Algebraic Enriched Coalgebras

Filippo Bonchi\(^4\)  Marcello Bonsangue\(^{1,2}\)  Jan Rutten\(^{1,3}\)
Alexandra Silva\(^1\)

\(^1\)Centrum Wiskunde en Informatica
\(^2\)LIACS - Leiden University
\(^3\)Radboud Universiteit Nijmegen
\(^4\)INRIA Saclay - LIX, École Polytechnique

Coalgebra Day, March 2010
Motivation (by example)

- Coalgebras are a suitable framework to study the behaviour of dynamical systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.

Motivation (by example)

- Coalgebras are a suitable framework to study the **behaviour** of dynamical systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.

  *J.J.M.M. Rutten. **Automata and coinduction (an exercise in coalgebra)**. CONCUR’98*
Motivation (by example)

- Coalgebras are a suitable framework to study the **behaviour** of dynamical systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.

Motivation (by example)

- Coalgebras are a suitable framework to study the behaviour of dynamical systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.

Motivation (by example)

- Coalgebras are a suitable framework to study the behaviour of dynamical systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.

Motivation (by example)

- Coalgebras are a suitable framework to study the behaviour of dynamical systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.

Coalgebras are a suitable framework to study the behaviour of dynamical systems.

Much of the coalgebraic approach can be nicely illustrated with deterministic automata.

Motivation (by example, cont.)

$q \neq q'$ (different branching structure)

$L_q = L_{q'} = \{ab, ac\}$

How do we study NDA wrt language equivalence?

Turn a non-deterministic automaton into a deterministic one via the
powerset construction and then apply usual semantics.
Motivation (by example, cont.)

How do we study NDA wrt language equivalence?

Turn a non deterministic automaton into a deterministic one via the powerset construction and then apply usual semantics.

\[ \{a, b, c\} \]

\[ \delta(q, a) = q' \]

\[ \delta(q, b) = q \]

\[ \delta(q, c) = q \]

\[ L_q = L_{q'} = \{a, b, c\} \]
Motivation (by example, cont.)

\[ q \neq [q'] \text{ (different branching structure)} \]

but:

\[ L_q = L_{q'} = \{ ab, ac \} \]

How do we study NDA wrt language equivalence?

Turn a non deterministic automaton into a deterministic one via the powerset construction and then apply usual semantics.

Alexandra Silva (CWI)
Motivation (by example, cont.)

\[ [q] \neq [q'] \] (different branching structure)
\[ [q] \neq [q'] \] (different branching structure) but: \( L_q = L_{q'} = \{ab, ac\} \)
Motivation (by example, cont.)

\[ [q] \neq [q'] \] (different branching structure) but: \( L_q = L_{q'} = \{ab, ac\} \)

How do we study NDA wrt language equivalence?
Motivation (by example, cont.)

\[
[q] \neq [q'] \text{ (different branching structure) but: } L_q = L_{q'} = \{ab, ac\}
\]

How do we study NDA wrt language equivalence?

Turn a non deterministic automaton into a deterministic one via the powerset construction and then apply usual semantics.
Example I: Determinizing (coalgebraically)

How do we study NDA wrt language equivalence?

\[ L_s = \left[ \begin{array}{c} \{s\} \end{array} \right] \]
Example I: Determinizing (coalgebraically)

\[ o(Q) = \begin{cases} 
1 & \exists q \in Q, o(q) = 1 \\
0 & \text{otherwise} 
\end{cases} \]

\[ t(Q)(a) = \bigcup_{q \in Q} t(q)(a) \]
Example I: Determinizing (coalgebraically)

\[
\overline{o}(Q) = \begin{cases} 
1 & \exists q \in Q \; o(q) = 1 \\
0 & \text{otherwise}
\end{cases}
\]

\[
\overline{t}(Q)(a) = \bigcup_{q \in Q} t(q)(a)
\]

How do we study NDA wrt language equivalence?

\[L_s = \left[ \left\{ S \right\} \right]\]
Example I: Determinizing (coalgebraically)

\[ \overline{o}(Q) = \begin{cases} 
1 & \exists q \in Q \ o(q) = 1 \\
0 & \text{otherwise}
\end{cases} \]

\[ \overline{t}(Q)(a) = \bigcup_{q \in Q} t(q)(a) \]

How do we study NDA wrt language equivalence?

\[ L_s = \llbracket \{ s \} \rrbracket \]
Motivation (by example, cont.)

$L_2$ are pairs of languages $\langle V, W \rangle$ (accepted words, domain)

$[q] = \langle c^*ab^*, b^*c^* + c^*ab^* \rangle \neq \langle c^*ab^*, c^* + c^*ab^* \rangle = [q']$

but:

$L_q = L_{q'} = c^*ab^*$

How do we study PA wrt (accepted) language equivalence?

Turn a partial automaton into a total deterministic one by adding a sink state and then apply usual semantics.
Motivation (by example, cont.)

$L_2$ are pairs of languages $\langle V, W \rangle$ (accepted words, domain)

$\text{q} = \langle c^*ab^*, b^*c^* + c^*ab^* \rangle \neq \langle c^*ab^*, c^* + c^*ab^* \rangle = \text{q'}$

How do we study PA wrt (accepted) language equivalence?

Turn a partial automaton into a total deterministic one by adding a sink state and then apply usual semantics.
Motivation (by example, cont.)

Let \( L \) be pairs of languages \( \langle V, W \rangle \) (accepted words, domain).

\[ \begin{align*}
q & = \langle c^*ab^*, b+c^*ab^* \rangle \\
q' & = \langle c^*ab^*, c^*+c^*ab^* \rangle
\end{align*} \]

but:

\[ L_q = L_{q'} = c^*ab^* \]

How do we study PA wrt (accepted) language equivalence?

Turn a partial automaton into a total deterministic one by adding a sink state and then apply usual semantics.

Alexandra Silva (CWI)
Algebraic Enriched Coalgebras
Coalgebra Day, March 2010
5 / 12
Motivation (by example, cont.)

\( L_2 \) are pairs of languages \( \langle V, W \rangle \) (\(<\text{accepted words, domain}>\))

\[
\begin{align*}
\lfloor q \rfloor &= \langle c^* ab^*, b + c^* + c^* ab^* \rangle \\
\neq \langle c^* ab^*, c^* + c^* ab^* \rangle = \lfloor q' \rfloor
\end{align*}
\]
$L_2$ are pairs of languages $\langle V, W \rangle$ (accepted words, domain)

$\llbracket q \rrbracket = \langle c^* ab^*, b + c^* + c^* ab^* \rangle \neq \langle c^* ab^*, c^* + c^* ab^* \rangle = \llbracket q' \rrbracket$

but: $L_q = L_{q'} = c^* ab^*$
Motivation (by example, cont.)

$L_2$ are pairs of languages $\langle V, W \rangle$ (<accepted words, domain>)

$$[q] = \langle c^* ab^*, b + c^* + c^* ab^* \rangle \neq \langle c^* ab^*, c^* + c^* ab^* \rangle = [q']$$

but: $L_q = L_{q'} = c^* ab^*$

How do we study PA wrt (accepted) language equivalence?
Motivation (by example, cont.)

$L_2$ are pairs of languages $\langle V, W \rangle$ (<accepted words, domain>)

\[ \llbracket q \rrbracket = \langle c^* ab^*, b + c^* + c^* ab^* \rangle \neq \langle c^* ab^*, c^* + c^* ab^* \rangle = \llbracket q' \rrbracket \]

but: $L_q = L_{q'} = c^* ab^*$

How do we study PA wrt (accepted) language equivalence?

Turn a partial automaton into a total deterministic one by adding a sink state and then apply usual semantics.
Example II: Totalizing (coalgebraically)

\[
S \xrightarrow{\langle o, t \rangle} 2 \times (1 + S)^A
\]
Example II: Totalizing (coalgebraically)

\[ \bar{o}(\ast) = 0 \]
\[ \bar{o}(s) = o(s) \]
\[ \bar{t}(\ast)(a) = \ast \]
\[ \bar{t}(s)(a) = t(s)(a) \]

How do we study PA wrt language equivalence?

\[ L_s = \left[ i(s) \right] \]
Example II: Totalizing (coalgebraically)

\[
\begin{align*}
\delta(\ast) & = 0 \\
\delta(s) & = \sigma(s) \\
\bar{t}(\ast)(a) & = \ast \\
\bar{t}(s)(a) & = t(s)(a)
\end{align*}
\]
Example II: Totalizing (coalgebraically)

\[
\begin{align*}
\bar{o}(\ast) &= 0 \\
\bar{o}(s) &= o(s) \\
\bar{t}(\ast)(a) &= * \\
\bar{t}(s)(a) &= t(s)(a)
\end{align*}
\]

How do we study PA wrt language equivalence?

\[L_s = \llbracket i(s) \rrbracket\]
Chasing the pattern...

How do we capture both examples (and more) in the same framework?
Chasing the pattern . . .

How do we capture both examples (and more) in the same framework?

The state space was enriched: $T$ monad $(\mathcal{P}, 1^+, \ldots)$.
Chasing the pattern . . .

How do we capture both examples (and more) in the same framework?

The state space was enriched: $T$ monad ($\mathcal{P}, 1+, \ldots$).
Transform an $FT$-coalgebra $(X, f)$ into an $F$-coalgebra ($T(X), f^\#$).
How do we capture both examples (and more) in the same framework?

The state space was \textit{enriched}: $T$ monad ($\mathcal{P}$, $1^+$, $\ldots$).

Transform an $FT$-coalgebra $(X, f)$ into an $F$-coalgebra $(T(X), f^\#)$.

If $F$ has final coalgebra: $x_1 \simeq_F^T x_2 \iff \llbracket \eta_X(x_1) \rrbracket = \llbracket \eta_X(x_2) \rrbracket$. 

In a nutshell... 

Ingredients:
- A monad $T$;
- A final coalgebra for $F$ (for instance, take $F$ to be bounded);
- An extension $f^\#$ of $f$;
In a nutshell... 

Ingredients:
- A monad $T$;
- A final coalgebra for $F$ (for instance, take $F$ to be bounded);
- An extension $f^\#$ of $f$; We can require $FT(X)$ to be a $T$-algebra:

$$(FT(X), h: T(FT(X)) \rightarrow FT(X))$$

$$f^\#: T(X) \xrightarrow{T(f)} T(F(T(X))) \xrightarrow{h} F(T(X))$$
Examples revisited

NFA  \( F(X) = 2 \times X^A, \ T = \mathcal{P}, \ 2 \times \mathcal{P}(X)^A \) is a join-semilattice;

PA  \( F(X) = 2 \times X^A, \ T = 1 + -, \ 2 \times (1 + X)^A \) is a pointed set.

What is the relation between \( \approx_T^F \) and \( \sim_F \)?
Examples revisited

\( F(X) = 2 \times X^A, \ T = \mathcal{P}, \ 2 \times \mathcal{P}(X)^A \) is a join-semilattice;
\( F(X) = 2 \times X^A, \ T = 1 + -, \ 2 \times (1 + X)^A \) is a pointed set.

What is the relation between \( \approx_T \) and \( \sim_F \)?
Examples revisited

NFA $F(X) = 2 \times X^A$, $T = \mathcal{P}$, $2 \times \mathcal{P}(X)^A$ is a join-semilattice;

PA $F(X) = 2 \times X^A$, $T = 1 + −$, $2 \times (1 + X)^A$ is a pointed set.

What is the relation between $\approx_T^F$ and $\sim_F$?
Examples revisited

NFA \( F(X) = 2 \times X^A, \; T = \mathcal{P}, \; 2 \times \mathcal{P}(X)^A \) is a join-semilattice.

PA \( F(X) = 2 \times X^A, \; T = 1 + -, \; 2 \times (1 + X)^A \) is a pointed set.

What is the relation between \( \approx_F^T \) and \( \sim_F \)?
Theorem

\[ \sim_F \Rightarrow \approx^T_F \]
Bisimilarity implies linear bisimilarity

Theorem

\[ \sim_F \Rightarrow \approx^T_F \]

The above theorem instantiates to well known facts:

- for NDA \((F(X) = 2 \times X^A, T = \mathcal{P})\) that bisimilarity implies language equivalence;

- for PA \((F(X) = 2 \times X^A, T = 1 + \_\_)\) that equivalences of pair of languages, consisting of defined paths and accepted words, implies equivalence of accepted words;

- for probabilistic automata \((F(X) = [0, 1] \times X^A, T = D_\omega)\) that probabilistic bisimilarity implies weighted language equivalence.
Examples, Examples, Examples, . . .

- Partial Mealy machines $S \rightarrow (B \times (1+S))^A$;
- Automata with exceptions $S \rightarrow 2 \times (E+S)^A$;
- Automata with side effects $S \rightarrow E^E \times ((E \times S)^E)^A$;
- Total subsequential transducers $S \rightarrow O^* \times (O^* \times S)^A$;
- Probabilistic automata $S \rightarrow [0, 1] \times (\mathcal{D}_\omega(X))^A$;
- Weighted automata $S \rightarrow \mathbb{R} \times (\mathbb{R}_\omega^X)^A$;
- . . .
Lifted *powerset construction* to the more general framework of *FT*-coalgebras;

Uniform treatment of several types of automata, recovery of known constructions/results;

Opens the door to the study of *linear equivalences* for many types of automata.

Thanks!!
Lifted \textit{powerset construction} to the more general framework of \textit{FT}-coalgebras;

Uniform treatment of several types of automata, recovery of known constructions/results;

Opens the door to the study of \textit{linear equivalences} for many types of automata.

Thanks!!