Algebra-Coalgebra Duality: applications in automata theory

Alexandra Silva

Radboud Universiteit Nijmegen and CWI

TbiLLC 2013: Tenth International Tbilisi Symposium on Language, Logic and Computation
The global message

- Two views on many problems: Algebra and coalgebra.
- The combination is essential!
- Coalgebra is semantics but also algorithms.
Specify and reason about systems.
(Co)algebra

Specify and reason about systems.

state-machines
e.g. DFA, LTS, PA
Specify and reason about systems.

Syntax
RE, CCS, ...

\[ b^* a (b^* a^*) \]

\[ a \cdot b \cdot 0 + a \cdot c \cdot 0 \]

\[ a \cdot (\frac{1}{2} \cdot 0 \oplus \frac{1}{2} \cdot 0) + \ldots \]

State-machines
E.g. DFA, LTS, PA
(Co)algebra

Specify and reason about systems.

Syntax
RE, CCS, ...

Axiomatization
KA, ...

RE, CCS, . . .

KA, . . .

e.g. DFA, LTS, PA

Can we do all of this uniformly in a single framework?

\[ b^*a(b^*a)^* \]

\[ a, b, 0 + a, c, 0 \]

\[ a, (1/2 \cdot 0 \oplus 1/2 \cdot 0) + \ldots \]

\[ 1 + a, a^* = a^* \]

\[ P + 0 = P \]

\[ P_1 \circ P_2, P = (P_1 + P_2), P \]
Specify and reason about systems.

Syntax
RE, CCS, ...

Axiomatization
KA, ...

state-machines
e.g. DFA, LTS, PA

Can we do all of this uniformly in a single framework?
What do this things have in common?

\[(S, t : S \rightarrow 2 \times S^A)\]
What do this things have in common?

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

$(S, t : S \rightarrow \mathcal{P}S^A)$
What do this things have in common?

\[(S, t : S \rightarrow 2 \times S^A)\]

\[(S, t : S \rightarrow \mathcal{P}S^A)\]

\[(S, t : S \rightarrow \mathcal{P}\mathcal{D}_\omega(S)^A)\]
What do this things have in common?

(S, t : S → 2 × S^A)

(S, t : S → P S^A)

(S, t : S → P D_ω(S)^A)

(S, t : S → D_ω(S) + (A × S) + 1)
What do this things have in common?

\[(S, t : S \to 2 \times S^A)\]

\[(S, t : S \to \mathcal{P} S^A)\]

\[(S, t : S \to \mathcal{P} \mathcal{D}_\omega(S)^A)\]

\[(S, t : S \to \mathcal{D}_\omega(S) + (A \times S) + 1)\]

\[(S, t : S \to \mathcal{P}(\mathcal{D}_\omega(\mathcal{P} S)^A))\]
What do these things have in common?

1. \((S, t : S \to 2 \times S^A)\)
2. \((S, t : S \to \mathcal{P} S^A)\)
3. \((S, t : S \to \mathcal{P} \mathcal{D}_\omega(S)^A)\)
4. \((S, t : S \to \mathcal{D}_\omega(S) + (A \times S) + 1)\)
5. \((S, t : S \to \mathcal{P}(\mathcal{D}_\omega(\mathcal{P} S)^A))\)
6. \((S, t : S \to TS)\)
What do these things have in common?

\[(S, t : S \rightarrow 2 \times S^A)\]

\[(S, t : S \rightarrow \mathcal{P} S^A)\]

\[(S, t : S \rightarrow \mathcal{P} \mathcal{D}_\omega(S)^A)\]

\[(S, t : S \rightarrow \mathcal{D}_\omega(S) + (A \times S) + 1)\]

\[(S, t : S \rightarrow \mathcal{P}(\mathcal{D}_\omega(\mathcal{P} S)^A))\]

\[(S, t : S \rightarrow TS) \quad T\text{-coalgebras}\]
The power of $T$

$$(S, t : S \rightarrow TS)$$
The power of $T$

$$(S, t : S \rightarrow TS)$$

The functor $T$ determines:

1. notion of observational equivalence (coalg. bisimulation)
   E.g. $T = 2 \times (\_)^A$: language equivalence
The power of \( T \)

\((S, t : S \to TS)\)

The functor \( T \) determines:

1. notion of observational equivalence (coalg. bisimulation)
   E.g. \( T = 2 \times (-)^A \): language equivalence

2. behaviour (final coalgebra)
   E.g. \( T = 2 \times (-)^A \): languages over \( A - 2^A^* \)
The power of $T$

$$(S, t : S \rightarrow TS)$$

The functor $T$ determines:

1. notion of observational equivalence (coalg. bisimulation)
   E.g. $T = 2 \times (-)^A$: language equivalence

2. behaviour (final coalgebra)
   E.g. $T = 2 \times (-)^A$: languages over $A – 2^A^*$

3. set of expressions describing finite systems

4. axioms to prove bisimulation equivalence of expressions

1 + 2 are classic coalgebra; 3 + 4 are recent work.
How about algorithms?

- Coalgebra has found its place in the semantic side of the world: operational/denotational semantics, logics, . . .
- Are there also opportunities for contributions in algorithms?
How about algorithms?

- Coalgebra has found its place in the semantic side of the world: operational/denotational semantics, logics, . . .
- Are there also opportunities for contributions in algorithms?

YES WE CAN!
Brzozowski’s algorithm (co)algebraically
Motivation

- duality between reachability and observability (Arbib and Manes 1975): beautiful, not very well-known.
- combined use of algebra and coalgebra.
- our understanding of automata is still very limited; cf. recent research: universal automata, àtomata, weighted automata (Sakarovitch, Brzozowski, . . . )
Credits

Bonchi, Bonsangue, Rutten
Credits

Bonchi, Bonsangue, Rutten

It all started with...

Prakash Panangaden
Credits

Bonchi, Bonsangue, Rutten

It all started with...

Nick Bezhanishvili & Clemens Kupke

Prakash Panangaden
Credits

Bonchi, Bonsangue, Rutten

It all started with...

Nick Bezhanishvili & Clemens Kupke

Prakash Panangaden

Helle Hansen & Dexter Kozen
Brzozowski algorithm (by example)

- initial state: $x$
- final states: $y$ and $z$
- $L(x) = \{a, b\}^* a$
Brzozowski algorithm (by example)

- initial state: $x$
- final states: $y$ and $z$
- $L(x) = \{a, b\}^* a$
- $X$ is reachable but not minimal: $L(y) = \varepsilon + \{a, b\}^* a = L(z)$
Reversing the automaton: \( rev(X) \)
Reversing the automaton: \( \text{rev}(X) \)

\[ X = \]

\[ \text{rev}(X) = \]
Reversing the automaton: $\text{rev}(X)$

$X =$

$\text{rev}(X) =$

- transitions are reversed
- initial states $\Leftrightarrow$ final states
Reversing the automaton: $\text{rev}(X)$

- transitions are reversed
- initial states $\Leftrightarrow$ final states
- $\text{rev}(X)$ is non-deterministic
Making it deterministic again: \( \text{det}(\text{rev}(X)) \)
Making it deterministic again: $\text{det}(\text{rev}(X))$
Making it deterministic again: \( \text{det}(\text{rev}(X)) \)

- new state space: \( 2^X = \{ V \mid V \subseteq \{ x, y, z \} \} \)
Making it deterministic again: $\text{det}(\text{rev}(X))$

- new state space: $2^X = \{ V \mid V \subseteq \{x, y, z\} \}$
- initial state: $\{y, z\}$  final states: all $V$ with $x \in V$
- $V \xrightarrow{a} W$  $W = \{ w \mid v \xrightarrow{a} w, v \in V \}$
The automaton $\text{det}(\text{rev}(X)) \ldots$
The automaton $\text{det}(\text{rev}(X))$ . . .

- . . . accepts the reverse of the language accepted by $X$:

$$L(\text{det}(\text{rev}(X))) = a\{a,b\}^* = \text{reverse}(L(X))$$
The automaton $\text{det}(\text{rev}(X))$ . . .

- . . . accepts the reverse of the language accepted by $X$:

$$L(\text{det}(\text{rev}(X))) = a\{a,b\}^* = \text{reverse}(L(X))$$

- . . . and is observable!
Today’s Theorem

If: a deterministic automaton $X$ is reachable and accepts $L(X)$
Today’s Theorem

If: a deterministic automaton $X$ is reachable and accepts $L(X)$

then: $det(rev(X))$ is minimal and

$$L(det(rev(X))) = reverse(L(X))$$
Taking the reachable part of \( \text{det}(\text{rev}(X)) \)
Taking the reachable part of $\text{det}(\text{rev}(X))$

- $\text{reach}(\text{det}(\text{rev}(X)))$
Taking the reachable part of $\text{det}(\text{rev}(X))$

- $\text{reach}(\text{det}(\text{rev}(X)))$ is reachable (by construction)
Repeating everything, now for \( \text{reach}(\text{det}(\text{rev}(X))) \)

\[
\begin{array}{c}
\text{a, b} \\
x, y, z \\
y, z \\
\emptyset \\
a, b
\end{array}
\]
Repeating everything, now for $\text{reach} (\text{det}(\text{rev}(X)))$
Repeating everything, now for $\text{reach}(\text{det}(\text{rev}(X)))$

• ... gives us $\text{reach}(\text{det}(\text{rev}(\text{reach}(\text{det}(\text{rev}(X)))))))$
Repeating everything, now for \( \text{reach}(\text{det}(\text{rev}(X))) \)

... gives us \( \text{reach}(\text{det}(\text{rev}(\text{reach}(\text{det}(\text{rev}(X))))))) \)

... which is (reachable and) minimal and accepts \( \{a, b\}^* a \).
All in all: Brzozowski’s algorithm

\[
\begin{align*}
X \text{ is reachable and accepts } & \{a, b\}^* \\
\text{reach}\left(\text{det}\left(\text{rev}\left(\text{reach}\left(\text{det}\left(\text{rev}\left(X\right)\right)\right)\right)\right)\right) \text{ also accepts } & \{a, b\}^* \\
\text{... and is minimal!!}
\end{align*}
\]
All in all: Brzozowski’s algorithm

• $X$ is reachable and accepts $\{a, b\}^*$
• $\text{reach}(\det(\text{rev}(\text{reach}(\det(\text{rev}(X))))))$ also accepts $\{a, b\}^*$
• ... and is minimal!!
All in all: Brzozowski’s algorithm

- $X$ is reachable and accepts $\{a, b\}^* a$
All in all: Brzozowski’s algorithm

- $X$ is reachable and accepts $\{a, b\}^* a$
- $\text{reach}(\text{det}(\text{rev}(\text{reach}(\text{det}(\text{rev}(X)))))))$ also accepts $\{a, b\}^* a$
All in all: Brzozowski’s algorithm

- $X$ is reachable and accepts $\{a,b\}^* a$
- $\text{reach}(\text{det}(\text{rev}(\text{reach}(\text{det}(\text{rev}(X)))))$ also accepts $\{a,b\}^* a$
- . . . and is minimal!!
Goal of the day

- Correctness of Brzozowski’s algorithm (co)algebraically
- Generalizations to other types of automata
(Co)algebra

algebras: $F(X) \xrightarrow{f} X$

coalgebras: $X \xrightarrow{f} F(X)$
Examples of algebras

\[ \mathbb{N} \times \mathbb{N} \]

\[ \downarrow \downarrow \]

\[ \mathbb{N} \]

\[ \mathbb{N} = \mathbb{N} \]
Examples of algebras

\[ \mathbb{N} \times \mathbb{N} \quad \rightarrow \quad \mathbb{N} \]

\[ 1 + \mathbb{N} \quad \rightarrow \quad [0, S] \quad \rightarrow \quad \mathbb{N} \]

\[ [0, S] \quad \equiv \quad 1 + \mathbb{N} \quad \rightarrow \quad 0 \quad \rightarrow \quad S \quad \equiv \quad 1 \quad \rightarrow \quad \mathbb{N} \]

\[ \mathbb{N} \]

\[ \mathbb{N} \]

\[ S \]

\[ \mathbb{N} \]
Examples of coalgebras

\[
X \xrightarrow{t} P(A \times X)
\]

\[
x \xrightarrow{a} y \iff \langle a, y \rangle \in t(x)
\]
Examples of coalgebras

$x \xrightarrow{a} y \iff \langle a, y \rangle \in t(x)$
Examples of coalgebras

\[ \langle \text{head}, \text{tail} \rangle \quad \equiv \quad \text{head} \quad \downarrow \quad \text{tail} \quad \equiv \quad 2^\omega \]

\[ 2 \times 2^\omega \quad \cong \quad 2^\omega \]

\[ \text{head}((b_0, b_1, b_2, \ldots)) = b_0 \]

\[ \text{tail}((b_0, b_1, b_2, \ldots)) = (b_1, b_2, b_3 \ldots) \]
Homomorphisms

\[
\begin{array}{ccc}
F(X) & \xrightarrow{F(h)} & F(Y) \\
\downarrow f & & \downarrow g \\
X & \xrightarrow{h} & Y
\end{array}
\]
Homomorphisms

\[ F(X) \xrightarrow{F(h)} F(Y) \]

\[ \downarrow f \quad \downarrow g \]

\[ X \xrightarrow{h} Y \]

\[ F(X) \xrightarrow{F(h)} F(Y) \]

\[ \downarrow h \quad \quad \downarrow \]
Initiality, finality

\[ F(A) \xrightarrow{F(h)} F(X) \]
\[ \alpha \downarrow \quad \quad \quad \downarrow f \]
\[ A \xrightarrow{\exists ! h} X \]

\[ X \xrightarrow{\exists ! h} Z \]
\[ f \downarrow \quad \quad \beta \downarrow \]
\[ F(X) \xrightarrow{F(h)} F(Z) \]
Initiality, finality

$F(A) \xrightarrow{F(h)} F(X)$

$A \xrightarrow{\exists! h} X$

$X \xrightarrow{\exists! h} Z$

$F(X) \xrightarrow{F(h)} F(Z)$

- initial algebras ↔ induction
Initiality, finality

\[ F(A) \xrightarrow{F(h)} F(X) \]

\[ A \xrightarrow{\exists! h} X \]

\[ X \xrightarrow{\exists! h} Z \]

\[ F(X) \xrightarrow{F(h)} F(Z) \]

- initial algebras \(\leftrightarrow\) induction
- final coalgebras \(\leftrightarrow\) coinduction
Automata, (co)algebraically

- Automata are complicated structures:
  part of them is algebra - part of them is coalgebra
Automata, (co)algebraically

- Automata are complicated structures:
  - part of them is algebra - part of them is coalgebra
- ( . . . in two different ways . . . )
A deterministic automaton

\[ X \quad t \quad X^A \]

where

\[ 1 = \{ 0 \} \]
\[ 2 = \{ 0, 1 \} \]
\[ X^A = \{ g \mid g : A \rightarrow X \} \]

\[ x \rightarrow y \leftrightarrow t(x)(a) = y \]

\[ i(0) \in X \]

is the initial state

\[ f(\_\_) = 1 \]

\[ x \text{ is final (or accepting)} \leftrightarrow f(x) = 1 \]
A deterministic automaton

\[
\begin{array}{c}
1 \\
\downarrow \quad i \\
\downarrow \quad f \\
X \\
\downarrow \quad t \\
X^A
\end{array}
\]

where

\[1 = \{0\} \quad 2 = \{0, 1\} \quad X^A = \{g \mid g : A \to X\}\]

\[x \xrightarrow{a} y \iff t(x)(a) = y\]

\[i(0) \in X \text{ is the initial state}\]

\[\bigcirc x \text{ is final (or accepting)} \iff f(x) = 1\]
Automata: algebra or coalgebra?

- initial state: algebraic – final states: coalgebraic

\[
\begin{array}{c}
1 \\
\downarrow i \\
X \\
\uparrow f \\
2
\end{array}
\]
Automata: algebra or coalgebra?

- initial state: algebraic – final states: coalgebraic

- transition function: both algebraic and coalgebraic

\[ X \xrightarrow{t} X^A \]

\[ X \xrightarrow{(A \xrightarrow{t} X)} \]

\[ X \times A \xrightarrow{t} X \]
Automata: algebra and coalgebra!

To take home: this picture!! … which we’ll explain next …
Automata: algebra and coalgebra!

To take home: this picture!! . . .
Automata: algebra and coalgebra!

To take home: this picture!! . . . which we’ll explain next . . .
The “automaton” of languages

$\epsilon?(L) = 1 \iff \epsilon \in L$

$2^A^* = \{g \mid g : A^* \to 2\} \cong \{L \mid L \subseteq A^*\}$

$\beta(L)(a) = L_a = \{w \in A^* \mid a \cdot w \in L\}$
The “automaton” of languages

\[\epsilon?(L) = 1 \iff \epsilon \in L\]

\[2^{A^*} = \{g \mid g : A^* \to 2\} \cong \{L \mid L \subseteq A^*\}\]

\[\beta(L)(a) = L_a = \{w \in A^* \mid a \cdot w \in L\}\]

- We say “automaton”: it does not have an initial state.
The automaton of languages

- transitions: \( L \xrightarrow{a} L_a \) where \( L_a = \{ w \in A^* \mid a \cdot w \in L \} \)
- for instance:
The automaton of languages

- transitions: $L \xrightarrow{a} L_a$ where $L_a = \{ w \in A^* \mid a \cdot w \in L \}$
- for instance:
The automaton of languages

- transitions: $L \xrightarrow{a} L_a$ where $L_a = \{ w \in A^* \mid a \cdot w \in L \}$
- for instance:

- note: every state $L$ accepts . . .
The automaton of languages

- transitions: $L \xrightarrow{a} L_a$ where $L_a = \{ w \in A^* \mid a \cdot w \in L \}$
- for instance:

![Automaton Diagram]

- note: every state $L$ accepts . . . . . . the language $L$ !!
The automaton of languages is . . . final

\[ o(x) = \{ w \in A^* \mid f(x_w) = 1 \} \]

= the language accepted by \( x \)
The automaton of languages is . . . final

\[ o(x) = \{ w \in A^* | f(x_w) = 1 \} \]

= the language accepted by \( x \)

where: \( x_w \) is the state reached after inputting the word \( w \),
and: \( o^A(g) = o \circ g \), all \( g \in X^A \).
Back to today’s picture

On the right: final coalgebra
On the left: initial algebra...
Back to today’s picture

On the right: final coalgebra

On the left: initial algebra . . .
Back to today’s picture

On the right: final coalgebra
On the left: initial algebra . . .
The “automaton” of words

$\epsilon$ is initial state

$\alpha(w)(a) = w \cdot a$

that is, transitions: $w \xrightarrow{a} w \cdot a$
The automaton of words is . . . initial

\[
\begin{align*}
1 & \downarrow \epsilon
\quad A^* & \downarrow \exists! r
\quad X & \downarrow t
\quad (A^*)^A & \downarrow r^A
\quad X^A &
\end{align*}
\]

\( i \in X \) = initial state
(to be precise: \( i(0) \))

\( r(w) = i_w \)
= the state reached from \( i \)
after inputting \( w \)

- Proof: easy exercise.
- Proof: formally, because \( A^* \) is an initial \( 1 + A \times (-)\)-algebra!
Reachability and observability are dual:

*Arbib* and *Manes*, 1975.

(Here observable = minimal)
Reachability and observability

\[ \begin{align*}
(A^*)^A & \rightarrow X^A & X & \rightarrow \mathcal{2}^{A^*} \\
A^* & \rightarrow X & \mathcal{2}^{A^*} & \rightarrow (A^*)^A
\end{align*} \]

- We call \( X \) reachable if \( r \) is surjective.
- We call \( X \) observable (= minimal) if \( o \) is injective.
Reachability and observability

We call $X$ reachable if $r$ is surjective.
We call $X$ observable (= minimal) if $o$ is injective.

$r(w) = \text{state reached on input } w$

$o(x) = \text{language accepted by } x$
Reachability and observability

- We call $X$ reachable if $r$ is surjective.

- We call $X$ observable (= minimal) if $o$ is injective.

$r(w) = \text{state reached on input } w$

$o(x) = \text{language accepted by } x$
Reachability and observability

We call $X$ reachable if $r$ is surjective.

We call $X$ observable (= minimal) if $o$ is injective.
Reversing the automaton

- Reachability $\leftrightarrow$ observability
- Being precise about homomorphisms is crucial.
- Forms the basis for proof Brzozowski’s algorithm.
Powerset construction

\[ 2^{(-)} : \quad g \quad \mapsto \quad 2^g \]

where

\[ 2^V = \{ S \mid S \subseteq V \} \]

and, for all \( S \subseteq W \),

\[ 2^g(S) = g^{-1}(S) = \{ v \in V \mid g(v) \in S \} \]

• This construction is contravariant!!

• Note: if \( g \) is surjective, then \( 2^g \) is injective.
Powerset construction

\[ \begin{array}{ccc}
V & \xrightarrow{g} & 2^V \\
\downarrow & & \uparrow 2^g \\
W & \xleftarrow{2^g(S)} & 2^W \\
\end{array} \]

where \( 2^V = \{ S \mid S \subseteq V \} \) and, for all \( S \subseteq W \),

\( 2^g(S) = g^{-1}(S) \)  

(\( = \{ v \in V \mid g(v) \in S \} \) )
Powerset construction

\[ 2^V : g \rightarrow 2^W \]

where \( 2^V = \{ S \mid S \subseteq V \} \) and, for all \( S \subseteq W \),

\[ 2^g(S) = g^{-1}(S) \quad (= \{ v \in V \mid g(v) \in S \}) \]

- This construction is contravariant !!
Powerset construction

\[ 2^{(-)} : \quad \begin{array}{c} V \\
\downarrow \quad g \\
W \\
\end{array} \quad \mapsto \quad \begin{array}{c} 2^V \\
\uparrow \quad 2^g \\
2^W \\
\end{array} \]

where \( 2^V = \{ S \mid S \subseteq V \} \) and, for all \( S \subseteq W \),

\[ 2^g(S) = g^{-1}(S) \quad (\text{=} \{ v \in V \mid g(v) \in S \}) \]

- This construction is **contravariant** !!
- Note: if \( g \) is surjective, then \( 2^g \) is injective.
Reversing transitions
Reversing transitions

\[
\begin{array}{c|c}
X & X \times A \\
\downarrow t & \downarrow \ \\
X^A & X \\
\end{array}
\]
Reversing transitions
Reversing transitions

\[
\begin{array}{c}
X \downarrow \quad \| \quad X \times A \\
X^A \downarrow \quad 2^{(-)} \quad \| \quad (2^X)^A \\
t \downarrow \quad 2^X \downarrow \\
X \quad 2^X \\
\end{array}
\]
Reversing transitions

\[
\begin{align*}
X & \quad \| \quad X \times A \\
\downarrow t & \quad \| \quad \downarrow \quad X \\
X^A & \quad \| \quad X
\end{align*}
\]

\[
\begin{align*}
2^{X \times A} & \quad \| \quad (2^X)^A \\
\downarrow 2^{X \times A} & \quad \| \quad \downarrow \quad (2^X)^A \\
2^X & \quad \| \quad 2^X \\
\end{align*}
\]

\[
\begin{align*}
2^X & \quad \| \quad 2^X \\
\downarrow 2^X & \quad \| \quad \downarrow \quad (2^X)^A \\
2^t & \quad \| \quad 2^t
\end{align*}
\]

\[
\begin{align*}
\frac{39}{51}
\end{align*}
\]
Initial ↔ final
Initial ↔ final

1\rightarrow i \rightarrow X \rightarrow 2^{\leftarrow (\cdot)} \leftarrow 2i \rightarrow 2

2X \rightarrow f \rightarrow 2^{\leftarrow (\cdot)} \leftarrow f \rightarrow 2
Initial $\leftrightarrow$ final

$1 \xrightarrow{i} X \xleftarrow{f} 2$

$2 \xrightarrow{\text{initial}} X \xleftarrow{\text{final}} 2^i \xrightarrow{2^{(-i)}} 2^x$
Initial $\leftrightarrow$ final

1 $\xrightarrow{i} X$

2 $\xleftarrow{f} 2^X$

2 $\xrightarrow{2^{(-)}} 2^X$

2 $\xleftarrow{2^i} 2$
Reversing the entire automaton

\[
\begin{array}{c}
1 \\
\uparrow \, i \\
\downarrow \\
X \\
\downarrow \, t \\
\downarrow \\
X^A
\end{array}
\begin{array}{c}
2 \\
\downarrow \, f \\
\downarrow \\
X \\
\downarrow \, t \\
\downarrow \\
X^A
\end{array}
\]

\[X^A \cdot \text{Initial and final are exchanged...}
\]
\[\rightarrow \text{transitions are reversed...}
\]
\[\rightarrow \text{and the result is again deterministic!}
\]
Reversing the entire automaton

\[
\begin{array}{c}
1 \rightarrow X^A \\
\downarrow t \\
X \\
\downarrow f \\
2 \\
\end{array}
\quad
\begin{array}{c}
2 \rightarrow (2^X)^A \\
\downarrow 2^t \\
2^X \\
\downarrow 2^{(-)} \\
1 \\
\end{array}
\]

• Initial and final are exchanged.
• Transitions are reversed.
• And the result is again deterministic!
Reversing the entire automaton

1. Initial and final are exchanged . . .
Reversing the entire automaton

• Initial and final are exchanged . . .
• transitions are reversed . . .
Reversing the entire automaton

• Initial and final are exchanged . . .
• transitions are reversed . . .
• and the result is again deterministic!
Our previous example

\[ X = \]

\[
\begin{array}{c}
\circ \circ \\
x & b & b \\
y & a & z \\
\end{array}
\]

Note that \( X \) has been reversed and determinized:

\[ 2 \times X = \text{det} \left( \text{rev} \left( X \right) \right) \]
Our previous example

\[ X = \]

\[ 2^X = \]
Our previous example

$X = \xymatrix@C=1em{ x \ar[r]^b \ar[d]^a & z \ar[l]^b \\
 y \ar[r]^b \\
 & a }$

$2^X = \xymatrix@C=1em{ x, y \ar[r]^b \ar[d]^a & x, y, z \ar[l]^b \\
 x, z \ar[r]^a \\
 & \emptyset \ar[u]^a \\
 y \ar[r]^b \\
 & \emptyset \ar[u]^a \\
 & x \ar[u]^a }$

- Note that $X$ has been reversed and determinized:

$$2^X = \text{det}(\text{rev}(X))$$
Proving today’s Theorem

If: a deterministic automaton $X$ is reachable and accepts $L(X)$
If: a deterministic automaton $X$ is reachable and accepts $L(X)$
then: $2^X (\equiv det(rev(X)))$ is minimal/observable and

$$L(2^X) = reverse(L(X))$$
Proof: by reversing $A^* \xrightarrow{r} X$
Proof: by reversing $A^* \xrightarrow{r} X$

$A^* \xrightarrow{r} X$

$A^* \xrightarrow{r} X$

$(A^*)^A \xrightarrow{r} X^A$

$2X \xrightarrow{2^t} (2^X)^A \xrightarrow{2^r} (2A^*)^A$

$2^\epsilon \xrightarrow{2^t} (2A^*)^A \xrightarrow{2^r} (2^X)^A \xrightarrow{2^i} 2X$

$2^\epsilon \xrightarrow{2^t} (2A^*)^A \xrightarrow{2^r} (2^X)^A \xrightarrow{2^i} 2X$

$1 \xrightarrow{i} X \xrightarrow{t} (A^*)^A \xrightarrow{r} X^A$

$2^\epsilon \xrightarrow{2^t} (2A^*)^A \xrightarrow{2^r} (2^X)^A \xrightarrow{2^i} 2X$

$2^\epsilon \xrightarrow{2^t} (2A^*)^A \xrightarrow{2^r} (2^X)^A \xrightarrow{2^i} 2X$

$1 \xrightarrow{i} X \xrightarrow{t} (A^*)^A \xrightarrow{r} X^A$
Proof: by reversing $A^* \xrightarrow{r} X$

- $X$ becomes $2^X$
Proof: by reversing $A^* \xrightarrow{r} X$

- $X$ becomes $2^X$
- initial automaton $A^*$ becomes (almost) final automaton $2^{A^*}$
Proof: by reversing $A^* \xrightarrow{r} X$

- $X$ becomes $2^X$
- Initial automaton $A^*$ becomes (almost) final automaton $2^{A^*}$
- $r$ is surjective $\Rightarrow$ $2^r$ is injective
Reachable becomes observable

\[
\begin{array}{c}
1 \\
\downarrow \epsilon \\
A^* \\
\downarrow \alpha \\
(A^*)^A
\end{array}
\quad \begin{array}{c}
i \quad r \\
\downarrow X \\
\downarrow t \\
X^A
\end{array}
\]
Reachable becomes observable

\[
\begin{array}{c}
\begin{array}{c}
1 \\
\epsilon \\
A^* \\
\alpha \\
(A^*)^A
\end{array}
\end{array}
\rightarrow
\begin{array}{c}
\begin{array}{c}
i \\
r \\
X \\
t \\
X^A
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
2 \\
2^i \\
2^\epsilon \\
\beta
\end{array}
\end{array}
\rightarrow
\begin{array}{c}
\begin{array}{c}
2 \\
2r \\
2^\alpha
\end{array}
\end{array}
\rightarrow
\begin{array}{c}
\begin{array}{c}
2^A \\
(2^A)^A
\end{array}
\end{array}
\rightarrow
\begin{array}{c}
\begin{array}{c}
2^{A^*} \\
(2^{A^*})^A
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
2^{(-)} \\
2^t
\end{array}
\end{array}
\rightarrow
\begin{array}{c}
\begin{array}{c}
2^X \\
2^A
\end{array}
\end{array}
\rightarrow
\begin{array}{c}
\begin{array}{c}
2^A^* \\
rev
\end{array}
\end{array}
\rightarrow
\begin{array}{c}
\begin{array}{c}
2^{A^*} \\
\epsilon?
\end{array}
\end{array}
\rightarrow
\begin{array}{c}
\begin{array}{c}
\epsilon
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
\text{If } r \text{ is surjective then } (2^r \text{ and hence}) \\
\text{rev} \circ 2^r \text{ is injective.}
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
\text{That is, } 2^X \text{ is observable (= minimal).}
\end{array}
\end{array}
\]
Reachable becomes observable

If $r$ is surjective then $(2^r$ and hence) $rev \circ 2^r$ is injective.
Reachable becomes observable

- If \( r \) is surjective then \( 2^r \) and hence \( \text{rev} \circ 2^r \) is injective.
- That is, \( 2^X \) is observable (= minimal).
Summarizing

\[ \begin{array}{c}
1 \\
\epsilon \\
A^* \\
\alpha \\
(A^*)^A
\end{array} \xrightarrow{i} \begin{array}{c}
2 \\
\downarrow r \\
X \\
\downarrow t \\
(X^A)
\end{array} \xrightarrow{f} \begin{array}{c}
\alpha \\
\downarrow \epsilon \\
\downarrow i \\
\downarrow f \\
\downarrow \epsilon
\end{array} \]

\text{If: } X \text{ is reachable, i.e., } \alpha \text{ is surjective then: } \text{rev} \circ r \text{ is injective, i.e., } X \text{ is observable = minimal.}

\text{And: } \text{rev} (\alpha (f)) = \text{rev} (o (i)), \text{ i.e., } L(2X) = \text{reverse}(L(X)) \]
If $X$ is reachable, i.e., $r$ is surjective then:

$\text{rev} \circ 2^r$ is injective, i.e., $2^X$ is observable = minimal.

And:

$L(2^X) = \text{reverse}(L(X))$
Summarizing

- If: $X$ is reachable, i.e., $r$ is surjective
Summarizing

If: \( X \) is reachable, i.e., \( r \) is surjective

then: \( \text{rev} \circ 2^r \) is injective, i.e., \( 2^X \) is observable = minimal.
If: $X$ is reachable, i.e., $r$ is surjective
then: $\text{rev} \circ 2^r$ is injective, i.e., $2^X$ is observable = minimal.

And: $\text{rev}(2^r(f)) = \text{rev}(o(i))$, i.e., $L(2^X) = \text{reverse}(L(X))$
Corollary: Brzozowski’s algorithm

- $X$ becomes $2^X$, accepting $\text{reverse}(L(X))$
Corollary: Brzozowski’s algorithm

- $X$ becomes $2^X$, accepting $\text{reverse}(L(X))$
- take reachable part: $Y = \text{reachable}(2^X)$
Corollary: Brzozowski’s algorithm

- $X$ becomes $2^X$, accepting $\text{reverse}(L(X))$
- take reachable part: $Y = \text{reachable}(2^X)$
- $Y$ becomes $2^Y$, which is minimal and accepts

$$\text{reverse}(\text{reverse}(L(X))) = L(X)$$
Generalizations

- A Brzozowski minimization algorithm for Moore automata.

\[ B^X = \{ \varphi \mid \varphi: X \to B \} \quad B^f(\varphi) = \varphi \circ f \]
Further generalizations

- Moore automata generalization: uniform algorithm for decorated traces and must testing (joint work with Bonchi, Caltais and Pous);
- Further generalizations to non-deterministic and weighted automata.
A uniform picture based on duality

\[ \begin{array}{ccc}
\text{Aut}_F & \perp & \text{Aut}(G) \\
\text{Coalg}(F) & \perp & \text{Alg}(G) \\
F & \perp & D \\
C & \perp & G
\end{array} \]
Conclusions

- Combination algebra-coalgebra is fruitful.
- Abstract analysis can bring new perspectives/results.
- (Co)algebra is not only semantics but also algorithms!

Thanks!
Conclusions

- Combination algebra-coalgebra is fruitful.
- Abstract analysis can bring new perspectives/results.
- (Co)algebra is not only semantics but also algorithms!

Thanks!