Monoidal Computer III: A coalgebraic view of computational complexity

Dusko Pavlovic and Muzamil Yahia

CMCS 2018
Thessaloniki, April 2018
Outline

Introduction

Background: Monoidal computer

Approach: Coalgebraic view

Result: Complexity evaluators

Applications: Speedup, gap, approximation, ... coalgebraically

Work: One-way and trapdoor
Outline

Introduction

Background: Monoidal computer

Approach: Coalgebraic view

Result: Complexity evaluators

Applications: Speedup, gap, approximation, ... coalgebraically

Work: One-way and trapdoor
Coauthor

Muzamil Yahia
Message of the paper

Background

Coalgebra captures dynamics and stateful behavior

News

Coalgebra provides gauges to measure complexity
Outline

Introduction

Background: Monoidal computer

Definition and structure

Examples

Fundamental Theorem

Some consequences

Approach: Coalgebraic view

Result: Complexity evaluators

Applications: Speedup, gap, approximation, etc. coalgebraically
Cartesian closed category

\[ C(X, [A, B]) \cong \lambda^{AB}_X C(X \times A, B) \]
Monoidal computer

\[
C(X, [A, B]) \xrightarrow{\varepsilon_X^{AB}} C(X \times A, B) \xleftrightarrow{\lambda_X^{AB}} C^*(X, P) \xrightarrow{\gamma_X^{AB}} C(X \otimes A, B)
\]
Monadic monoidal computer

\[ C(X, [A, B]) \xrightarrow{\varepsilon_{X}^{AB}} \overset{\cong}{\xrightarrow{\lambda_{X}^{AB}}} C(X \times A, B) \]

\[ \mathcal{C}(X, \mathcal{P}) \xrightarrow{\gamma_{X}^{AB}} \mathcal{C}(X \times A, MB) \]

where \( M \) is a commutative monad on \( \mathcal{C} \).
Monadic monoidal computer coalgebraically

\[
\begin{align*}
C(X, [A, B]) & \overset{\varepsilon^A_X}{\cong} C(X \times A, B) \\
C(X, P) & \overset{\nu^A_X}{\rightarrow} C(X \times A, M(X \times B))
\end{align*}
\]

where \( M : C \to C \) is a commutative monad
### From CC to MC

<table>
<thead>
<tr>
<th>models</th>
<th>static</th>
<th>dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>extensional</strong></td>
<td>$[A, B] \times A \xrightarrow{\varepsilon} B$</td>
<td>$[A^+, B] \times A$</td>
</tr>
<tr>
<td><strong>models</strong></td>
<td>$\exists f : X \times A \xrightarrow{\forall f}$</td>
<td>$\exists [q] : X \times A \xrightarrow{\forall q}$</td>
</tr>
<tr>
<td><strong>Cartesian</strong></td>
<td>abstractions $\xrightarrow{\varepsilon}$ $\leftarrow \lambda$ applications</td>
<td>behaviors $\xrightarrow{[-]}$ $\leftarrow$ machines</td>
</tr>
<tr>
<td><strong>Closed</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>intensional</strong></td>
<td>$P \otimes A \rightarrow {} B$</td>
<td>$P \otimes B$</td>
</tr>
<tr>
<td><strong>models</strong></td>
<td>$\exists F : X \times A \xrightarrow{\forall f}$</td>
<td>$\exists Q : X \times A $</td>
</tr>
<tr>
<td><strong>Monoidal</strong></td>
<td>programs $\rightarrow$ computations</td>
<td>adaptive prog’s $\rightarrow$ processes</td>
</tr>
<tr>
<td><strong>Computers</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# From CC to MMC

<table>
<thead>
<tr>
<th>models</th>
<th>static</th>
<th>dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>extensional models:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cartesian Closed</td>
<td><img src="extensional_diagram.png" alt="Diagram" /></td>
<td><img src="dynamic_diagram.png" alt="Diagram" /></td>
</tr>
<tr>
<td>abstractions $\xrightarrow{\varepsilon}$ applications</td>
<td><img src="extensional_diagram.png" alt="Diagram" /></td>
<td><img src="dynamic_diagram.png" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>intensional models:</strong></td>
<td><img src="intensional_diagram.png" alt="Diagram" /></td>
<td><img src="dynamic_diagram.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Monadic Monoidal Computers</td>
<td><img src="intensional_diagram.png" alt="Diagram" /></td>
<td><img src="dynamic_diagram.png" alt="Diagram" /></td>
</tr>
<tr>
<td>programs $\rightarrow$ computations</td>
<td><img src="intensional_diagram.png" alt="Diagram" /></td>
<td><img src="dynamic_diagram.png" alt="Diagram" /></td>
</tr>
<tr>
<td>adaptive prog's $\rightarrow$ processes</td>
<td><img src="intensional_diagram.png" alt="Diagram" /></td>
<td><img src="dynamic_diagram.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>
MC structure

Proposition

The following structures are equivalent

a) $X$-natural transformation $\mathcal{C}(X, \mathcal{P}) \xrightarrow{\gamma^A_B} \mathcal{C}(X \times A, MB)$

b) A universal evaluator $\{\} \in \mathcal{C}(\mathcal{P} \times A, MB)$, such that

$\forall g \in \mathcal{C}(X \times A, MB) \exists G \in \mathcal{C}(X, \mathcal{P})$

$$g(x, a) = \{ G(x) \} a$$

\[ B \]
\[ g \]
\[ X \]
\[ A \]

\[ X \]
\[ A \]

\[ B \]
\[ \{ \} \]

\[ G \]
MC structure

Proposition

The following structures are equivalent

a) $X$-natural transformation $\mathbb{C}(X, P) \xrightarrow{\gamma_{X}^{AB}} \mathbb{C}(X \times A, MB)$

b) a universal evaluator $\{\}^{AB} \in \mathbb{C}(P \times A, MB)$, and a partial evaluator $[]^{(AB)C} \in \mathbb{C}(P \times A, P)$, such that

$$\forall f \in \mathbb{C}(A, MB) \exists F \in \mathbb{C}(X, P)$$

$$f(a) = \{F\}a \quad \{G\}(a, b) = \{[G]a\}b$$

\[\begin{array}{c}
\text{f} \\
A \\
\end{array} = \begin{array}{c}
\{\} \\
F \\
A \\
\end{array} \quad \begin{array}{c}
\{\} \\
A \\
\end{array} = \begin{array}{c}
[] \\
P \\
A \\
B \\
\end{array}
\]
MC structure

Overview

\[ g(x, y) = \{G\}(x, y) = \{[G] x\}y \]
MC structure

Definition

A monoidal computer (MC) is a

- monoidal category $\mathbb{C}$ with
- commutative comonoids $A \otimes A \xleftrightarrow{\Delta} A \xrightarrow{T} I$ for all $A$
- a distinguished type $\mathbb{P}$ of programs
- equivalent structures from the Proposition.
 MMC structure

Definition

An *monadic monoidal computer (MMC)* is a

- cartesian category $\mathbb{C}$ with a
- commutative monad $M : \mathbb{C} \to \mathbb{C}$
- a distinguished type $\mathbb{P}$ of programs
- equivalent structures from the Proposition.
Examples?
Any computer is monoidal

\[ \mathbb{D} = 2^* \quad \text{(binaries)} \]

\[ \mathbb{D} \xleftarrow{\text{run}} \mathbb{D} \times \mathbb{D} \]

\[ \mathbb{D} \xrightarrow{\langle \kappa_0, \kappa_1 \rangle} \mathbb{D} \times \mathbb{D} \]

\[ \mathbb{D} \xrightarrow{\mathbb{D} \times \mathbb{D}} \]
Computable universe $\mathcal{C}$

- **types**: sets where each element is tagged by some $\mathcal{D}$-labels

  \[
  |\mathcal{C}| = \bigsqcup_{A \in |S|} \left\{ A \xleftrightarrow{\rho_A} |A| \subseteq \mathcal{D} \right\}
  \]

- **morphisms**: functions that are traced on the tags by $\mathcal{D}$-implementable computations

  \[
  \mathcal{C}(A, B) = \left\{ f \in S(A, B) \mid \exists F \in \mathcal{D}. \begin{array}{c}
  |X| \xrightarrow{\text{run}(F)} |B| \\
  |A| \xleftarrow{\rho_A} |B| \xrightarrow{\rho_B} |B| \\
  A \xrightarrow{f} B
  \end{array} \right\}
  \]
Computable monads: Maybe

\[ ? : \mathbb{C} \rightarrow \mathbb{C} \]

\[ A \mapsto \left( 1 + A \xleftarrow{\text{run}(-,0)} \left| ?A \right| \right) \]

where

\[ \left| ?A \right| = \left\{ F \in \mathbb{D} \mid \text{run}(F, 0) \downarrow \implies \text{run}(F, 0) \in A \right\} \]
Computable monads: Power

\[ \mathcal{P} : \mathbb{C} \rightarrow \mathbb{C} \]

\[ A \mapsto \left( \mathcal{P}A \leftarrow |\mathcal{P}A| \right) \]

where

\[ |\mathcal{P}A| = \left\{ F \in \mathbb{D} \mid \forall a \in A. \ run(F, a) \downarrow \land run(F, a) \in \{0, 1\} \right\} \]

\[ \mathcal{P}A = |\mathcal{P}A|/\equiv \text{ where} \]

\[ F \equiv G \iff run(F) = run(G) \]
The Fundamental Theorem of Computability

Theorem

For every computation $g \in \mathbb{C}(P \times A, MB)$ there is a program $\Gamma \in \mathbb{C}(P)$ such that

$$g(\Gamma, a) = \{\Gamma\} a$$

$\Gamma$ is Kleene’s fixed point of $g$. 
The Fundamental Theorem of Computability

Proof

\[ g(\Gamma, a) = g([G, G], a) = \{G\}(G, a) = \{[G, G]\}a = \{\Gamma\}a \]
Complete MCs

Proposition about idempotents

An MC is finitely complete and cocomplete if and only if it is Cauchy complete, i.e. if the idempotents split in it.
Computability monads are partial

Proposition about partiality

Every MC contains partial maps.
Computability monads are partial

Proposition about partiality

Every MC contains partial maps.

If there is an MMC over a monad $M : \mathbb{C} \to \mathbb{C}$, then

$? \subseteq M$
Natural numbers in MCs

Branching

\[ \text{if}(b, x, y) = \begin{cases} 
  x & \text{if } b = \top \\
  y & \text{if } b = \bot 
\end{cases} \]

\[ \begin{array}{c}
\text{if} \\
\top \\
\end{array} = \begin{array}{c}
\text{if} \\
\bot \\
\end{array} \]

\[ \begin{array}{c}
\text{if} \\
\bot \\
\end{array} = \begin{array}{c}
\text{if} \\
\bot \\
\end{array} \]
Natural numbers in MCs

Pairing

Define $\langle -, - \rangle : \mathcal{P} \times \mathcal{P} \rightarrow \mathcal{P}$

where

$\{ - \}$

$\text{if}$
Natural numbers in MCs

Projections

Define \((-)_0, (-)_1 : \mathcal{P} \rightarrow \mathcal{P}\)

\(\begin{align*}
(-)_0 & = \langle - \rangle_0 \\
(-)_1 & = \langle - \rangle_1
\end{align*}\)
Natural numbers in MCs

Numbers as programs

\[
\begin{align*}
\{ - \} & \quad = \quad \langle -, - \rangle \\
\overline{0} & \quad = \quad \langle -, - \rangle \\
\overline{n+1} & \quad = \quad \langle -, - \rangle \\
\end{align*}
\]
Natural numbers in MCs

Successor, predecessor, and zero test

\[ \mathcal{S} = \langle -, - \rangle \]
\[ \rho = (-)_1 \]
\[ 0? = (-)_0 \]
Natural numbers in MCs

\( \mathbb{N} \) as idempotent splitting

The type \( \mathbb{N} \) can be defined as the domain of \( \rho \).
Natural numbers in MCs

Proposition about $\mathbb{N}$

Every complete MC has a natural numbers object.
Outline

Introduction

Background: Monoidal computer

Approach: Coalgebraic view

Result: Complexity evaluators

Applications: Speedup, gap, approximation, ... coalgebraically

Work: One-way and trapdoor
Step computations

Definition

An *M-step computation* from $A$ to $B$ is a morphism

\[ X \times A \xrightarrow{q} M(X \times B) \]

in an MMC $\mathcal{C}$ with a commutative monad $M$, where

- $A$ is the type of inputs
- $B$ is the type of outputs
- $X$ is the state space
Step computations

Definition

A *step evaluation* of the $M$-step computation $X \times A \xrightarrow{q} M(X \times B)$ in another $M$-step computation $Y \times A \xrightarrow{r} M(Y \times B)$, is a morphism $f \in \mathcal{C}(X, Y)$ with

\[
\begin{array}{c}
Y \\
\downarrow \\
X \\
\downarrow \\
A
\end{array} \xrightarrow{f} \begin{array}{c}
Y \\
\downarrow \\
X \\
\downarrow \\
A
\end{array}
= \begin{array}{c}
Y \\
\downarrow \\
X \\
\downarrow \\
A
\end{array} \xrightarrow{q} \begin{array}{c}
Y \\
\downarrow \\
X \\
\downarrow \\
A
\end{array}
\]

where $B$ is an object, $f \in \mathcal{C}(X, Y)$, $M(Y \times B)$, and $M(f \times B)$. The diagram commutes in the category $\mathcal{C}$. The morphisms $r$ and $q$ represent the step evaluations in the respective computations.
Step computations

Definition

A step evaluation of the $M$-step computation $X \times A \xrightarrow{q} M(X \times B)$ in another $M$-step computation $Y \times A \xrightarrow{r} M(Y \times B)$, is a morphism $f \in C(X, Y)$ with $B^r f X = B^q f X$.

$\mathbb{C}_{MAB}$ denotes the category of $M$-step computations from $A$ to $B$ with step evaluations as morphisms.
Universal state space

Definition

A type $S$ in an MMC $C$ is a universal state space if for every $A$ and $B$ there is a weakly final $M$-step computation

$$\exists \{\} \in C(S \times A, M(S \times B))$$

\[ S \quad B \]
\[ \exists \{\} \]
\[ X \quad A \]

\[ S \quad B \]
\[ \forall q \]
\[ X \quad A \]

\[ M(S \times B) \]
\[ M(Q \times B) \]

\[ S \times A \]
\[ \exists Q \times A \]
\[ X \times A \]
\[ \forall q \]
Universal step evaluator

Definition

A type $S$ in an MMC $C$ is a *universal state space* if for every $A$ and $B$ there is a weakly final $M$-step computation

$\{\} \in C(S \times A, M(S \times B))$

This weakly final step computation $\{\}$ is a *universal step evaluator*. 
Monoidal computer coalgebraically

Proposition

A cartesian category $\mathcal{C}$ with a commutive monad $M$ is an MMC if and only if it has universal step evaluators.
Monoidal computer coalgebraically

Proposition

A cartesian category $\mathbb{C}$ with a commutative monad $M$ is an MMC if and only if it has universal step evaluators.

There is a bijective correspondence between the families

- universal evaluators $\{\} \in \mathbb{C}(P \times A, MB)$
- universal step evaluators $\{\|\} \in \mathbb{C}(S \times A, M(S \times B))$

indexed over $A$ and $B$ in $\mathbb{C}$. 
Monoidal computer coalgebraically

Proposition

A cartesian category $\mathcal{C}$ with a commutative monad $M$ is an MMC if and only if it has universal step evaluators.

There is a bijective correspondence between the families

- universal evaluators $\{\cdot\} \in \mathcal{C}(P \times A, MB)$
- universal step evaluators $\{\|$ \in \mathcal{C}(S \times A, M(S \times B))$

indexed over $A$ and $B$ in $\mathcal{C}$.

The types $P$ and $S$ can be taken to coincide.
Monoidal computer coalgebraically

Proof (1)

Given

- universal evaluator \( \{} \in C(\mathcal{P} \times A, MB) \)
- partial evaluator \([\] \in C(\mathcal{P} \times A, \mathcal{P}) \) derived from \( \{} \),
- a step computation \( q \in C(X \times A, M(X \times B)) \)
Monoidal computer coalgebraically

Proof (2)

...construct a Kleene fixed point $\hat{Q}$ of $\hat{q}$
Monoidal computer coalgebraically

Proof (3)

The step evaluation $Q \in \mathbb{C}(X, P)$ is $Q(x) = \hat{Q} x$. 
Outline

Introduction

Background: Monoidal computer

Approach: Coalgebraic view

Result: Complexity evaluators

Beyond time and space

Kleene’s normal form

Recursion and approximation theorems

Applications: Speedup, gap, approximation, ... coalgebraically
Ideas beyond Turing machine complexity

- **Kleene’s normal form**: traces as a hardness measure

- **Cobham’s intrinsic hardness**: it should only depend on functions, not on models

- **Blum’s abstract complexity**: machine independent complexity through *step counting*
  - time, space, alternations…
  - Kleene’s trace
  - project to universal evaluators!
A *complexity evaluator* is a computation\( \Xi^{AB} \in \mathcal{C}(P \times A, M(N \times B)) \), such that

\[ \Xi \text{ is a universal evaluator} \]

\[ \{ \bot, \top \} \]

\[ \leq n \]

\[ \Xi \text{ is decidable} \]
Step-counting functions

Definition

A *step-counting function* $\Phi^{AB} \in C(P \times A, MN)$ is the first projection of a complexity evaluator $\Xi^{AB} \in C(P \times A, M(N \times B))$
Step-counting functions project to evaluators

Theorem 1

There is a family of step-counting functions $K$ such that

a) for every step-counting function $\Phi$ there is a primitive recursive function $f$ with

$$\Phi = f \circ K$$

b) for every universal evaluator $\{-\}$ there is a primitive recursive function $u$ with

$$\{-\} = u \circ K$$
Complexity evaluators

Theorem 2

Every complete MC contains all step counting functions.
Not presented
Complexity by Coalgebra

DP and MY

Intro

CC to MC

MMC to CMC

Step counting

Speedup etc

Work

Outline

Introduction

Background: Monoidal computer

Approach: Coalgebraic view

Result: Complexity evaluators

Applications: Speedup, gap, approximation, ... coalgebraically

Work: One-way and trapdoor
Not presented
Outline

Introduction

Background: Monoidal computer

Approach: Coalgebraic view

Result: Complexity evaluators

Applications: Speedup, gap, approximation, ... coalgebraically

Work: One-way and trapdoor
Work

Question

Why yet another model of computation?
Work

Question

Why yet another model of computation?

Answer

Why is it that

- practice of computation has been revolutionized by high level programming languages, but
- theory of computation is still confined to low level machine languages

?
Work

Task 1

Teach computability and complexity to 2nd year students: http://www.asecolab.org/courses/ics-222/

Task 2

Develop a reasonable theory of *absolute one-way functions*.