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.

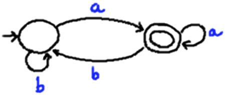
(Co)algebra

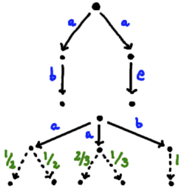
Specify and reason about systems.

(Co)algebra

Specify and reason about systems.

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





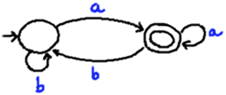
(Co)algebra

Specify and **reason** about **systems**.

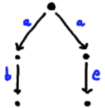
Syntax
RE, CCS, . . .

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

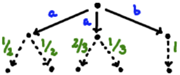
$b^*a(b^*a)^*$



$a.b.0 + a.c.0$

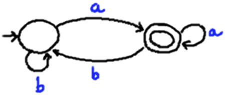
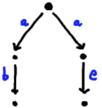
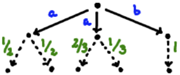


$a.(\frac{1}{2}\cdot 0 \oplus \frac{1}{2}\cdot 0) + \cdots$



(Co)algebra

Specify and **reason** about **systems**.

Specify	reason	systems
Syntax	Axiomatization	state-machines
RE, CCS, …	KA,…	e.g. DFA, LTS, PA
$b^*a(b^*a)^*$	$1 + aa^* = a^*$ ⋮	
$a.b.0 + a.c.0$	$P + 0 = P$ ⋮	
$a.(\frac{1}{2}\cdot 0 \oplus \frac{1}{2}\cdot 0) + \cdots$	$p.P \oplus p'.P = (p+p').P$ ⋮	

(Co)algebra

Specify and **reason** about **systems**.

Syntax
RE, CCS, . . .

$b^*a(b^*a)^*$

$a.b.0 + a.c.0$

$a.(\frac{1}{2}\cdot 0 \oplus \frac{1}{2}\cdot 0) + \cdots$

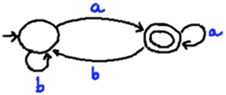
Axiomatization
KA,. . .

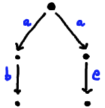
$1 + a a^* = a^*$
$\vdots$

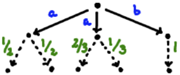
$P + 0 = P$
$\vdots$

$p.P \oplus p'.P = (p+p').P$
$\vdots$

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

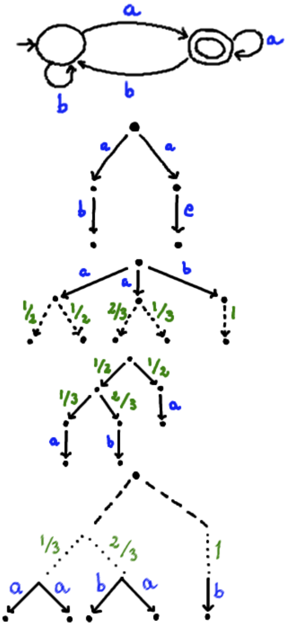






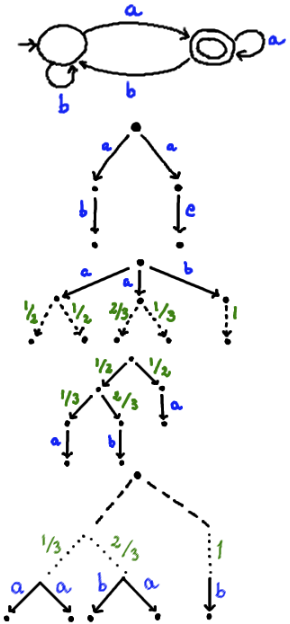
Can we do all of this uniformly in a single framework?

What do this things have in common?



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

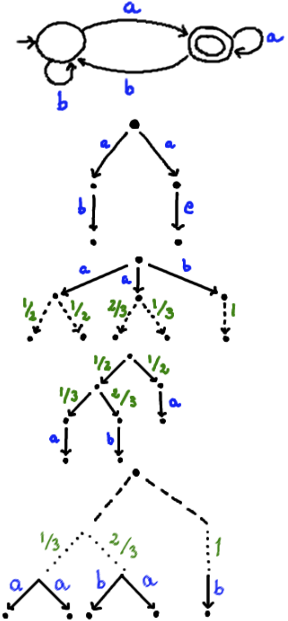
What do this things have in common?



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

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

What do this things have in common?

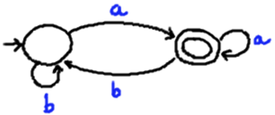


$(S, t : S \to 2 \times S^A)$

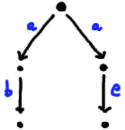
$(S, t : S \to \mathcal{P}S^A)$

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

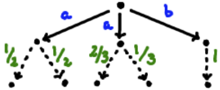
What do this things have in common?



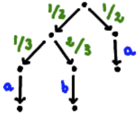
$$(S, t : S \to 2 \times S^A)$$



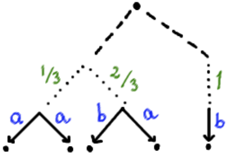
$$(S, t : S \to \mathcal{P}S^A)$$



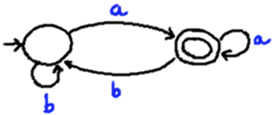
$$(S, t : S \to \mathcal{P}\mathcal{D}_\omega(S)^A)$$



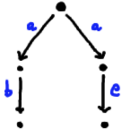
$$(S, t : S \to \mathcal{D}_\omega(S) + (A \times S) + 1)$$



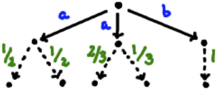
What do this things have in common?



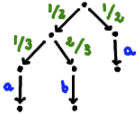
$$(S, t : S \to 2 \times S^A)$$



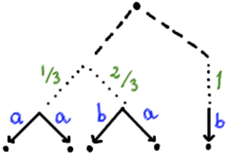
$$(S, t : S \to \mathcal{P}S^A)$$



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

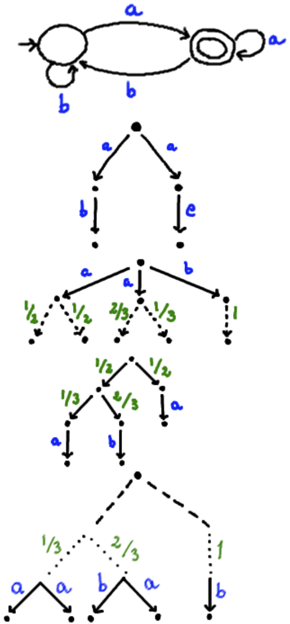


$$(S, t : S \to \mathcal{D}_\omega(S) + (A \times S) + 1)$$



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

What do this things have in common?



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

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

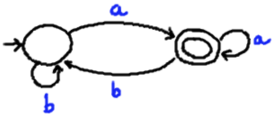
$$(S, t : S \to \mathcal{P}\mathcal{D}_\omega(S)^A)$$

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

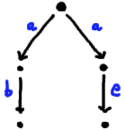
$$(S, t : S \to \mathcal{P}(\mathcal{D}_\omega(\mathcal{P}S)^A))$$

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

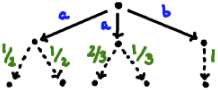
What do this things have in common?



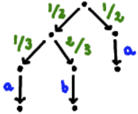
$(S, t : S \to 2 \times S^A)$



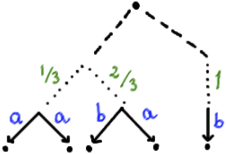
$(S, t : S \to \mathcal{P}S^A)$



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



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



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

$(S, t : S \to TS)$ T-coalgebras

The power of T

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

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

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

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

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^*}
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?

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

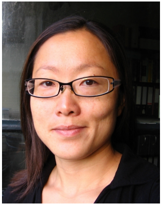
Credits

Bonchi, Bonsangue, Rutten

It all started with...

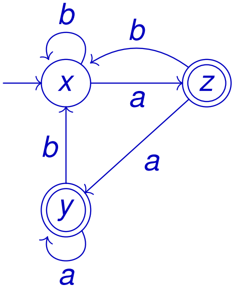
Prakash Panangaden

Nick Bezhanishvili & Clemens Kupke



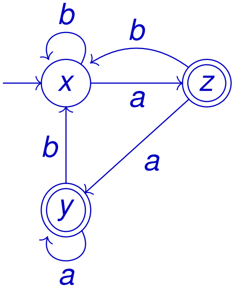
Helle Hansen & Dexter Kozen

Brzozowski algorithm (by example)



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

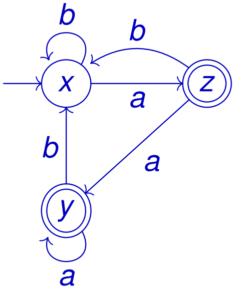
Brzozowski algorithm (by example)



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

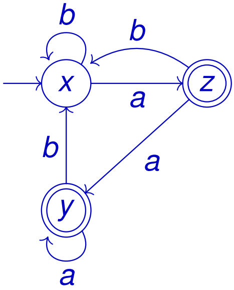
Reversing the automaton: *rev*(X)

$X =$

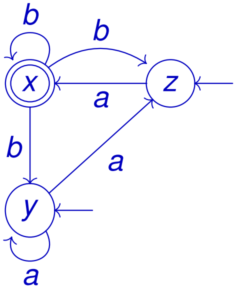


Reversing the automaton: *rev*(X)

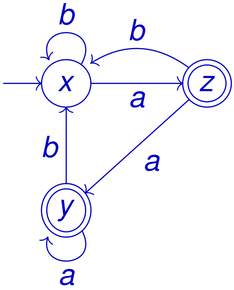
$X =$

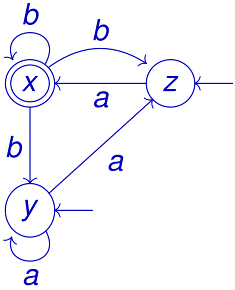


$rev(X) =$



Reversing the automaton: $rev(X)$

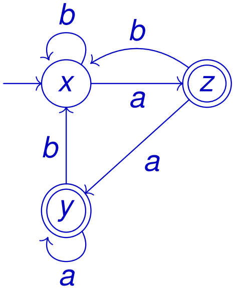
$X =$ 

$rev(X) =$ 

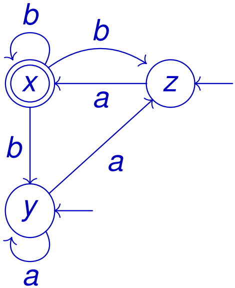
- transitions are reversed
- initial states $\Leftrightarrow$ final states

Reversing the automaton: $rev(X)$

$X =$

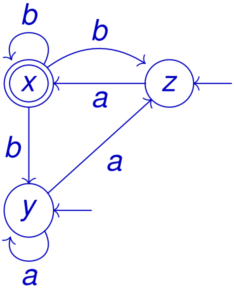


$rev(X) =$

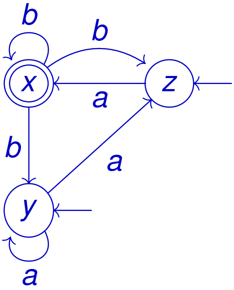


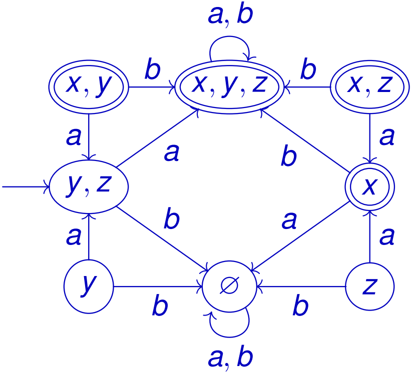
- transitions are reversed
- initial states $\Leftrightarrow$ final states
- $rev(X)$ is non-deterministic

Making it deterministic again: $det(rev(X))$

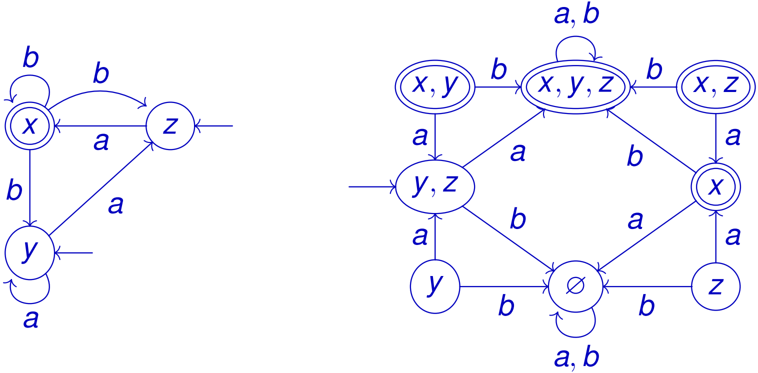


Making it deterministic again: $det(rev(X))$



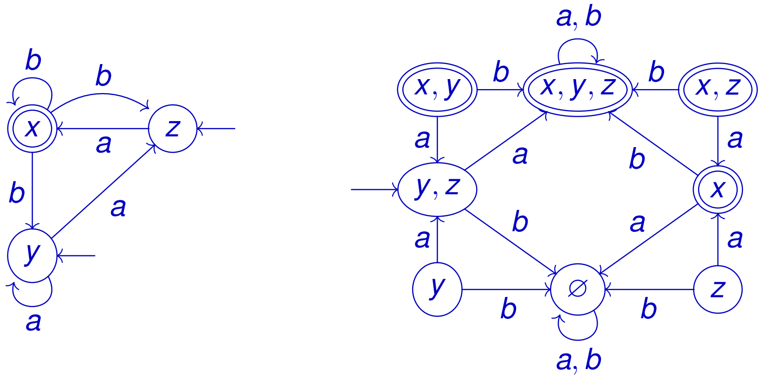


Making it deterministic again: $det(rev(X))$



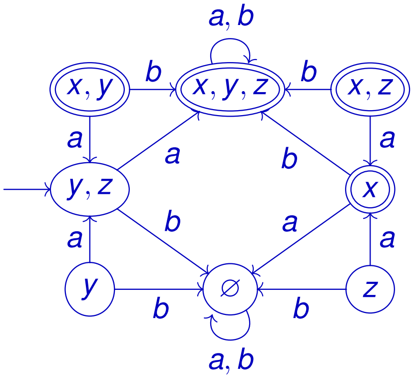
- new state space: $2^X = \{ V \mid V \subseteq \{x, y, z\} \}$

Making it deterministic again: $det(rev(X))$

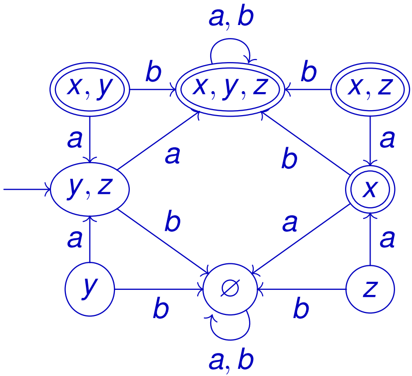


- new state space: $2^X = \{ V \mid V \subseteq \{x, y, z\} \}$
- initial state: $\{y, z\}$ final states: all V with $x \in V$
- $V \xrightarrow{a} W \qquad W = \{ w \mid v \xrightarrow{a} w, v \in V \}$

The automaton $det(rev(X))$ . . .



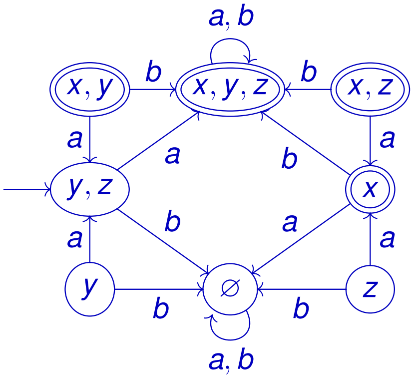
The automaton $det(rev(X))$ . . .



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

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

The automaton $det(rev(X))$ . . .



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

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

- . . . and is observable!

Today's Theorem

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

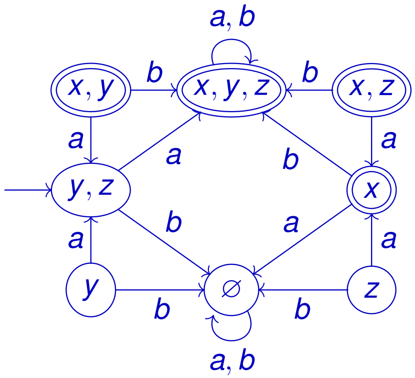
Today's Theorem

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

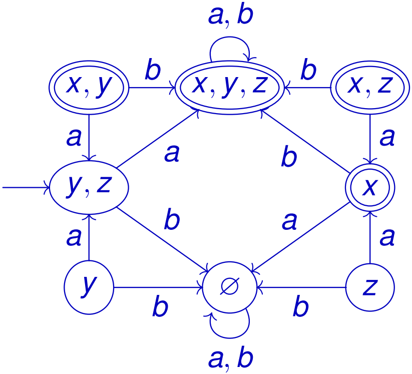
then: $det(rev(X))$ is minimal and

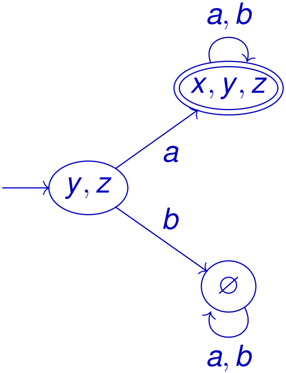
$$L(\,det(rev(X))\,) \;=\; reverse(\,L(X)\,)$$

Taking the reachable part of $det(rev(X))$



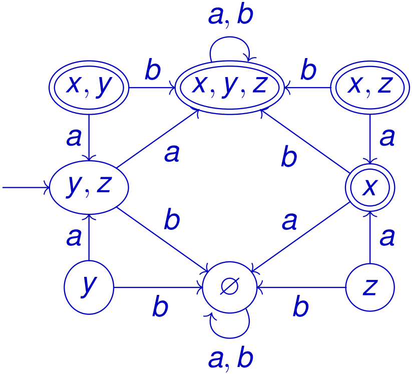
Taking the reachable part of *det*(*rev*(*X*))

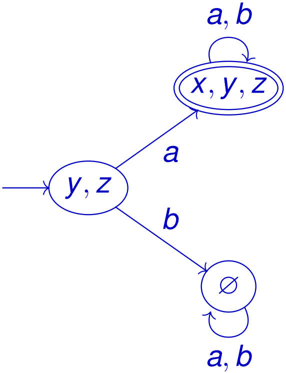




- *reach*(*det*(*rev*(*X*)))

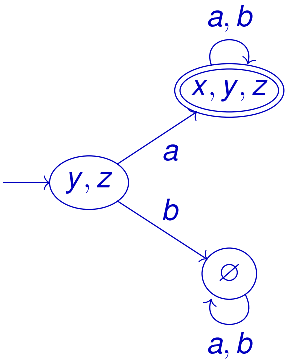
Taking the reachable part of $det(rev(X))$



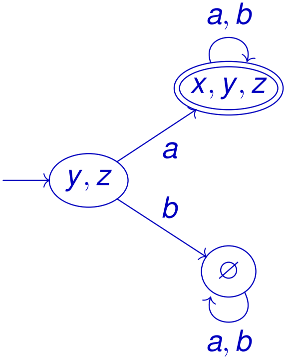


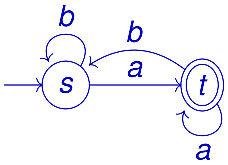
- $reach(det(rev(X)))$ is reachable (by construction)

Repeating everything, now for *reach*(*det*(*rev*(*X*)))

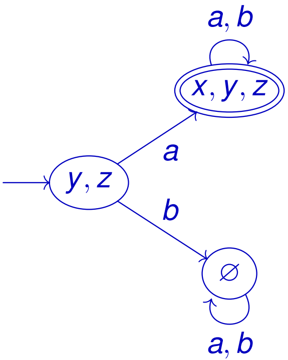


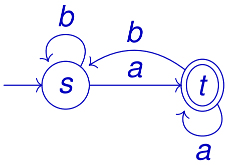
Repeating everything, now for *reach*(*det*(*rev*(*X*)))





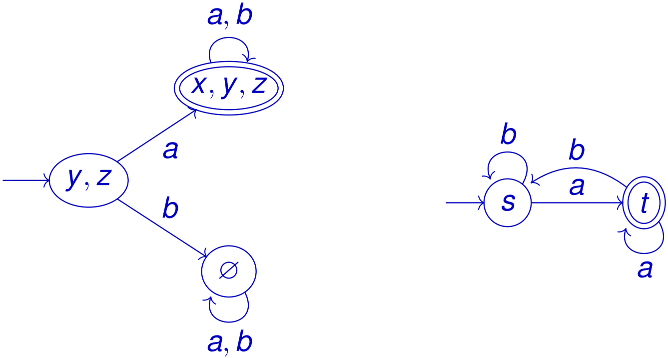
Repeating everything, now for *reach*(*det*(*rev*(*X*)))





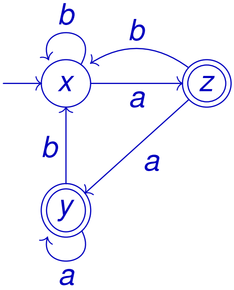
- . . . gives us $reach(det(rev(reach(det(rev(X))))))$

Repeating everything, now for *reach*(*det*(*rev*(X)))

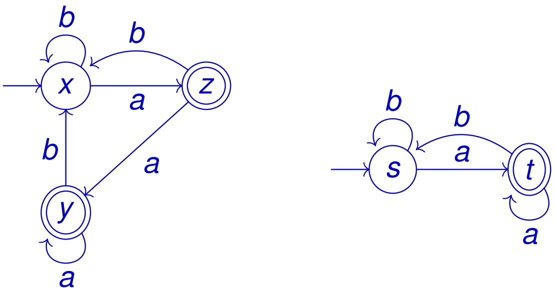


- ... gives us *reach*(*det*(*rev*(*reach*(*det*(*rev*(X))))))
- which is (reachable and) minimal and accepts $\{a, b\}^* a$.

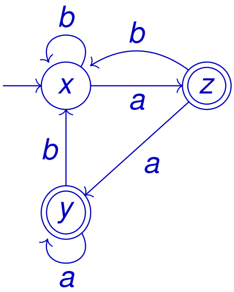
All in all: Brzozowski's algorithm

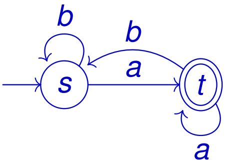


All in all: Brzozowski's algorithm



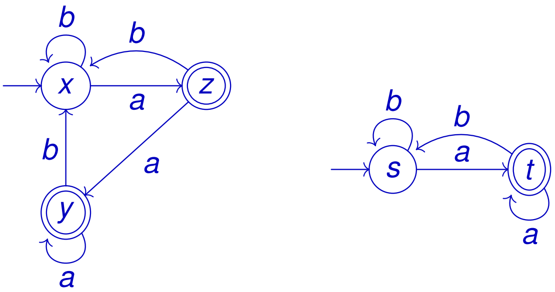
All in all: Brzozowski's algorithm





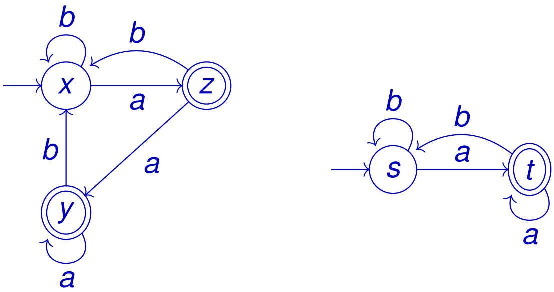
- X is reachable and accepts $\{a, b\}^* \, a$

All in all: Brzozowski's algorithm



- X is reachable and accepts $\{a, b\}^* a$
- $reach(det(rev(reach(det(rev(X))))))$ also accepts $\{a, b\}^* a$

All in all: Brzozowski's algorithm



- X is reachable and accepts $\{a, b\}^* \, a$
- $reach(det(rev(reach(det(rev(X))))))$ 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

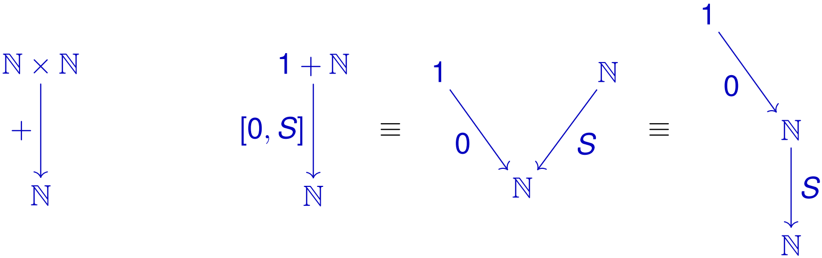
algebras: $$\begin{array}{c} F(X) \\ \downarrow f \\ X \end{array}$$

coalgebras: $$\begin{array}{c} X \\ \downarrow f \\ F(X) \end{array}$$

Examples of algebras

$$\begin{array}{c} \mathbb{N} \times \mathbb{N} \\ \Big\downarrow {\scriptstyle +} \\ \mathbb{N} \end{array}$$

Examples of algebras



Examples of coalgebras

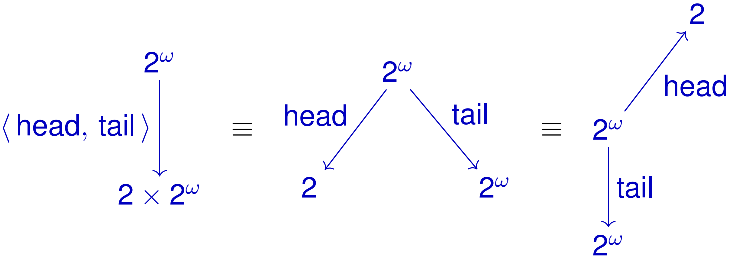
$$\begin{array}{c} X \\ \downarrow{\scriptstyle t} \\ \mathcal{P}(A \times X) \end{array} \qquad\qquad x \xrightarrow{\;a\;} y \quad \leftrightarrow \quad \langle a, y \rangle \in t(x)$$

Examples of coalgebras

$$\begin{array}{c} X \\ \big\downarrow t \\ \mathcal{P}(A \times X) \end{array} \qquad x \xrightarrow{a} y \quad \leftrightarrow \quad \langle a, y \rangle \in t(x)$$

$$\begin{array}{c} X \\ \big\downarrow \langle Left, label, Right \rangle \\ X \times A \times X \end{array}$$

Examples of coalgebras



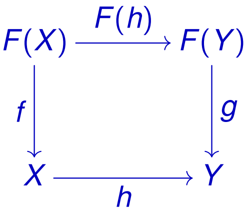
$$\text{head}((b_0, b_1, b_2, \ldots)) = b_0$$

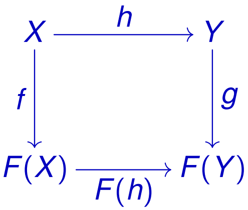
$$\text{tail}((b_0, b_1, b_2, \ldots)) = (b_1, b_2, b_3 \ldots)$$

Homomorphisms

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

Homomorphisms

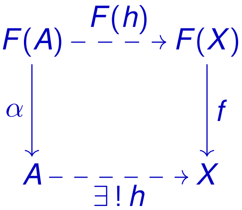


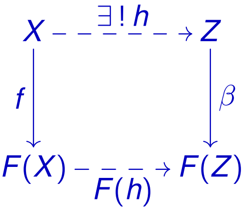


Initiality, finality

$$
\begin{array}{ccc}
F(A) & \xrightarrow{F(h)} & F(X) \\
\downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle f} \\
A & \xrightarrow[\exists !\, h]{} & X
\end{array}
\qquad\qquad
\begin{array}{ccc}
X & \xrightarrow{\exists !\, h} & Z \\
\downarrow{\scriptstyle f} & & \downarrow{\scriptstyle \beta} \\
F(X) & \xrightarrow[F(h)]{} & F(Z)
\end{array}
$$

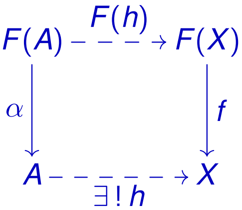
Initiality, finality

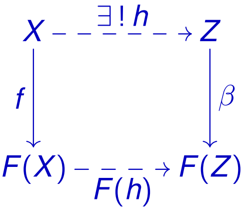




- initial algebras ↔ induction

Initiality, finality





- initial algebras $\leftrightarrow$ induction
- final coalgebras $\leftrightarrow$ coinduction

Automata, (co)algebraically

- Automata are complicated structures:

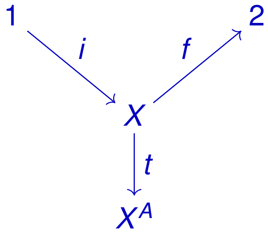
  part of them is algebra - part of them is coalgebra

Automata, (co)algebraically

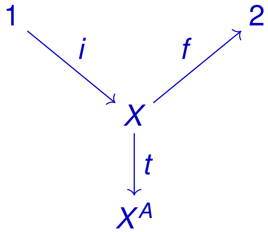
- Automata are complicated structures:

  part of them is algebra - part of them is coalgebra
- (. . . in two different ways . . .)

A deterministic automaton

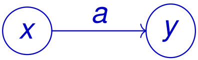


A deterministic automaton



where

$$1 = \{0\} \qquad 2 = \{0, 1\} \qquad X^A = \{\, g \mid g : A \to X \,\}$$



$\leftrightarrow \quad t(x)(a) = y$

$i(0) \in X$ is the initial state

is final (or accepting) $\leftrightarrow \quad f(x) = 1$

Automata: algebra or coalgebra?

- initial state: algebraic – final states: coalgebraic

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

Automata: algebra or coalgebra?

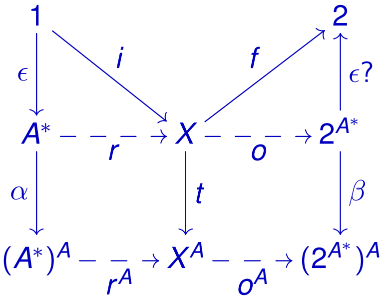
- initial state: algebraic – final states: coalgebraic

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

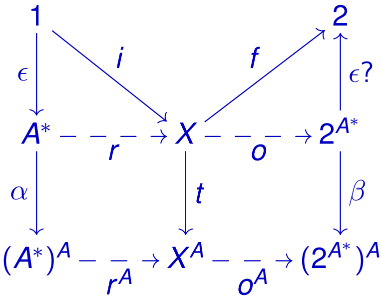
- transition function: both algebraic and coalgebraic

$$\dfrac{\dfrac{X \xrightarrow{\;t\;} X^A}{X \longrightarrow (A \longrightarrow X)}}{X \times A \xrightarrow{\;t\;} X}$$

Automata: algebra and coalgebra!

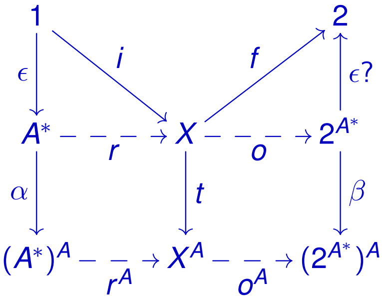


Automata: algebra and coalgebra!



To take home: this picture!! . . .

Automata: algebra and coalgebra!



To take home: this picture!! . . . which we'll explain next . . .

The "automaton" of languages

$$\begin{array}{c} 2 \\ \uparrow {\scriptstyle \epsilon?} \\ 2^{A^*} \\ \downarrow {\scriptstyle \beta} \\ (2^{A^*})^A \end{array}$$

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

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

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

The "automaton" of languages

$$\begin{array}{c} 2 \\ \uparrow \epsilon? \\ 2^{A^*} \\ \downarrow \beta \\ (2^{A^*})^A \end{array}$$

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

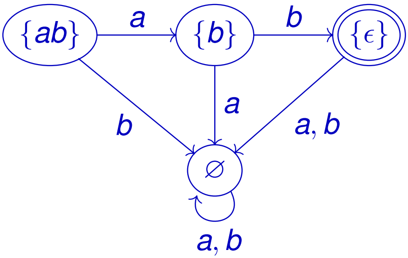
$$2^{A^*} = \{g \mid g : A^* \to 2\} \cong \{L \mid L \subseteq A^*\}$$

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

- We say "automaton": it does not have an initial state.

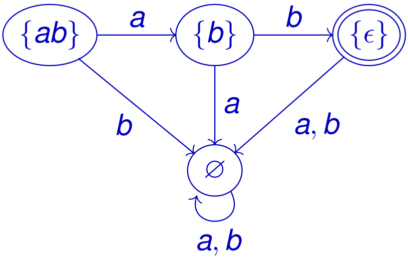
The automaton of languages

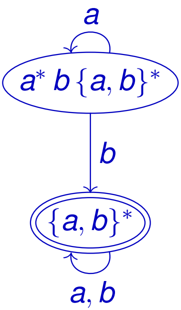
- transitions: $L \xrightarrow{a} L_a$ where $L_a = \{w \in A^* \mid a \cdot w \in L\}$
- for instance:



The automaton of languages

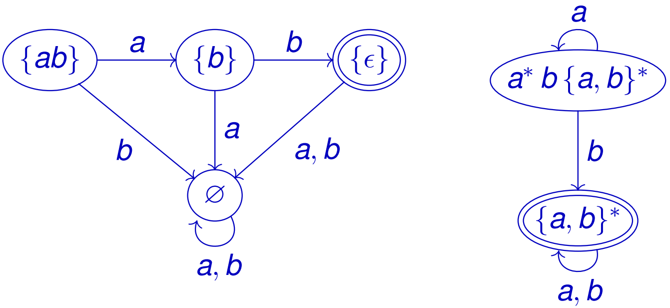
- transitions: $L \xrightarrow{a} L_a$ where $L_a = \{w \in A^* \mid a \cdot w \in L\}$
- for instance:





The automaton of languages

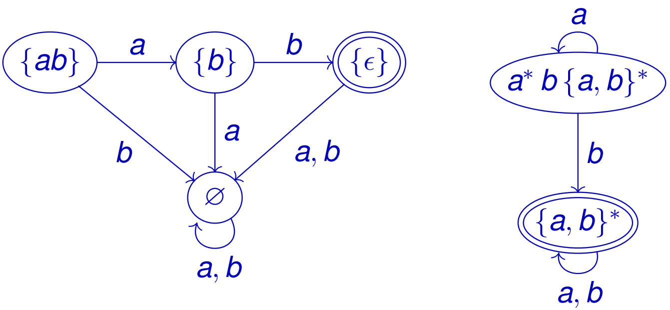
- transitions: $L \xrightarrow{a} L_a$ where $L_a = \{w \in A^* \mid a \cdot w \in L\}$
- for instance:



- note: every state L accepts . . .

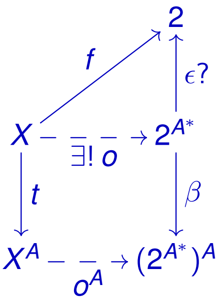
The automaton of languages

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



- note: every state L accepts . . . . . . the language L !!

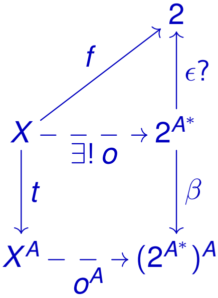
The automaton of languages is . . . final



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

$$= \text{the language accepted by } x$$

The automaton of languages is . . . final



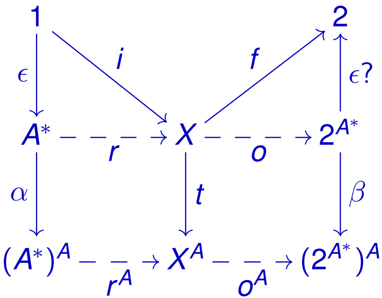
$$o(x) = \{w \in A^* \mid f(x_w) = 1\}$$

$$= \text{the language accepted by } x$$

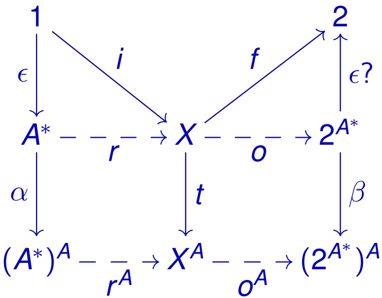
where: x_w is the state reached after inputting the word w,

and: $o^A(g) = o \circ g$, all $g \in X^A$.

Back to today's picture

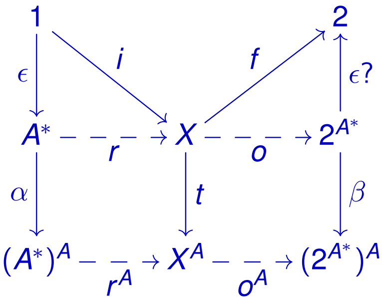


Back to today's picture



On the right: final coalgebra

Back to today's picture



On the right: final coalgebra

On the left: initial algebra . . .

The “automaton” of words

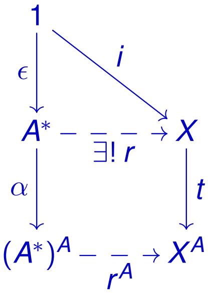
$$\begin{array}{c} 1 \\ \epsilon \big\downarrow \\ A^* \\ \alpha \big\downarrow \\ (A^*)^A \end{array}$$

ϵ is initial state

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

that is, transitions: $w \xrightarrow{a} w \cdot a$

The automaton of words is . . . initial



$i \in X$ = 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!

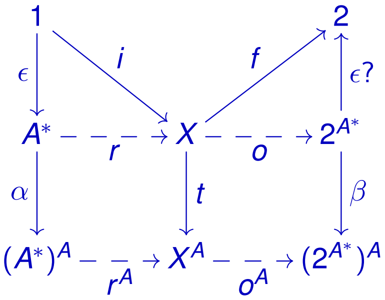
Duality

- Reachability and observability are dual:

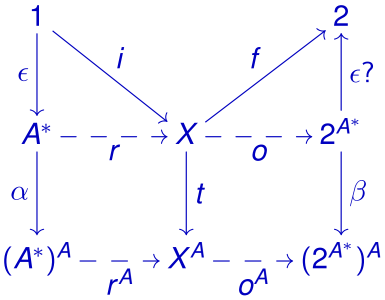
  Arbib and Manes, 1975.

- (here observable = minimal)

Reachability and observability



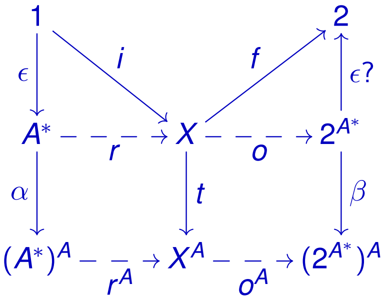
Reachability and observability



$r(w)$ = state reached on input w

$o(x)$ = language accepted by x

Reachability and observability

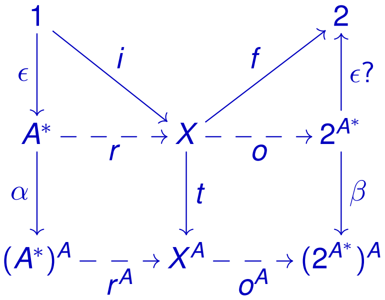


$r(w) \;=\;$ state reached on input w

$o(x) \;=\;$ language accepted by x

- We call X reachable if r is surjective.

Reachability and observability



$r(w)$ $=$ state reached on input w

$o(x)$ $=$ language accepted by x

- 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

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

Powerset construction

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

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

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

Powerset construction

$$2^{(-)} \; : \quad \begin{array}{c} V \\ {\scriptstyle g}\big\downarrow \\ W \end{array} \quad \mapsto \quad \begin{array}{c} 2^V \\ \big\uparrow{\scriptstyle 2^g} \\ 2^W \end{array}$$

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

$$2^g(S) = \; g^{-1}(S) \quad (= \; \{v \in V \mid \; g(v) \in S\} \,)$$

- This construction is contravariant !!

Powerset construction

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

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

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

- This construction is contravariant !!
- Note: if g is surjective, then 2^g is injective.

Reversing transitions

$$X \xrightarrow{t} X^A$$

Reversing transitions

$$\begin{array}{c|c} X & X \times A \\ {\scriptstyle t}\big\downarrow & \big\downarrow \\ X^A & X \end{array}$$

Reversing transitions

$$
\begin{array}{c|c}
X & X \times A \\
\big\downarrow t & \big\downarrow \\
X^A & X
\end{array}
\quad \overset{2^{(-)}}{\longmapsto} \quad
\begin{array}{c}
2^{X \times A} \\
\big\uparrow \\
2^X
\end{array}
$$

Reversing transitions

$$
\begin{array}{c|c}
X & X \times A \\
{\scriptstyle t}\downarrow & \downarrow \\
X^A & X
\end{array}
\quad \xmapsto{\;2^{(-)}\;} \quad
\begin{array}{c|c}
2^{X \times A} & (2^X)^A \\
\uparrow & \uparrow \\
2^X & 2^X
\end{array}
$$

Reversing transitions

$$
\begin{array}{c}
X \\
t \big\downarrow \\
X^A
\end{array}
\;\Bigg\|\;
\begin{array}{c}
X \times A \\
\big\downarrow \\
X
\end{array}
\quad \overset{2^{(-)}}{\longmapsto} \quad
\begin{array}{c}
2^{X \times A} \\
\big\uparrow \\
2^X
\end{array}
\;\Bigg\|\;
\begin{array}{c}
(2^X)^A \\
\big\uparrow \\
2^X
\end{array}
\;\Bigg\|\;
\begin{array}{c}
2^X \\
\big\downarrow 2^t \\
(2^X)^A
\end{array}
$$

Initial $\leftrightarrow$ final

$$1 \xrightarrow{i} X$$

Initial $\leftrightarrow$ final

$$1 \xrightarrow{\;i\;} X \qquad \xmapsto{\;2^{(-)}\;} \qquad 2^X \xrightarrow{\;2^i\;} 2$$

Initial ↔ final

$$1 \xrightarrow{\;i\;} X \qquad \xmapsto{\;2^{(-)}\;} \qquad 2^X \xrightarrow{\;2^i\;} 2$$

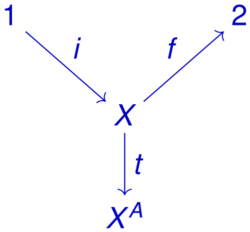
$$X \xrightarrow{\;f\;} 2$$

Initial $\leftrightarrow$ final

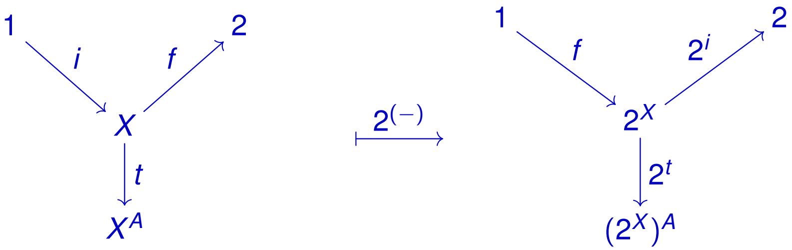
$$1 \xrightarrow{i} X \qquad \overset{2^{(-)}}{\longmapsto} \qquad 2^X \xrightarrow{2^i} 2$$

$$X \xrightarrow{f} 2 \qquad \longleftrightarrow \qquad 1 \xrightarrow{f} 2^X$$

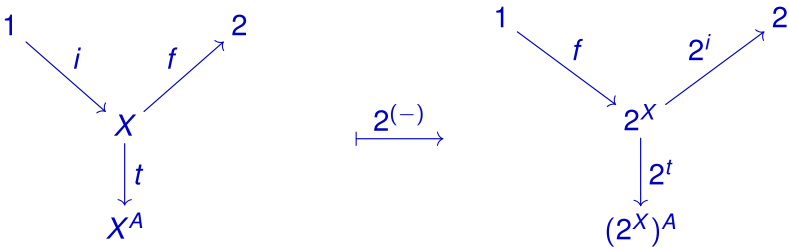
Reversing the entire automaton



Reversing the entire automaton

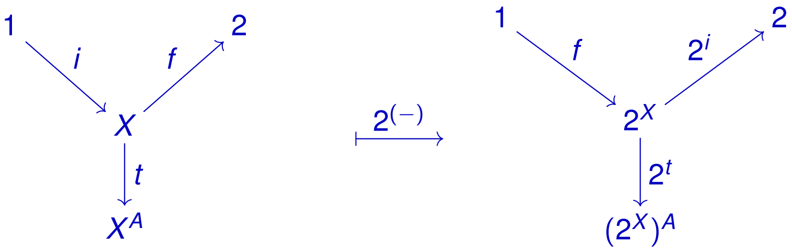


Reversing the entire automaton



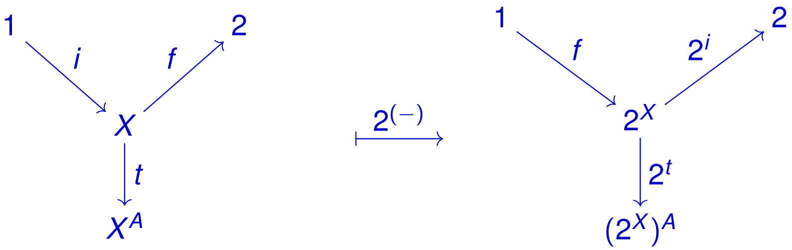
- Initial and final are exchanged . . .

Reversing the entire automaton



- Initial and final are exchanged . . .
- transitions are reversed . . .

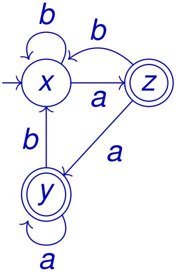
Reversing the entire automaton



- Initial and final are exchanged . . .
- transitions are reversed . . .
- and the result is again deterministic!

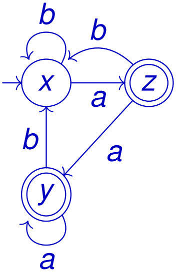
Our previous example

$X =$

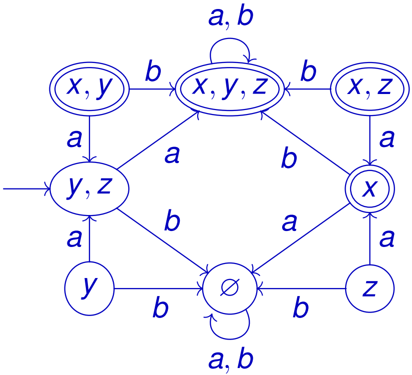


Our previous example

$X =$

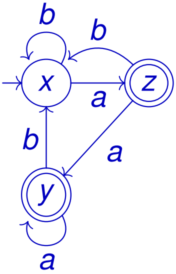


$2^X =$

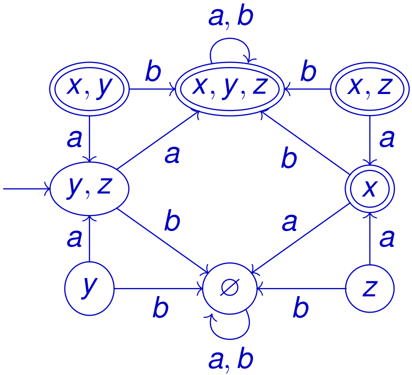


Our previous example

$X =$



$2^X =$



- Note that X has been reversed and determinized:

$$2^X = det(rev(X))$$

Proving today's Theorem

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

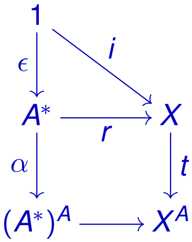
Proving today's Theorem

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

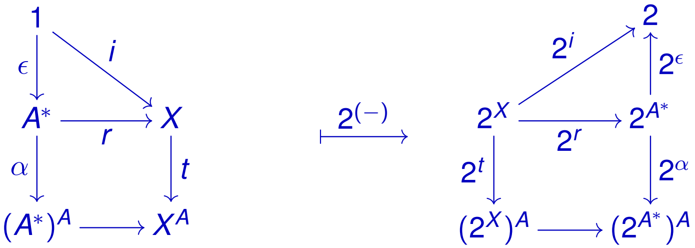
then: 2^X ($= \; det(rev(X))$) is minimal/observable and

$$L(2^X) = reverse(L(X))$$

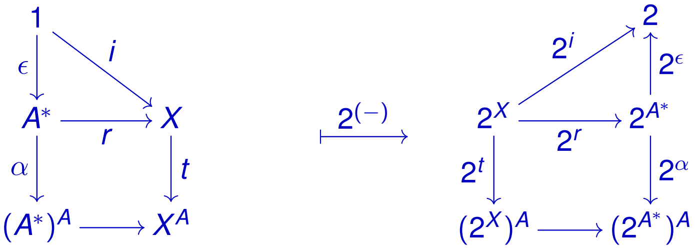
Proof: by reversing $A^* \xrightarrow{r} X$



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

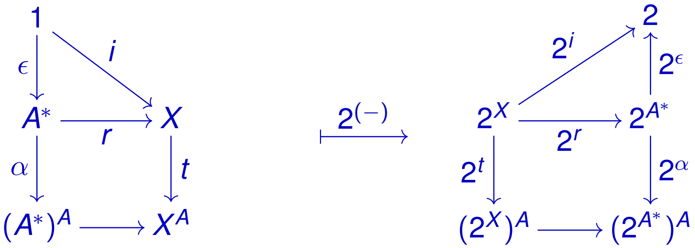


Proof: by reversing $A^* \xrightarrow{r} X$



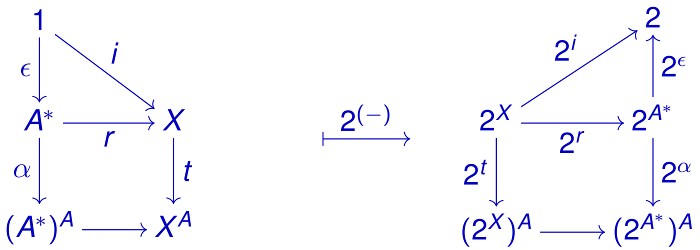
- X becomes 2^X

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



- X becomes 2^X
- initial automaton A^* becomes (almost) final automaton 2^{A^*}

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

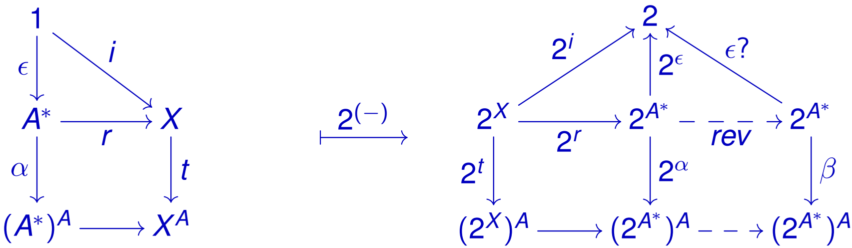


- X becomes 2^X
- initial automaton A^* becomes (almost) final automaton 2^{A^*}
- r is surjective $\Rightarrow$ 2^r is injective

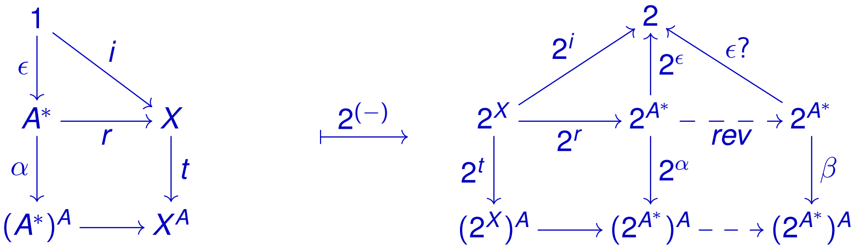
Reachable becomes observable

$$\begin{array}{ccc} 1 & & \\ {\scriptstyle \epsilon}\downarrow & \searrow^{i} & \\ A^* & \xrightarrow[r]{} & X \\ {\scriptstyle \alpha}\downarrow & & \downarrow{\scriptstyle t} \\ (A^*)^A & \longrightarrow & X^A \end{array}$$

Reachable becomes observable

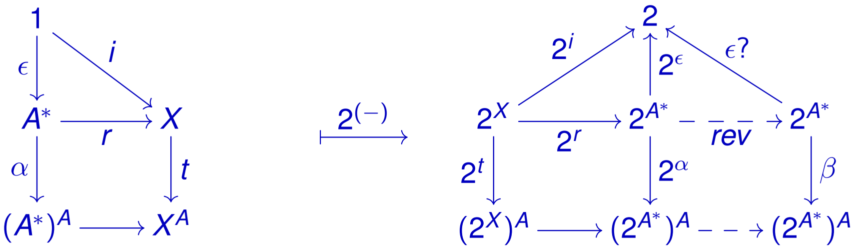


Reachable becomes observable



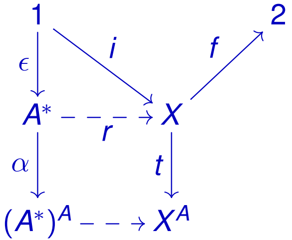
- If r is surjective then (2^r and hence) $rev \circ 2^r$ is injective.

Reachable becomes observable

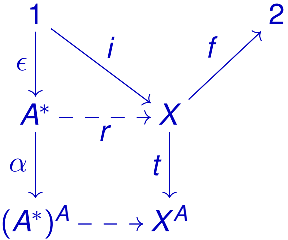


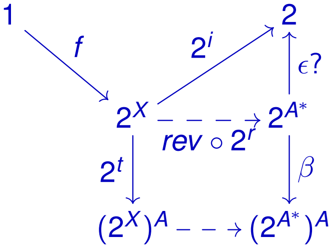
- If r is surjective then (2^r and hence) $rev \circ 2^r$ is injective.
- That is, 2^X is observable (= minimal).

Summarizing

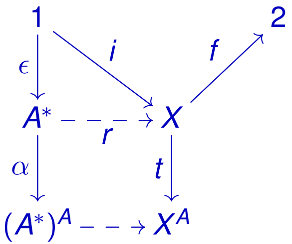


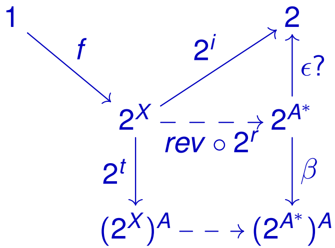
Summarizing





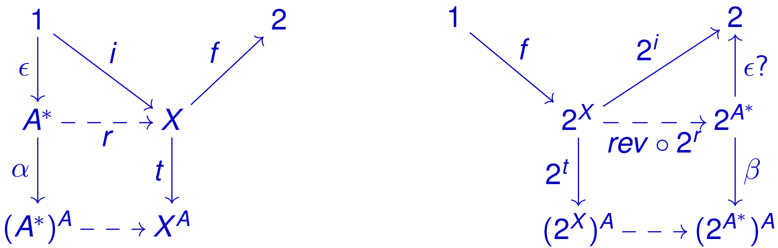
Summarizing





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

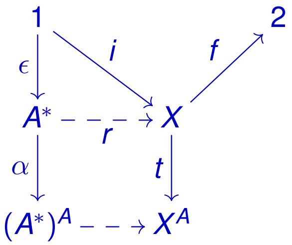
Summarizing

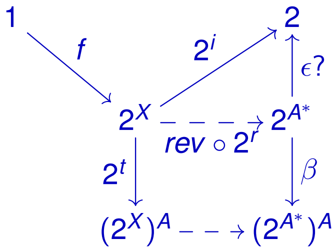


- If: X is reachable, i.e., r is surjective

  then: $rev \circ 2^r$ is injective, i.e., 2^X is observable = minimal.

Summarizing





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

  then: $rev \circ 2^r$ is injective, i.e., 2^X is observable = minimal.
- And: $rev(2^r(f)) = rev(o(i))$, i.e., $L(\, 2^X \,) \; = \; reverse(\, L(X)\,)$

Corollary: Brzozowski's algorithm

- X becomes 2^X, accepting $reverse(L(X))$

Corollary: Brzozowski's algorithm

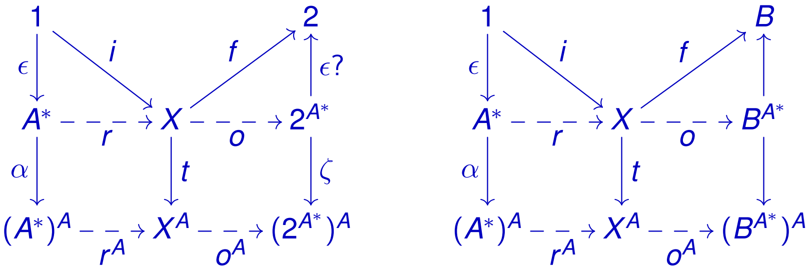
- X becomes 2^X, accepting $\mathit{reverse}(L(X))$
- take reachable part: $Y = \mathit{reachable}(2^X)$

Corollary: Brzozowski's algorithm

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

$$\mathit{reverse}(\mathit{reverse}(L(X))) = L(X)$$

Generalizations



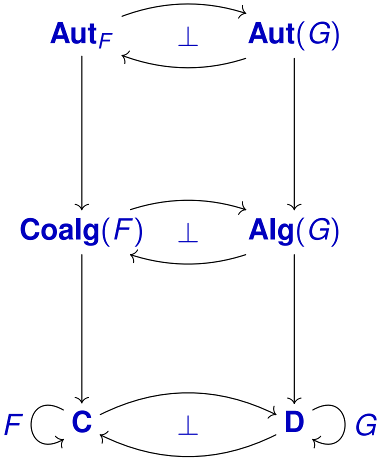
- A Brzozowski minimization algorithm for Moore automata.

$$B^X = \{\varphi \mid \varphi\colon X \to B\} \qquad B^f(\varphi) = \varphi \circ f$$

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



Conclusions

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

Conclusions

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

Thanks!