Equational Theories for Real-Time Coalgebraic State Machines

Sergey Goncharov[a] Stefan Milius[a] Alexandra Silva[b]

ICTAC 2018, October 15-19, Stellenbosch

[a]Friedrich-Alexander-Universität Erlangen-Nürnberg
[b]University College London

Some Related Work

- Deterministic automata as coalgebras **[Rutten, 1998]**
- Generalized regular expressions and Kleene theorem for Kripke-polynomial functors **[Silva, 2010]**
- Generalized powerset construction **[Silva, Bonchi, Bonsangue, and Rutten, 2010]**
- Regular expressions for equationally presented functors and monads **[Myers, 2013]**
- Context-free languages, coalgebraically **[Winter, Bonsangue, and Rutten, 2013]**

This talk is based on

- Goncharov, Milius, and Silva 2014, Towards a Coalgebraic Chomsky Hierarchy (TCS 2014)
- Goncharov, Milius, and Silva 2018, Towards a Uniform Theory of Effectful State Machines (ArXiv preprint)

Overview

What's in here:

- A theory of effectful real-time state machines and expressions
- Semantics over the space of formal power series
- Equational theories for effects
- Kleene theorem

What isn't here:

- Results on the equational theory of fixpoint expressions, e.g. completeness
- Non-linear semantics (e.g. with trees instead of words)
- Infinite trace semantics
- Unguarded expressions/non-real-time machines (but see the paper)

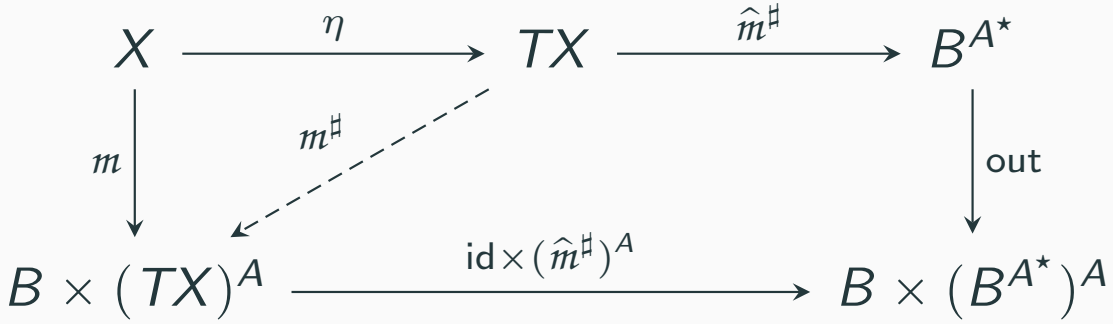
Prelude: Coalgebraic Powerset Construction

$\mathbb{T}$-automata and Generalized Powerset Construction

For a monad $\mathbb{T}$ and a $\mathbb{T}$-algebra $a^m : TB \to B$, we dub a triple of maps

$$o^m : X \to B, \qquad t^m : X \times A \to TX, \qquad a^m : TB \to B$$

a $\mathbb{T}$-automaton. Equivalently, it is a coalgebra $m : X \to B \times (TX)^A$



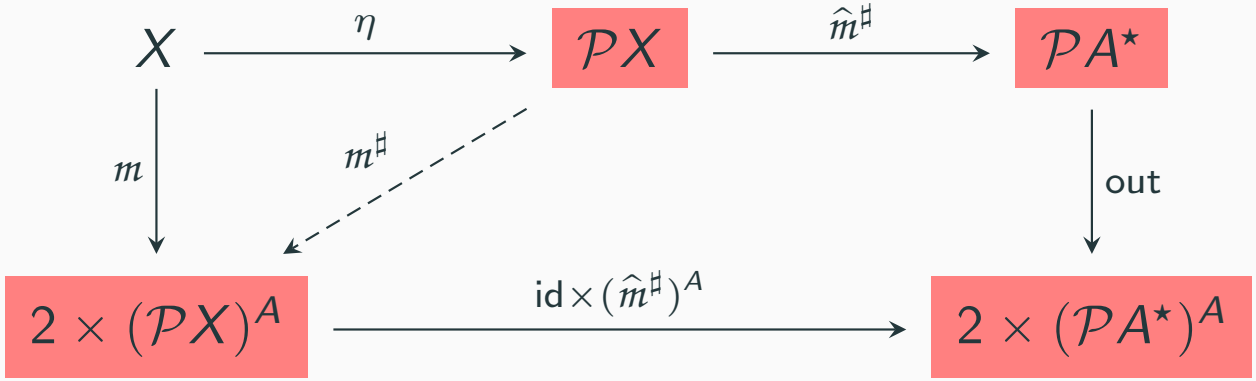
Factorization $m = m^\sharp \eta$ is unique and we define $[\![x]\!]_m = [\![\eta(x)]\!]_{m^\sharp} \in B^{A^\star}$, meaning that the formal power series $[\![x]\!]_m : A^\star \to B$ is the semantics of m in state x

$\mathbb{T}$-automata and Generalized Powerset Construction

For a monad $\mathbb{T}$ and a $\mathbb{T}$-algebra $a^m : TB \to B$, we dub a triple of maps

$$o^m : X \to B, \qquad t^m : X \times A \to TX, \qquad a^m : TB \to B$$

a $\mathbb{T}$-automaton. Equivalently, it is a coalgebra $m : X \to B \times (TX)^A$



Factorization $m = m^\sharp \eta$ is unique and we define $[\![x]\!]_m = [\![\eta(x)]\!]_{m^\sharp} \in B^{A^\star}$, meaning that the formal power series $[\![x]\!]_m : A^\star \to B$ is the semantics of m in state x

Towards Kleene Theorem

- We seek a syntactic counterpart of the generalized powerset construction
- Hence, we need a syntax for the corresponding fixpoint expressions, corresponding to the classical regular expressions
- Such expressions must include
  - reactive constructs for actions from A and final outputs from B
    (Think of prefixing $a.-$ and 0, 1 of Kleene algebra)
  - effectful constructs for representing side-effecting transitions of the corresponding $\mathbb{T}$-automaton
    (Think of nondeterministic $+$ and 0)

Equational Theories for Effects

Monads for Effects

Mac Lane: A monad $\mathbb{T}$ is just a monoid in the category of endofunctors

Monads for Effects

Mac Lane: A monad $\mathbb{T}$ is just a monoid in the category of endofunctors

Moggi: A monad $\mathbb{T}$ is a (generalized) computational effect

Monads for Effects

Moggi: A monad $\mathbb{T}$ is a (generalized) computational effect

Any monad $\mathbb{T}$ supports inclusion of a value into a computation $\eta : X \to TX$ and a Kleisli lifting $(f : X \to TY) \mapsto (f^\star : TX \to TY)$

Examples:

- (finitary) nondeterminism: $TX = \mathcal{P}_\omega X$; "nondeterministic functions" $A \to \mathcal{P}_\omega B$ are relations
- probabilistic nondeterminism: $TX = \{\rho : X \to [0,1] \mid \sum \rho = 1\}$; "probabilistic functions" $A \to TB$ are "probabilistic relations"
- (finite) background store: $TX = (X \times S)^S$; side-effecting functions $A \to TB$ are functions $A \times S \to B \times S$

Algebraic Theories

Moggi: A monad is a (generalized) computational effect

Algebraic Theories

Moggi: A monad is a (generalized) computational effect

Plotkin & Power: A monad is a (generalized) algebraic theory

Algebraic Theories

Plotkin & Power: A monad is a (generalized) algebraic theory

An algebraic theory $\mathcal{E}$ can be presented by a signature Σ and a set of equations. Any $\mathcal{E}$ defines a monad:

- $T_{\mathcal{E}}X =$ 'set of Σ-terms over X modulo $\mathcal{E}$'
- η coerces a variable to a term
- $\sigma^\star(t)$ applies substitution $\sigma : X \to T_{\mathcal{E}}Y$ to $t : T_{\mathcal{E}}X$

Example: Finite powerset monad $\mathcal{P}_\omega \iff$ join semilattices with bottom $\iff$ idempotent commutative monoids

Example: Finite probability distributions $\mathcal{D}_\omega \iff$ barycentric algebras; $\Sigma = \{+_p \mid p \in [0,1]\}$, satisfying e.g. "biased associativity":

$$(x +_p y) +_q z = x +_{p/(p+q-pq)} (y +_{p+q-pq} z)$$

Stack Monad

The store monad $TX = (\Gamma^\star \times X)^{\Gamma^\star}$ with $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ could be regarded as a monad for stack transformations. But it contains too much!

Definition: The stack monad is the submonad $\mathbb{T}_{stk}$ of the store monad $(- \times \Gamma^\star)^{\Gamma^\star}$ formed by $\langle r : \Gamma^\star \to X, t : \Gamma^\star \to \Gamma^\star \rangle$, which satisfy restriction:

$$\exists k.\, w \in \Gamma^k.\, \forall u \in \Gamma^\star.\; r(wu) = r(w) \wedge t(wu) = t(w)u$$

Intuitively, this ensures that the underlying stack may only be finitely read. E.g. nonexample: $\text{empty} = \langle \lambda - .\, \star, \lambda - .\, \epsilon \rangle \notin T_{stk}1$

The stack signature consists of unary $\mathbf{push}_i$ $(i \leqslant n)$ and $(n+1)$-ary $\mathbf{pop}$:

$$[\![\mathbf{push}_i]\!](p \in T_{stk}X)(w) = p(\gamma_i w)$$
$$[\![\mathbf{pop}]\!](p_1 \in T_{stk}X, \ldots, p_n \in T_{stk}X, q \in T_{stk}X)(\gamma_i w) = p_i(w)$$
$$[\![\mathbf{pop}]\!](p_1 \in T_{stk}X, \ldots, p_n \in T_{stk}X, q \in T_{stk}X)(\epsilon) = p_i(\epsilon)$$

Stack Theory

Theorem: The following stack theory is complete w.r.t. the stack monad

$$\begin{aligned}
\mathbf{push}_i(\mathbf{pop}(x_1, \ldots, x_n, y)) &= x_i \\
\mathbf{pop}(\mathbf{push}_1(x), \ldots, \mathbf{push}_n(x), x) &= x \\
\mathbf{pop}(x_1, \ldots, x_n, \mathbf{pop}(y_1, \ldots, y_n, z)) &= \mathbf{pop}(x_1, \ldots, x_n, z)
\end{aligned}$$

Moreover, each $T_{stk}X$ is the free algebra of the stack theory over X

Proof: Interpreting the axioms as a rewriting system (it is strongly normalizing, no non-trivial critical pairs) and identifying each $T_{stk}X$ with the set of normal forms

Adding Nondeterminism

Store-like effects can be sensibly combined by tensoring[1]: a tensor product of two theories $\mathcal{E}_1$ and $\mathcal{E}_2$ is obtaining by joining signatures and equations and adding the tensor laws: for all $f \in \mathcal{E}_1$, $g \in \mathcal{E}_2$

$$f(g(x_1^1, \ldots, x_k^1), \ldots g(x_1^n, \ldots, x_k^n)) = g(f(x_1^1, \ldots, x_1^n), \ldots f(x_k^n, \ldots, x_k^n))$$

Theorem [Freyd]: Tensor product of any theory with a semiring-module theory is again a semiring-module theory

This yields a complete axiomatization of $\mathcal{P}_\omega \otimes T_{stk}$:

$$u_i o_i = 1 \quad u_i o_j = 0 \quad u_i e = 0 \quad o_1 u_1 + \ldots + o_n u_n + e = 1 \quad e o_i = 0 \quad ee = e \quad (i \neq j)$$

where

$$e(x) = \mathbf{pop}(\emptyset, \ldots, \emptyset, x) \quad o_i(x) = \mathbf{pop}(\emptyset, \ldots, x, \ldots, \emptyset) \quad p_i(x) = \mathbf{push}(x)$$

[1] Freyd 1966, Algebra valued functors in general and tensor products in particular

(Turing) Tape Monad

The tape monad $\mathbb{T}_{tp}$ is analogously defined as a submonad of $(- \times \mathbb{Z} \times \Gamma^{\mathbb{Z}})^{\mathbb{Z} \times \Gamma^{\mathbb{Z}}}$ ($\mathbb{Z}$ = integer numbers)

Signature: **rd** (n-ary), $\mathbf{wr}_i$ (unary, $1 \leqslant i \leqslant n$), $\mathbf{mv}_1, \mathbf{mv}_{\text{-}1}$ (unary)

Equations:

$$\mathbf{mv}_{\text{-}1}(\mathbf{mv}_1(x)) = x$$

$$\mathbf{rd}(\mathbf{wr}_1(x), \ldots, \mathbf{wr}_n(x)) = x \qquad \mathbf{mv}_1(\mathbf{mv}_{\text{-}1}(x)) = x$$

$$\mathbf{wr}_i(\mathbf{rd}(x_1, \ldots, x_n)) = \mathbf{wr}_i(x_i) \qquad \mathbf{wr}_i(\mathbf{wr}_j(x)) = \mathbf{wr}_j(x)$$

$$\mathbf{wr}_i(\mathbf{mv}_k(\mathbf{wr}_j(\mathbf{mv}_{\text{-}k}(x)))) = \mathbf{mv}_k(\mathbf{wr}_j(\mathbf{mv}_{\text{-}k}(\mathbf{wr}_i(x)))) \quad (k \neq 0)$$

Proposition: Tape theory is not finitely axiomatizable

Combining Effects with Reactivity

Reactive Expressions: Syntax

Reactive expressions E_{Σ,B_0} are closed δ-expressions generated by the grammar for a given signature Σ and generators B_0 of a Σ-algebra B:

$$\begin{aligned}
\delta &::= x \mid \gamma \mid f(\delta, \ldots, \delta) && (x \in X, f \in \Sigma)\\
\gamma &::= \mu x.\ (a_1.\delta \pitchfork \cdots \pitchfork a_k.\delta \pitchfork \beta) && (x \in X, a_i \in A)\\
\beta &::= b \mid f(\beta, \ldots, \beta) && (b \in B_0, f \in \Sigma)
\end{aligned}$$

Example (Nondeterministic Automata) (using $1 + 1 = 1$ from B):

$$\delta ::= x \mid \gamma \mid \emptyset \mid \delta + \delta \qquad \gamma ::= \mu x.\ (a_1.\delta \pitchfork \cdots \pitchfork a_k.\delta \pitchfork 1)$$

Example (Deterministic PDA) (using $\mathbf{push}_i(1) = 0$ from B):

$$\begin{aligned}
\delta &::= x \mid \gamma \mid \mathbf{push}_i(\delta) \mid \mathbf{pop}(\delta, \ldots, \delta)\\
\gamma &::= \mu x.\ (a_1.\delta \pitchfork \cdots \pitchfork a_k.\delta \pitchfork \beta)\\
\beta &::= 0 \mid 1 \mid \mathbf{pop}(\beta, \ldots, \beta)
\end{aligned}$$

Reactive Expressions: Semantics

For $e \in \mathrm{E}_{\Sigma,B_0}$, we define

- Brzozowski derivatives $\partial_a(e) \in \mathrm{E}_{\Sigma,B_0}$ by induction, most notably

$$\partial_{a_i}\big(\mu x.\,(a_1.t_1 \pitchfork \cdots \pitchfork a_k.t_k \pitchfork r)\big) = t_i[e/x]$$

  and by further induction, $\partial_\epsilon(e) = e$, $\partial_{aw}(e) = \partial_a\partial_w(e)$
- final outputs $o(e) \in B$, again by induction, specifically

$$o\big(\mu x.\,(a_1.t_1 \pitchfork \cdots \pitchfork a_k.t_k \pitchfork r)\big) = r$$

Reactive Expressions: Semantics

For $e \in \mathrm{E}_{\Sigma, B_0}$, we define

- Brzozowski derivatives $\partial_a(e) \in \mathrm{E}_{\Sigma, B_0}$ by induction, most notably

$$\partial_{a_i}\big(\mu x.\,(a_1.t_1 \pitchfork \cdots \pitchfork a_k.t_k \pitchfork r)\big) = t_i[e/x]$$

  and by further induction, $\partial_\epsilon(e) = e$, $\partial_{aw}(e) = \partial_a \partial_w(e)$
- final outputs $o(e) \in B$, again by induction, specifically

$$o\big(\mu x.\,(a_1.t_1 \pitchfork \cdots \pitchfork a_k.t_k \pitchfork r)\big) = r$$

This induces the semantics $[\![e]\!] : A^\star \to B$, by putting $[\![e]\!](w) = o(\partial_w(e))$

Reactive Expressions: Semantics

For $e \in \mathrm{E}_{\Sigma,B_0}$, we define

- Brzozowski derivatives $\partial_a(e) \in \mathrm{E}_{\Sigma,B_0}$ by induction, most notably

$$\partial_{a_i}\big(\mu x.\,(a_1.t_1 \curlywedge \cdots \curlywedge a_k.t_k \curlywedge r)\big) = t_i[e/x]$$

  and by further induction, $\partial_\epsilon(e) = e$, $\partial_{aw}(e) = \partial_a\partial_w(e)$
- final outputs $o(e) \in B$, again by induction, specifically

$$o\big(\mu x.\,(a_1.t_1 \curlywedge \cdots \curlywedge a_k.t_k \curlywedge r)\big) = r$$

This induces the semantics $[\![e]\!] : A^\star \to B$, by putting $[\![e]\!](w) = o(\partial_w(e))$

Kleene Theorem: for any $e \in \mathrm{E}_{\Sigma,B_0}$ there is a $\mathbb{T}$-automaton $\mathcal{m}$ over X and a state $x \in X$ such that $[\![e]\!] = [\![x]\!]_{\mathcal{m}}$ and vice versa

Classes of Instances

- If $B = 2$, e.g. for non-deterministic or alternating automata, $[\![e]\!] \in 2^{A^\star} \cong \mathcal{P}A^\star$ is the formal language of e
- For weighted automata, B is a semiring, $\mathbb{T}$ is the corresponding B-module monad and we obtain standard semantics
- For PDA-like machines, $[\![e]\!]$ is not the ultimate semantics:

  Theorem: if B is finite then $[\![e]\!]$ is regular regardless of the monad

  We take B to be the quotient of $T_{stk}2$ by

  $$\mathbf{push}_i(0) = \mathbf{push}_i(1) = 0,$$

  equivalently, B consists of predicates $\Gamma^\star \to 2$ depending on a bounded portion of stack, hence, $[\![e]\!] : \Gamma^\star \to \mathcal{P}(A^\star)$
- For valence automata, $\mathbb{T} = \mathcal{P}_\omega \otimes (- \times M) = \mathcal{P}_\omega(- \times M)$, $B = \mathcal{P}_\omega(M)$ where M is a (polycyclic) monoid emulating storage mechanism

Expressivity of Stack Machines

Theorem: With m ranging over $\mathbb{T}_{stk}$-automata, x_0 over the respective state spaces and γ_0 over Γ, the sets

$$\left\{ w \in A^\star \mid [\![x_0]\!]_m(w)(\gamma_0) = 1 \right\}$$

exhaustively cover all real-time deterministic context-free languages

Theorem: the classes of languages recognized by $\mathcal{P}_\omega \otimes \mathbb{T}_{stk} \otimes \ldots \otimes \mathbb{T}_{stk}$, where m is the number of copies of $\mathbb{T}_{stk}$, are as follows:

1. All context-free languages if $m = 1$
2. All nondeterministic linear time languages if $m \geqslant 3$
3. The case $m = 2$ correspond to a language class properly between the above two

Proof: Reduction to results on real-time machines from 60-s

Further Work

- Reactive expressions come with a canonical equational calculus, which cannot always be complete (otherwise equivalence of PDA would be decidable). Can one derive completeness from constraints on the effect theory[2]?
- ϵ-elimination: for $\mathbb{T}$-automata over a given B, can we obtain a useful description of $\mathbb{T}'$-automata over B' such that every original non-real-time automaton is equivalent to a real-time automaton w.r.t. $\mathbb{T}'$ and B'? In particular, when $\mathbb{T} = \mathbb{T}'$ and $B = B'$? This is very hard already for valence automata[3]
- Identifying further language and complexity classes. How to describe the relation between the theory of effects and the complexity of languages recognized?

[2] Bonsangue, Milius, and Silva 2013, Sound and Complete Axiomatizations of Coalgebraic Language Equivalence

[3] Zetzsche 2013, Silent Transitions in Automata with Storage

Thank You for Your Attention!

References

Marcello M. Bonsangue, Stefan Milius, and Alexandra Silva. Sound and complete axiomatizations of coalgebraic language equivalence. *ACM Trans. Comput. Log.*, 14(1):7:1–7:52, 2013.

Peter Freyd. Algebra valued functors in general and tensor products in particular. *Colloq. Math.*, 14:89–106, 1966.

Sergey Goncharov, Stefan Milius, and Alexandra Silva. Towards a coalgebraic chomsky hierarchy. In *TCS'14*, volume 8705, pages 265–280. Springer, 2014.

Sergey Goncharov, Stefan Milius, and Alexandra Silva. Towards a uniform theory of effectful state machines. *CoRR*, abs/1401.5277, 2018. URL `http://arxiv.org/abs/1401.5277`.

References II

Robert Myers. *Rational Coalgebraic Machines in Varieties: Languages, Completeness and Automatic Proofs*. PhD thesis, Imperial College London, 2013.

Jan J. M. M. Rutten. Automata and coinduction (an exercise in coalgebra). In Davide Sangiorgi and Robert de Simone, editors, *CONCUR*, volume 1466 of *Lecture Notes in Computer Science*, pages 194–218. Springer, 1998.

Alexandra Silva. *Kleene coalgebra*. PhD thesis, Radboud Univ. Nijmegen, 2010.

Alexandra Silva, Filippo Bonchi, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Generalizing the powerset construction, coalgebraically. In *FSTTCS*, volume 8 of *LIPIcs*, pages 272–283, 2010.

References III

Joost Winter, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Coalgebraic characterizations of context-free languages. *LMCS*, 9(3), 2013.

Georg Zetzsche. Silent transitions in automata with storage. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, volume 7966 of *Lecture Notes in Computer Science*, pages 434–445. Springer, 2013.