. $A = L(r)$, for some regular expression r.

In the proof of the theorem, one learns how to transform RE in DFA and vice-versa.

Kleene's Theorem

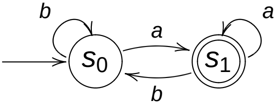
DFA = RE

Let $A \subseteq \Sigma^*$. The following are equivalent.

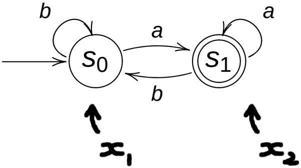
1. $A = L(\mathcal{A})$, for some finite automaton $\mathcal{A}$.
2. $A = L(r)$, for some regular expression r.

In the proof of the theorem, one learns how to transform RE in DFA and vice-versa.

From DFA to RE

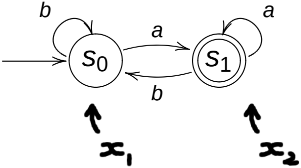


From DFA to RE



$$x_1 = bx_1 + ax_2$$
$$x_2 = \epsilon + ax_2 + bx_1$$

From DFA to RE



$$x_1 = bx_1 + ax_2$$
$$x_2 = \epsilon + ax_2 + bx_1$$

$$x_1 = b^*a(b^*a)^*$$
$$x_2 = b^*ax_1$$

Solve
$x = r + sx$
$\Downarrow$
$x = s^* r$

Beyond deterministic automata

Deterministic automata

$Q \to 2 \times Q^{\Sigma}$

$\Updownarrow$

Regular Expressions

$\Updownarrow$

Formal Languages

Beyond deterministic automata

Deterministic automata $Q \to 2 \times Q^\Sigma$	$\rightsquigarrow$	G-coalgebras $Q \to GQ$

$$\Updownarrow$$

Regular Expressions

$$\Updownarrow$$

Formal Languages

Beyond deterministic automata

Deterministic automata $Q \to 2 \times Q^{\Sigma}$	$\rightsquigarrow$	G-coalgebras $Q \to GQ$
$\Updownarrow$		
Regular Expressions	$\rightsquigarrow$	?
$\Updownarrow$		
Formal Languages	$\rightsquigarrow$	?

Beyond deterministic automata

Deterministic automata $Q \to 2 \times Q^\Sigma$	$\rightsquigarrow$	G-coalgebras $Q \to GQ$
$\Updownarrow$		$\Updownarrow$
Regular Expressions	$\rightsquigarrow$	G-expressions
$\Updownarrow$		$\Updownarrow$
Formal Languages	$\rightsquigarrow$	Final coalgebra

Polynomial coalgebras

- Generalizations of deterministic automata
- Polynomial coalgebras: set of states $S + t\ :\ S \to GS$

$$G ::= Id \mid B \mid G \times G \mid G + G \mid G^A$$

Examples

- $G = 2 \times Id^A$ – Deterministic automata
- $G = (B \times Id)^A$ – Mealy machines
- $G = Id \times A \times Id$ – Binary tree automata

Polynomial coalgebras

- Generalizations of deterministic automata
- Polynomial coalgebras: set of states S + $t \;:\; S \rightarrow GS$

$$G ::= Id \mid B \mid G \times G \mid G + G \mid G^A$$

Examples

- $G = 2 \times Id^A$ – Deterministic automata
- $G = (B \times Id)^A$ – Mealy machines
- $G = Id \times A \times Id$ – Binary tree automata

Beyond deterministic automata

Regular expressions for polynomial coalgebras

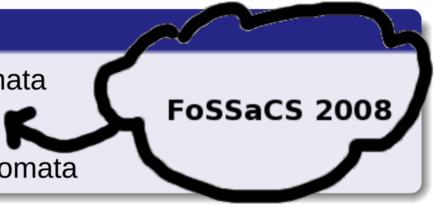
Polynomial coalgebras

- Generalizations of deterministic automata
- Polynomial coalgebras: set of states $S + t \ : \ S \to GS$

$$G ::= Id \mid B \mid G \times G \mid G + G \mid G^A$$

Examples

- $G = 2 \times Id^A$ – Deterministic automata
- $G = (B \times Id)^A$ – Mealy machines
- $G = Id \times A \times Id$ – Binary tree automata



In a nutshell

Our contributions are:

- A (syntactic) notion of *G-expressions* for polynomial coalgebras: each expression will denote an element of the final coalgebra.
- We show the equivalence between *G*-expressions and finite *G*-coalgebras (analogously to Kleene's theorem).

G-expressions

$$E \quad ::= \quad \emptyset \mid \epsilon \mid E \cdot E \mid E + E \mid E^*$$

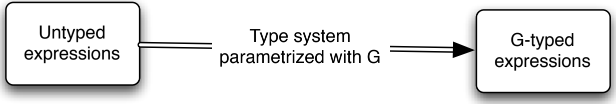
$$E_G \quad ::= \quad ?$$

G-expressions

$$E \quad ::= \quad \emptyset \mid \epsilon \mid E \cdot E \mid E + E \mid E^*$$

$$E_G \quad ::= \quad ?$$

How do we define E_G?

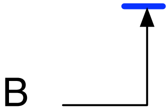


Untyped expressions

$$Exp \ni \varepsilon \quad ::= \quad \emptyset \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid b \mid l(\varepsilon) \mid r(\varepsilon) \mid l[\varepsilon] \mid r[\varepsilon] \mid a(\varepsilon)$$

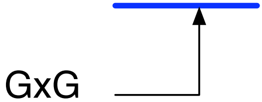
Untyped expressions

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



Untyped expressions

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



Untyped expressions

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

Untyped expressions

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

G^A

G-expressions

$$Exp_G = \{\varepsilon \in Exp \mid \vdash \varepsilon : G \lhd G\}$$

The type system is defined inductively on the ingredients of G – $F \lhd G$ – in the expected way.

$$\frac{}{\vdash \emptyset \ : \ F \lhd G} \quad \frac{}{\vdash b \ : \ B \lhd G} \quad \frac{\vdash \varepsilon \ : \ F_1 \lhd G}{\vdash l(\varepsilon) \ : \ F_1 \times F_2 \lhd G} \quad \cdots$$

G-expressions

$$Exp_G = \{\varepsilon \in Exp \mid \vdash \varepsilon : G \lhd G\}$$

The type system is defined inductively on the ingredients of G – $F \lhd G$ – in the expected way.

$$\frac{}{\vdash \emptyset \ : \ F \lhd G} \quad \frac{}{\vdash b \ : \ B \lhd G} \quad \frac{\vdash \varepsilon \ : \ F_1 \lhd G}{\vdash l(\varepsilon) \ : \ F_1 \times F_2 \lhd G} \quad \cdots$$

Examples

Deterministic automata expressions – $G = 2 \times Id^A$

$$\varepsilon \quad ::= \quad \emptyset \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid l(1) \mid l(0) \mid r(a(\varepsilon))$$

Examples

Deterministic automata expressions – $G = 2 \times Id^A$

$$\varepsilon \quad ::= \quad \emptyset \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid l(1) \mid l(0) \mid r(a(\varepsilon))$$

Mealy expressions – $G = (B \times Id)^A$

$$\varepsilon \quad ::= \quad \emptyset \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid a(l(b)) \mid a(r(\varepsilon))$$

Examples

Deterministic automata expressions – $G = 2 \times Id^A$

$$\varepsilon \quad ::= \quad \emptyset \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid l(1) \mid l(0) \mid r(a(\varepsilon))$$

Mealy expressions – $G = (B \times Id)^A$

$$\varepsilon \quad ::= \quad \emptyset \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid a(l(b)) \mid a(r(\varepsilon))$$

Binary tree expressions – $G = (Id \times (A \times Id))$

$$\varepsilon \quad ::= \quad \emptyset \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid l(\varepsilon) \mid r(l(a)) \mid r(r(\varepsilon))$$

Examples

Remark

Deterministic automata expressions – $G = 2 \times Id^A$

$$\varepsilon \quad ::= \quad \emptyset \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid l(1) \mid l(0) \mid r(a(\varepsilon))$$

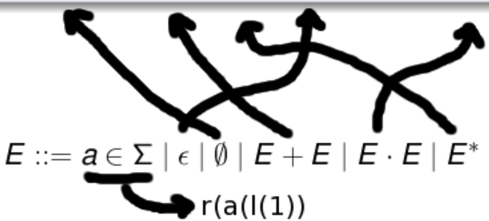
$$E ::= a \in \Sigma \mid \epsilon \mid \emptyset \mid E + E \mid E \cdot E \mid E^*$$

Examples

Remark

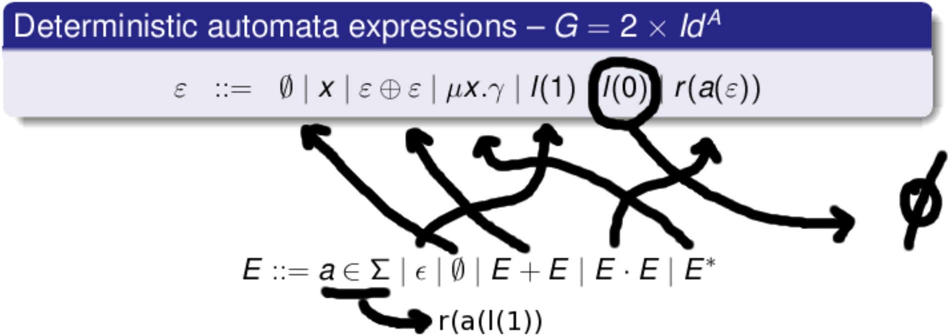
Deterministic automata expressions – $G = 2 \times Id^A$

$$\varepsilon \quad ::= \quad \emptyset \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid l(1) \mid l(0) \mid r(a(\varepsilon))$$



Examples

Remark



Kleene's theorem

The goal is:

G – *expressions* **correspond to** Finite *G* – *coalgebras* and vice-versa.

What does it mean **correspond**?

Final coalgebras exist for polynomial coalgebras.

correspond $\equiv$ mapped to the same element of the final coalgebra

Kleene's theorem

The goal is:

$G - expressions$ **correspond to** Finite $G - coalgebras$ and vice-versa.

What does it mean **correspond**?

Final coalgebras exist for polynomial coalgebras.

$$
\begin{array}{ccccc}
S & \overset{h}{\dashrightarrow} & \Omega_G & \overset{[\![\cdot]\!]}{\dashleftarrow} & Exp_G \\
\downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \omega_G} & & \\
GS & \underset{Gh}{\dashrightarrow} & G\Omega_G & &
\end{array}
$$

correspond $\equiv$ **mapped to the same element of the final coalgebra**

Kleene's theorem

The goal is:

$G-$*expressions* **correspond to** Finite $G-$*coalgebras* and vice-versa.

What does it mean **correspond**?

Final coalgebras exist for polynomial coalgebras.

$$\begin{array}{ccccc} S & \overset{h}{\dashrightarrow} & \Omega_G & \overset{[\![\cdot]\!]}{\dashleftarrow} & Exp_G \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \omega_G} & & \\ GS & \underset{Gh}{\dashrightarrow} & G\Omega_G & & \end{array}$$

correspond $\equiv$ mapped to the same element of the final coalgebra

Kleene's theorem

The goal is:

$G-$*expressions* **correspond to** Finite $G-$*coalgebras* and vice-versa.

What does it mean **correspond**?

Final coalgebras exist for polynomial coalgebras.

$$\begin{array}{ccccc} S & \overset{h}{\dashrightarrow} & \Omega_G & \overset{[\![\cdot]\!]}{\dashleftarrow} & Exp_G \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \omega_G} & & \\ GS & \underset{Gh}{\dashrightarrow} & G\Omega_G & & \end{array}$$

correspond $\equiv$ **mapped to the same element of the final coalgebra**

Semantics

G-expressions $\Leftrightarrow$ Final coalgebra

We turn Exp_G into a G-coalgebra:

$$Exp_G \xrightarrow[\lambda_G]{} GExp_G$$

which provides a final semantics:

$$\begin{array}{ccc} Exp_G & \overset{[\![\cdot]\!]}{\dashrightarrow} & \Omega_G \\ {\scriptstyle \lambda_G}\downarrow & & \downarrow{\scriptstyle \omega_G} \\ GExp_G & \underset{G[\![\cdot]\!]}{\dashrightarrow} & G\Omega_G \end{array}$$

and a notion of equivalence: $\sim_G$.

We turn Exp_G into a G-coalgebra:

$$Exp_G \xrightarrow[\lambda_G]{} GExp_G$$

which provides a final semantics:

$$\begin{array}{ccc} Exp_G & \overset{[\![\cdot]\!]}{\dashrightarrow} & \Omega_G \\ {\scriptstyle \lambda_G}\downarrow & & \downarrow{\scriptstyle \omega_G} \\ GExp_G & \underset{G[\![\cdot]\!]}{\dashrightarrow} & G\Omega_G \end{array}$$

and a notion of equivalence: $\sim_G$.

A generalized Kleene theorem

G-coalgebras $\Leftrightarrow$ G-expressions

Theorem

1. *Let (S,g) be a G-coalgebra. If S is* finite *then there exists for any $s \in S$ a G-expression ε_s such that $\varepsilon_s \sim s$.*
2. *For all G-expressions ε, there exists a finite G-coalgebra (S, g) such that $\exists_{s \in S}\ s \sim \varepsilon$.*

Proof(sketch)

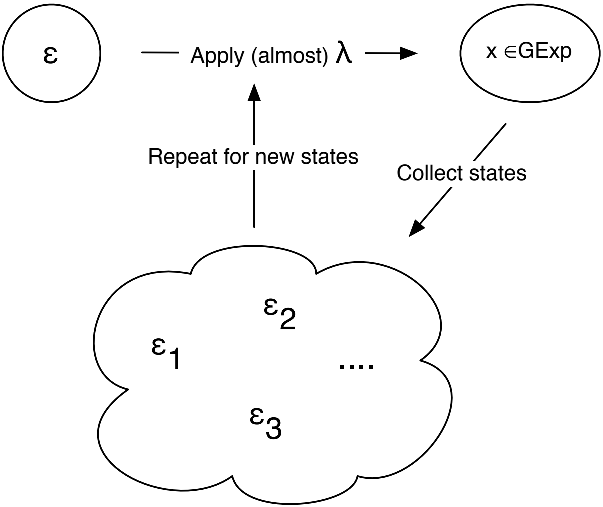
For **1**, we use a similar strategy as for DFA. We associate with every state $s \in S$ a variable $x_s \in X$ and an expression $\varepsilon_s = \mu x_s.\, \varepsilon_s^G$ defined by induction on the structure of G as follows.

$$\begin{aligned}
&\varepsilon_s^{Id} = \emptyset \\
&\varepsilon_s^{B} = g(s) \\
&\varepsilon_s^{G_1 \times G_2} = l(\varepsilon^{G_1}_{\pi_1 \circ g(s)}) \oplus r(\varepsilon^{G_2}_{\pi_2 \circ g(s)}) \\
&\vdots
\end{aligned}$$

and then we prove that $R = \{(s, \varepsilon_s) \mid s \in S\}$ is a bisimulation.

Proof(sketch)

For **2**:



Example

Blackboard

Conclusions and Future work

Conclusions

- Language of regular expressions for polynomial coalgebras
- Generalization of Kleene theorem

Future work

- Enlarge the class of functors treated: add $\mathcal{P}$
- Axiomatization of the language
- Model checking

Conclusions and Future work

Conclusions

- Language of regular expressions for polynomial coalgebras
- Generalization of Kleene theorem

Future work

- Enlarge the class of functors treated: add $\mathcal{P}$
- Axiomatization of the language
- Model checking