A decision procedure for bisimilarity of generalized regular expressions

M. Bonsangue[1,2] G. Caltais[5] E. Goriac[5] D. Lucanu[4]
J. Rutten[1,3] A. Silva[1]

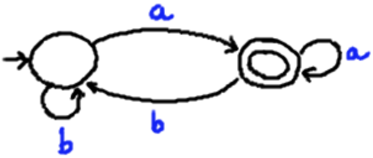
[1]Centrum Wiskunde & Informatica, The Netherlands
[2]LIACS - Leiden University, The Netherlands
[3]Radboud Universiteit Nijmegen, The Netherlands
[4]Faculty of Computer Science - Alexandru Ioan Cuza University, Romania
[5]School of Computer Science - Reykjavik University, Iceland

IPA Herfstdagen, November 2010

Motivation

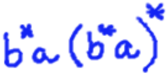
Deterministic automata (DA)

- Widely used model in Computer Science.
- Acceptors of languages



Regular expressions

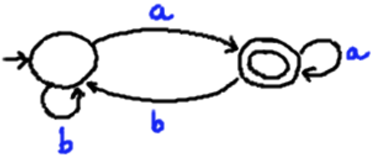
- *User-friendly* alternative to DA notation.
- Many applications: pattern matching (`grep`), specification of circuits, ...



Motivation

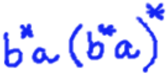
Deterministic automata (DA)

- Widely used model in Computer Science.
- Acceptors of languages



Regular expressions

- *User-friendly* alternative to DA notation.
- Many applications: pattern matching (`grep`), specification of circuits, ...

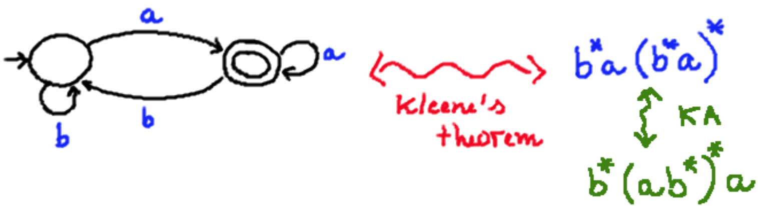


Kleene's Theorem

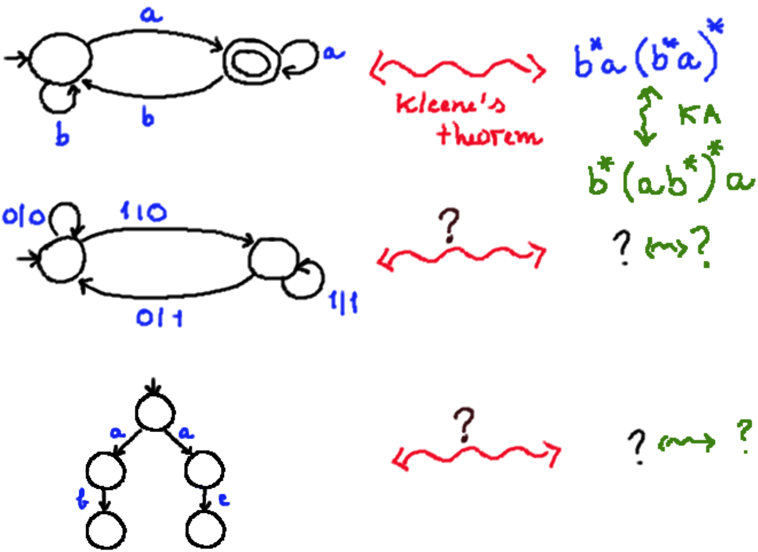
Let $A \subseteq \Sigma^*$. The following are equivalent.

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

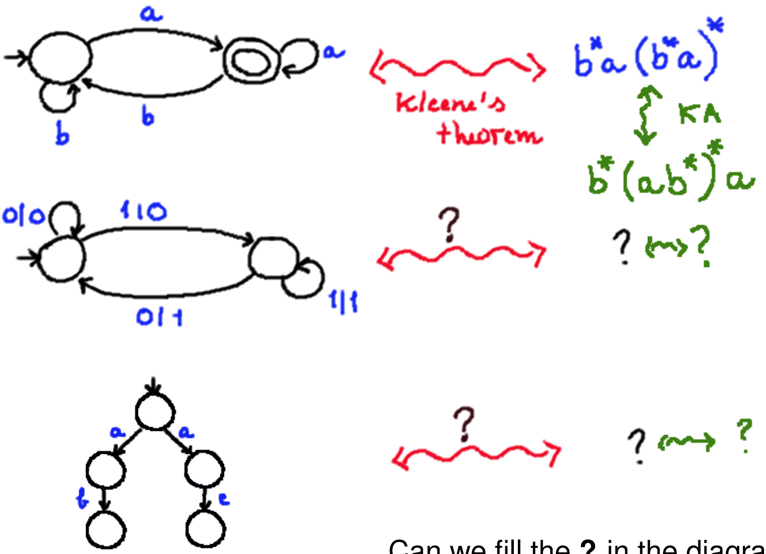
Motivation



Motivation



Motivation



Can we fill the **?** in the diagram?

In previous work . . .

We presented:

- a generalized notion of regular expressions;
- an analogue of Kleene's theorem;
- and sound and complete axiomatizations with respect to bisimilarity

for a large class of systems (labelled transition systems, Mealy machines, probabilistic automata).

All the above was derived modularly from the type of each system.

Question: Can we automate the reasoning on equivalence of expressions, also in a modular way?

In previous work . . .

We presented:

- a generalized notion of regular expressions;
- an analogue of Kleene's theorem;
- and sound and complete axiomatizations with respect to bisimilarity

for a large class of systems (labelled transition systems, Mealy machines, probabilistic automata).

All the above was derived modularly from the type of each system.

Question: Can we automate the reasoning on equivalence of expressions, also in a modular way?

In previous work . . .

We presented:

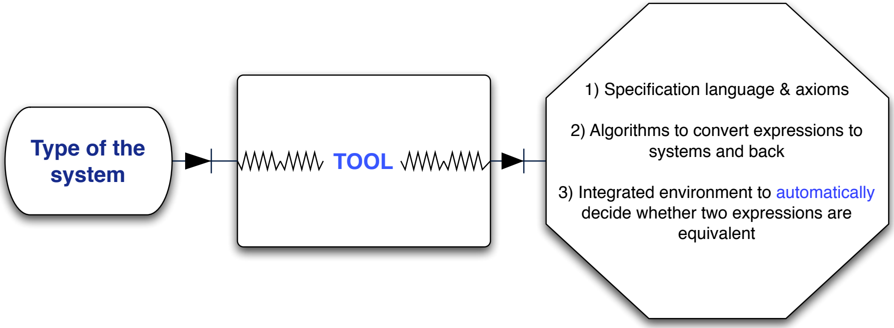
- a generalized notion of regular expressions;
- an analogue of Kleene's theorem;
- and sound and complete axiomatizations with respect to bisimilarity

for a large class of systems (labelled transition systems, Mealy machines, probabilistic automata).

All the above was derived modularly from the type of each system.

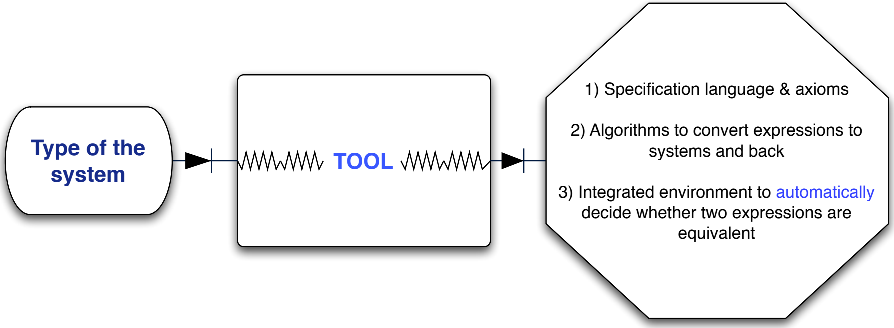
Question: Can we automate the reasoning on equivalence of expressions, also in a modular way?

The ultimate goal. . .



In this talk, we will be focusing on 1) and 3).

The ultimate goal. . .

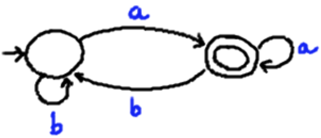


In this talk, we will be focusing on 1) and 3).

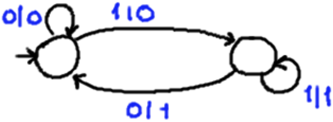
Outline

- Generalized regular expressions
- Equivalence of expressions
- Snapshot of the tool

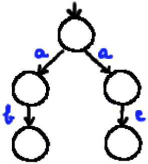
What do these things have in common?



$(S, \delta : S \to 2 \times S^A)$

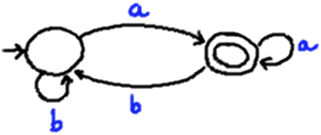


$(S, \delta : S \to (B \times S)^A)$

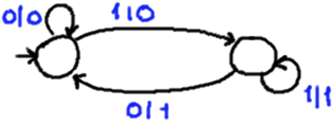


$(S, \delta : S \to 1 + (\mathcal{P}S)^A)$

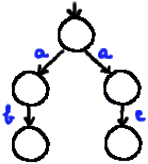
What do these things have in common?



$(S, \delta : S \to 2 \times S^A)$

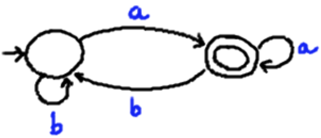


$(S, \delta : S \to (B \times S)^A)$

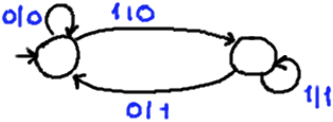


$(S, \delta : S \to 1 + (\mathcal{P}S)^A)$

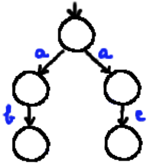
What do these things have in common?



$(S, \delta : S \to 2 \times S^A)$

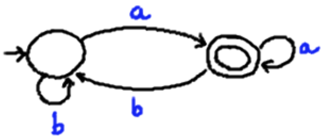


$(S, \delta : S \to (B \times S)^A)$

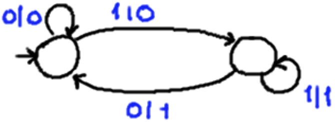


$(S, \delta : S \to 1 + (\mathcal{P}S)^A)$

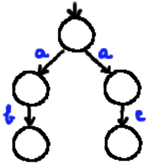
What do these things have in common?



$$(S, \delta : S \to 2 \times S^A)$$

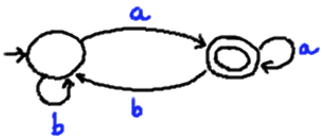


$$(S, \delta : S \to (B \times S)^A)$$

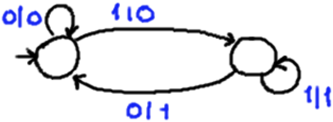


$$(S, \delta : S \to 1 + (\mathcal{P}S)^A)$$

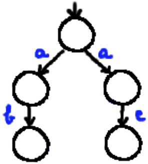
What do these things have in common?



$$(S, \delta : S \rightarrow 2 \times S^A)$$

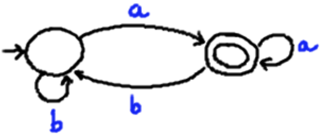


$$(S, \delta : S \rightarrow (B \times S)^A)$$

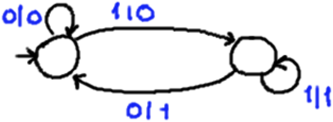


$$(S, \delta : S \rightarrow 1 + (\mathcal{P}S)^A)$$

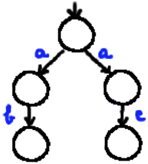
What do these things have in common?



$$(S, \delta : S \to 2 \times S^A)$$



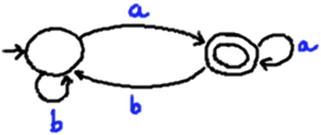
$$(S, \delta : S \to (B \times S)^A)$$



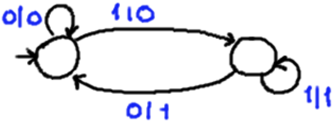
$$(S, \delta : S \to 1 + (\mathcal{P}S)^A)$$

$$(S, \delta : S \to \mathcal{G}S)$$

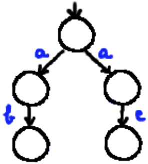
What do these things have in common?



$$(S, \delta : S \to 2 \times S^A)$$



$$(S, \delta : S \to (B \times S)^A)$$



$$(S, \delta : S \to 1 + (\mathcal{P}S)^A)$$

$(S, \delta : S \to \mathcal{G}S)$ **$\mathcal{G}$-coalgebras**

Coalgebras

Kripke polynomial coalgebras

- Generalizations of deterministic automata
- Kripke polynomial coalgebras: set of states S and $t : S \to GS$

$$\mathcal{G} ::= Id \mid B \mid \mathcal{G} \times \mathcal{G} \mid \mathcal{G} + \mathcal{G} \mid \mathcal{G}^A \mid \mathcal{P}\mathcal{G}$$

$\mathcal{P}$ finite

Examples

- $\mathcal{G} = 2 \times Id^A$ Deterministic automata
- $\mathcal{G} = (B \times Id)^A$ Mealy machines
- $\mathcal{G} = 1 + (\mathcal{P} Id)^A$ LTS (with explicit termination)
- ...

Coalgebras

Kripke polynomial coalgebras

- Generalizations of deterministic automata
- Kripke polynomial coalgebras: set of states S and $t : S \to GS$

$$\mathcal{G} ::= Id \mid B \mid \mathcal{G} \times \mathcal{G} \mid \mathcal{G} + \mathcal{G} \mid \mathcal{G}^A \mid \mathcal{P}\mathcal{G}$$

$\mathcal{P}$ finite

Examples

- $\mathcal{G} = 2 \times Id^A$ — Deterministic automata
- $\mathcal{G} = (B \times Id)^A$ — Mealy machines
- $\mathcal{G} = 1 + (\mathcal{P}Id)^A$ — LTS (with explicit termination)
- ...

The power of $\mathcal{G}$

The functor $\mathcal{G}$ determines:

1. notion of observational equivalence (coalg. bisimulation)
2. behaviour (final coalgebra)
3. set of expressions describing finite systems
4. axioms to prove bisimulation equivalence of expressions

The power of $\mathcal{G}$

The functor $\mathcal{G}$ determines:

1. notion of observational equivalence (coalg. bisimulation)
2. behaviour (final coalgebra)
3. set of expressions describing finite systems
4. axioms to prove bisimulation equivalence of expressions

The power of $\mathcal{G}$

The functor $\mathcal{G}$ determines:

1. notion of observational equivalence (coalg. bisimulation)
2. behaviour (final coalgebra)
3. set of expressions describing finite systems
4. axioms to prove bisimulation equivalence of expressions

1 + 2 are standard universal coalgebra; 3 + 4 are [BRS10]

In a nutshell — beyond deterministic automata

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

In a nutshell — beyond deterministic automata

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

$\mathcal{G}$-expressions

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

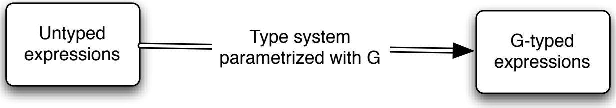
$$E_{\mathcal{G}} \quad ::= \quad ?$$

$\mathcal{G}$-expressions

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

$$E_{\mathcal{G}} \quad ::= \quad ?$$

How do we define $E_{\mathcal{G}}$?



$\mathcal{G}$-expressions

$$
\begin{array}{llll}
\text{Exp} \ni \varepsilon & ::= & \underline{\emptyset} \mid \varepsilon \oplus \varepsilon & \mid \mu x.\gamma \\
 & & & \mid b & B \\
 & & & \mid l\langle\varepsilon\rangle \mid r\langle\varepsilon\rangle & \mathcal{G}_1 \times \mathcal{G}_2 \\
 & & & \mid l[\varepsilon] \mid r[\varepsilon] & \mathcal{G}_1 + \mathcal{G}_2 \\
 & & & \mid a(\varepsilon) & \mathcal{G}^A \\
 & & & \mid \{\varepsilon\} & \mathcal{P}\mathcal{G}
\end{array}
$$

$\mathcal{G}$-expressions

$$
\begin{array}{llll}
\text{Exp} \ni \varepsilon & ::= & \underline{\emptyset} \mid \varepsilon \oplus \varepsilon & \mid \mu x.\gamma & \\
& & & \mid b & B \\
& & & \mid l\langle\varepsilon\rangle \mid r\langle\varepsilon\rangle & \mathcal{G}_1 \times \mathcal{G}_2 \\
& & & \mid l[\varepsilon] \mid r[\varepsilon] & \mathcal{G}_1 + \mathcal{G}_2 \\
& & & \mid a(\varepsilon) & \mathcal{G}^A \\
& & & \mid \{\varepsilon\} & \mathcal{P}\mathcal{G}
\end{array}
$$

$\mathcal{G}$-expressions

$$
\begin{array}{rclll}
\mathrm{Exp} \ni \varepsilon & ::= & \underline{\emptyset} \mid \varepsilon \oplus \varepsilon & \mid \mu x.\gamma & \\
& & & \mid b & B \\
& & & \mid l\langle\varepsilon\rangle \mid r\langle\varepsilon\rangle & \mathcal{G}_1 \times \mathcal{G}_2 \\
& & & \mid l[\varepsilon] \mid r[\varepsilon] & \mathcal{G}_1 + \mathcal{G}_2 \\
& & & \mid a(\varepsilon) & \mathcal{G}^A \\
& & & \mid \{\varepsilon\} & \mathcal{P}\mathcal{G}
\end{array}
$$

$\mathcal{G}$-expressions

$$
\begin{array}{rcll}
\text{Exp} \ni \varepsilon & ::= & \underline{\emptyset} \mid \varepsilon \oplus \varepsilon & \mid \mu x.\gamma \\
 & & & \mid b \qquad\qquad B \\
 & & & \mid l\langle\varepsilon\rangle \mid r\langle\varepsilon\rangle \qquad \mathcal{G}_1 \times \mathcal{G}_2 \\
 & & & \mid l[\varepsilon] \mid r[\varepsilon] \qquad \mathcal{G}_1 + \mathcal{G}_2 \\
 & & & \mid a(\varepsilon) \qquad \mathcal{G}^A \\
 & & & \mid \{\varepsilon\} \qquad \mathcal{P}\mathcal{G}
\end{array}
$$

$\mathcal{G}$-expressions

$$
\begin{array}{rclll}
\text{Exp} \ni \varepsilon & ::= & \underline{\emptyset} \mid \varepsilon \oplus \varepsilon & \mid \mu x.\gamma & \\
 & & & \mid b & B \\
 & & & \mid l\langle\varepsilon\rangle \mid r\langle\varepsilon\rangle & \mathcal{G}_1 \times \mathcal{G}_2 \\
 & & & \mid l[\varepsilon] \mid r[\varepsilon] & \mathcal{G}_1 + \mathcal{G}_2 \\
 & & & \mid a(\varepsilon) & \mathcal{G}^A \\
 & & & \mid \{\varepsilon\} & \mathcal{P}\mathcal{G}
\end{array}
$$

$\mathcal{G}$-expressions

$$
\begin{array}{llll}
\text{Exp} \ni \varepsilon & ::= & \underline{\emptyset} \mid \varepsilon \oplus \varepsilon & \mid \mu x.\gamma & \\
& & & \mid b & B \\
& & & \mid l\langle\varepsilon\rangle \mid r\langle\varepsilon\rangle & \mathcal{G}_1 \times \mathcal{G}_2 \\
& & & \mid l[\varepsilon] \mid r[\varepsilon] & \mathcal{G}_1 + \mathcal{G}_2 \\
& & & \mid a(\varepsilon) & \mathcal{G}^A \\
& & & \mid \{\varepsilon\} & \mathcal{P}\mathcal{G}
\end{array}
$$

Examples

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

$$\varepsilon \quad ::= \quad \underbrace{\underline{\emptyset} \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma}_{\mathcal{G}} \mid$$

Examples

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

$$\varepsilon \;::=\; \underbrace{\underline{\emptyset} \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma}_{\mathcal{G}} \mid \underbrace{l\langle \qquad \rangle \qquad\qquad \mid r\langle \qquad \rangle}_{\times}$$

Examples

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

$$\varepsilon \quad ::= \quad \underbrace{\underline{\emptyset} \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma}_{\mathcal{G}} \mid \underbrace{l\langle \underbrace{1}_{2} \rangle \mid l\langle \underbrace{0}_{2} \rangle \mid r\langle \underbrace{a(\varepsilon)}_{Id^A} \rangle}_{\times}$$

Examples

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

$$\varepsilon \quad ::= \quad \underbrace{\underline{\emptyset} \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma}_{\mathcal{G}} \mid \underbrace{l\langle \underbrace{1}_{2} \rangle \mid l\langle \underbrace{0}_{2} \rangle \mid r\langle \underbrace{a(\varepsilon)}_{Id^A} \rangle}_{\times}$$

LTS expressions – $\mathcal{G} = 1 + (\mathcal{P}Id)^A$

$$\varepsilon \quad ::= \quad \underline{\emptyset} \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma$$

Examples

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

$$\varepsilon \quad ::= \quad \underbrace{\underline{\emptyset} \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma}_{\mathcal{G}} \mid \underbrace{l\langle \underbrace{1}_{2} \rangle \mid l\langle \underbrace{0}_{2} \rangle \mid r\langle \underbrace{a(\varepsilon)}_{Id^A} \rangle}_{\times}$$

LTS expressions – $\mathcal{G} = 1 + (\mathcal{P}Id)^A$

$$\varepsilon \quad ::= \quad \underline{\emptyset} \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid \quad \surd \quad \mid \quad \partial \quad \mid \quad a.\varepsilon$$

Examples

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

$$\varepsilon \quad ::= \quad \underbrace{\underline{\emptyset} \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma}_{\mathcal{G}} \mid \underbrace{l\langle \underbrace{1}_{2} \rangle \mid l\langle \underbrace{0}_{2} \rangle \mid r\langle \underbrace{a(\varepsilon)}_{Id^A} \rangle}_{\times}$$

LTS expressions – $\mathcal{G} = 1 + (\mathcal{P}Id)^A$

$$\varepsilon \quad ::= \quad \underline{\emptyset} \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid \underbrace{\surd}_{l[*]} \mid \underbrace{\partial}_{r[\underline{\emptyset}]} \mid \underbrace{a.\varepsilon}_{r[a(\{\varepsilon\})]}$$

Operational semantics

The set of $\mathcal{G}$-expressions has a coalgebraic structure given by

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

$\delta_{\mathcal{G}}$ ...

- ...provides an operational semantics for the set of expressions
- ...defines the dynamics of the system
- ...is used for observing the behaviour of the system

Operational semantics

The set of $\mathcal{G}$-expressions has a coalgebraic structure given by

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

$\delta_{\mathcal{G}}$ ...

- ...provides an operational semantics for the set of expressions
- ...defines the dynamics of the system
- ...is used for observing the behaviour of the system

Operational semantics

The set of $\mathcal{G}$-expressions has a coalgebraic structure given by

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

$\delta_{\mathcal{G}}$ ...

- ...provides an operational semantics for the set of expressions
- ...defines the dynamics of the system
- ...is used for observing the behaviour of the system

Operational semantics

The set of $\mathcal{G}$-expressions has a coalgebraic structure given by

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

$\delta_{\mathcal{G}}$ ...

- ...provides an operational semantics for the set of expressions
- ...defines the dynamics of the system
- ...is used for observing the behaviour of the system

Example

$$\varepsilon \quad ::= \quad \underline{\emptyset} \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid \surd \mid \delta \mid a.\varepsilon$$

$\delta \colon \mathrm{Exp} \to 1 + (\mathcal{P}\mathrm{Exp})^A$

$$\vdots$$

$$\begin{array}{lcl} \delta(\surd) & = & \star \\ \delta(\partial) & = & \lambda a.\emptyset \\ \delta(a.\varepsilon) & = & \lambda a'.\begin{cases} \{\varepsilon\} & a = a' \\ \emptyset & oth. \end{cases} \end{array}$$

Example

$$\varepsilon \quad ::= \quad \underline{\emptyset} \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid \surd \mid \delta \mid a.\varepsilon$$

$\delta \colon \mathrm{Exp} \to 1 + (\mathcal{P}\mathrm{Exp})^A$

$\vdots$

$$\begin{array}{lcl}
\delta(\surd) & = & \star \\
\delta(\partial) & = & \lambda a.\emptyset \\
\delta(a.\varepsilon) & = & \lambda a'.\begin{cases} \{\varepsilon\} & a = a' \\ \emptyset & oth. \end{cases} \\
\delta(\varepsilon_1 \oplus \varepsilon_2) & = & \begin{cases} \star & \delta(\varepsilon_1) = \star \text{ and } \delta(\varepsilon_2) = \star \\ s_1 \cup s_2 & \delta(\varepsilon_1)(a) = s_1 \text{ and } \delta(\varepsilon_2)(a) = s_2 \\ \bot & \text{oth.} \end{cases}
\end{array}$$

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 by example (2.)

In the proof of ❷ lies the kernel of decidability of equivalence of expressions.

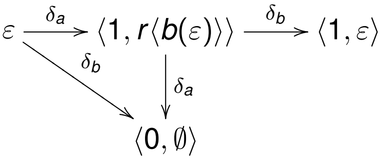
$$\varepsilon = \mu x.r\langle a(r\langle b(x)\rangle)\rangle \oplus l\langle 1\rangle$$

$$\varepsilon \xrightarrow{\delta_a} \langle 1, r\langle b(\varepsilon)\rangle\rangle \xrightarrow{\delta_b} \langle 1, \varepsilon\rangle$$

Proof by example (2.)

In the proof of ② lies the kernel of decidability of equivalence of expressions.

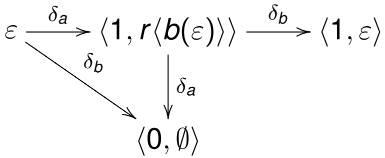
$$\varepsilon = \mu x.r\langle a(r\langle b(x)\rangle)\rangle \oplus l\langle 1\rangle$$

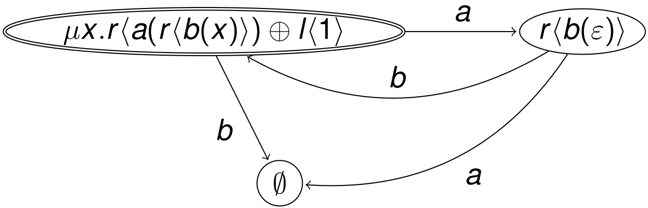


Proof by example (2.)

In the proof of 2 lies the kernel of decidability of equivalence of expressions.

$$\varepsilon = \mu x.r\langle a(r\langle b(x)\rangle)\rangle \oplus l\langle 1\rangle$$





Proof by example (2.)

It's all about unraveling! But ...

$$\varepsilon = \mu x.r\langle a(x \oplus x)\rangle$$

Proof by example (2.)

It's all about unraveling! But ...

$$\varepsilon = \mu x.r\langle a(x \oplus x)\rangle$$

$\varepsilon \overset{\delta}{\longmapsto} \langle 0, \varepsilon \oplus \varepsilon \rangle$

Proof by example (2.)

It's all about unraveling! But ...

$$\varepsilon = \mu x.r\langle a(x \oplus x)\rangle$$

$$\varepsilon \stackrel{\delta}{\longmapsto} \langle 0, \varepsilon \oplus \varepsilon\rangle \stackrel{\delta}{\longmapsto} \langle 0, (\varepsilon \oplus \varepsilon) \oplus (\varepsilon \oplus \varepsilon)\rangle \stackrel{\delta}{\longmapsto} \langle 0, (\varepsilon \oplus \varepsilon) \oplus (\varepsilon \oplus \varepsilon) \oplus (\varepsilon \oplus \varepsilon)\rangle \ldots$$

Proof by example (2.)

It's all about unraveling! But ...

$$\varepsilon = \mu x.r\langle a(x \oplus x)\rangle$$

$$\varepsilon \stackrel{\delta}{\longmapsto} \langle 0, \varepsilon \oplus \varepsilon\rangle \stackrel{\delta}{\longmapsto} \langle 0, (\varepsilon \oplus \varepsilon) \oplus (\varepsilon \oplus \varepsilon)\rangle \stackrel{\delta}{\longmapsto} \langle 0, (\varepsilon \oplus \varepsilon) \oplus (\varepsilon \oplus \varepsilon) \oplus (\varepsilon \oplus \varepsilon)\rangle \ldots$$

We need **ACI**!

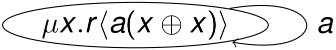
Proof by example (2.)

It's all about unraveling! But ...

$$\varepsilon = \mu x.r\langle a(x \oplus x)\rangle$$

$$\varepsilon \stackrel{\delta}{\longmapsto} \langle 0, \varepsilon \oplus \varepsilon\rangle \stackrel{\delta}{\longmapsto} \langle 0, (\varepsilon \oplus \varepsilon) \oplus (\varepsilon \oplus \varepsilon)\rangle \stackrel{\delta}{\longmapsto} \langle 0, (\varepsilon \oplus \varepsilon) \oplus (\varepsilon \oplus \varepsilon) \oplus (\varepsilon \oplus \varepsilon)\rangle \ldots$$

We need **ACI**!



Decision procedure for bisimilarity of regular expressions

Unraveling the expressions modulo ACI guarantees that only a finite number of states are reachable. The *bisimulation game* is then decidable!

Decision procedure for bisimilarity of regular expressions

Unraveling the expressions modulo ACI guarantees that only a finite number of states are reachable. The *bisimulation game* is then decidable!

CIRC

Coinductive prover based on algebraic specifications

language of expressions ($\mathcal{G}$-expressions)

coalgebraic structure ($\delta_{\mathcal{G}}$)

algebraic specification

Conclusions and Future work

Conclusions

- Framework to uniformly derive language and axioms for Kripke polynomial coalgebras
- Generalization of Kleene theorem and Kleene algebra, parametric on the functor.
- Automation in `Circ`: decision procedure for equivalence of expressions.

Future work

- Making the tool more user friendly;
- Apply it to a serious case study (circuit design, compiler optimization, . . .)

Conclusions and Future work

Conclusions

- Framework to uniformly derive language and axioms for Kripke polynomial coalgebras
- Generalization of Kleene theorem and Kleene algebra, parametric on the functor.
- Automation in `Circ`: decision procedure for equivalence of expressions.

Future work

- Making the tool more user friendly;
- Apply it to a serious case study (circuit design, compiler optimization, . . .)