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.

  J.J.M.M. Rutten. ***Automata and coinduction (an exercise in coalgebra).*** *CONCUR'98*

Motivation (by example)

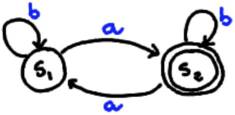
- Coalgebras are a suitable framework to study the behaviour of dynamical systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.

  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.

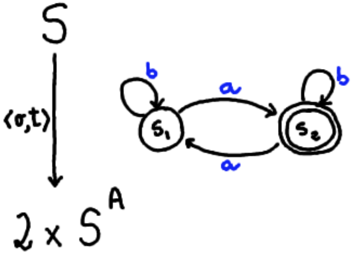
  J.J.M.M. Rutten. ***Automata and coinduction (an exercise in coalgebra).*** *CONCUR'98*



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.

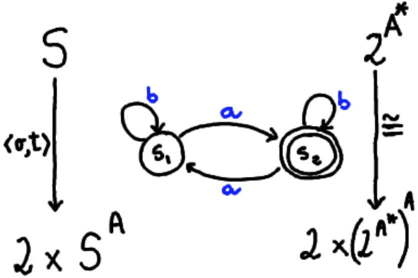
  J.J.M.M. Rutten. ***Automata and coinduction (an exercise in coalgebra).*** *CONCUR'98*



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.

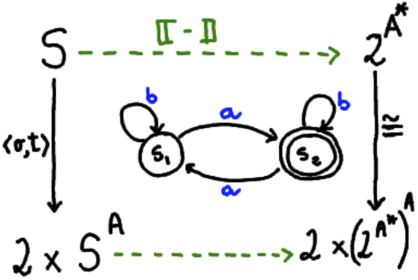
  J.J.M.M. Rutten. ***Automata and coinduction (an exercise in coalgebra).*** *CONCUR'98*



Motivation (by example)

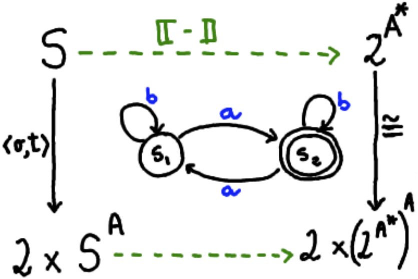
- Coalgebras are a suitable framework to study the behaviour of dynamical systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.

  J.J.M.M. Rutten. ***Automata and coinduction (an exercise in coalgebra).*** *CONCUR'98*



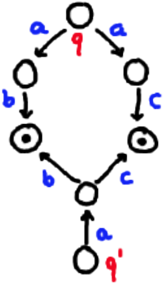
Motivation (by example)

- Coalgebras are a suitable framework to study the behaviour of dynamical systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.
  J.J.M.M. Rutten. ***Automata and coinduction (an exercise in coalgebra).*** *CONCUR'98*

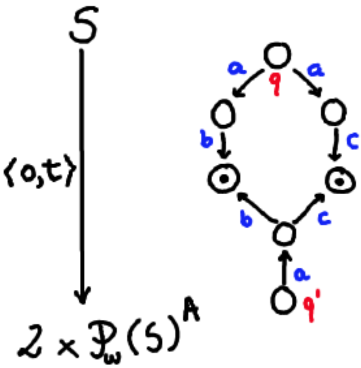


$[\![s_1]\!] = \{ w \mid |w|_a \text{ is odd} \}$

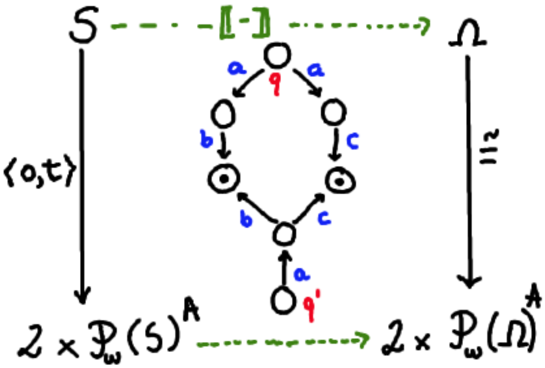
Motivation (by example, cont.)



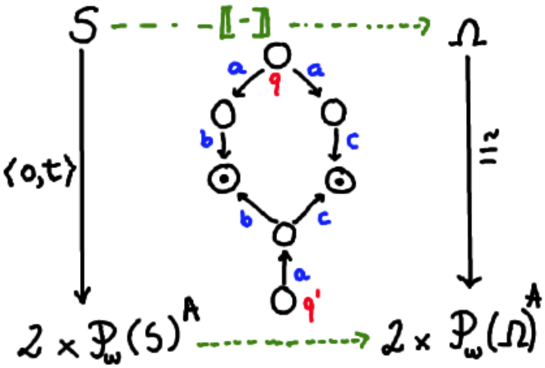
Motivation (by example, cont.)



Motivation (by example, cont.)

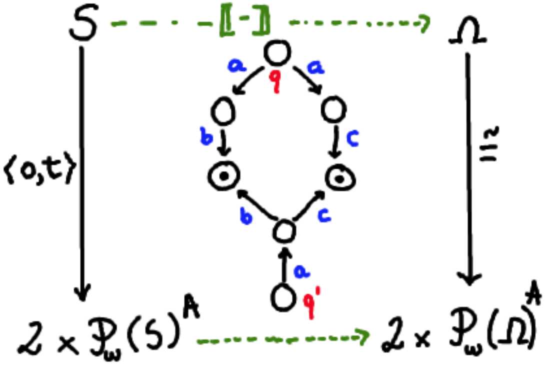


Motivation (by example, cont.)



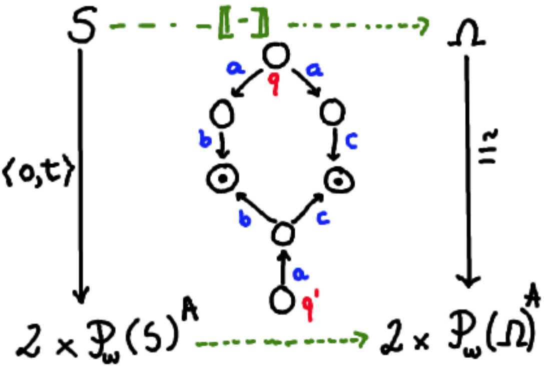
$[\![q]\!] \neq [\![q']\!]$ (different branching structure)

Motivation (by example, cont.)



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

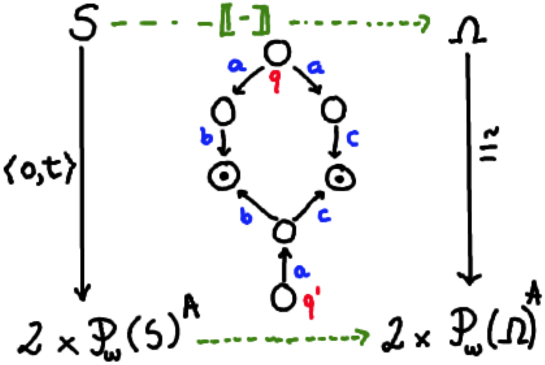
Motivation (by example, cont.)



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

How do we study NDA wrt language equivalence?

Motivation (by example, cont.)

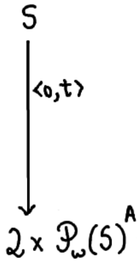


$[\![q]\!] \neq [\![q']\!]$ (different branching structure) but: $L_q = L_{q'} = \{ab, ac\}$

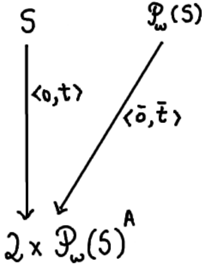
How do we study NDA wrt language equivalence?

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

Example I: Determinizing (coalgebraically)

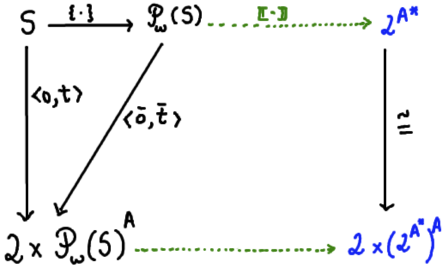


Example I: Determinizing (coalgebraically)



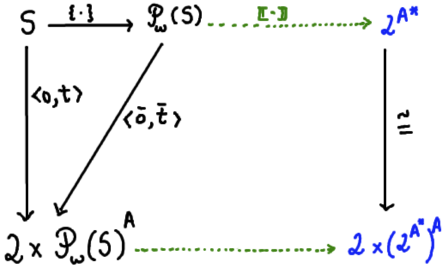
$$\overline{o}(Q) = \begin{cases} 1 & \exists_{q\in Q} o(q) = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \bar{t}(Q)(a) = \bigcup_{q\in Q} t(q)(a)$$

Example I: Determinizing (coalgebraically)



$$\overline{o}(Q) = \begin{cases} 1 & \exists_{q \in Q} o(q) = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \overline{t}(Q)(a) = \bigcup_{q \in Q} t(q)(a)$$

Example I: Determinizing (coalgebraically)

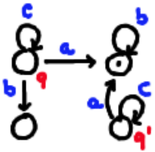


$$\overline{o}(Q) = \begin{cases} 1 & \exists_{q \in Q} o(q) = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \overline{t}(Q)(a) = \bigcup_{q \in Q} t(q)(a)$$

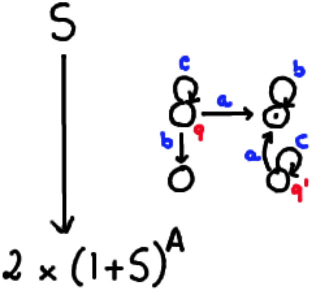
How do we study NDA wrt language equivalence?

$$L_s = [\![\{s\}]\!]$$

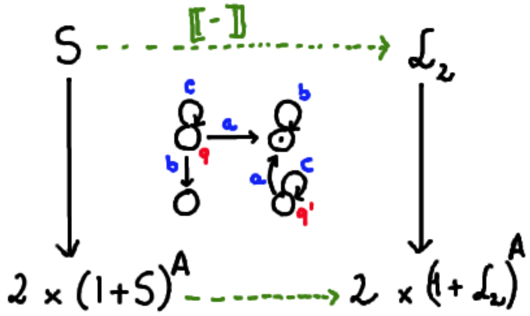
Motivation (by example, cont.)



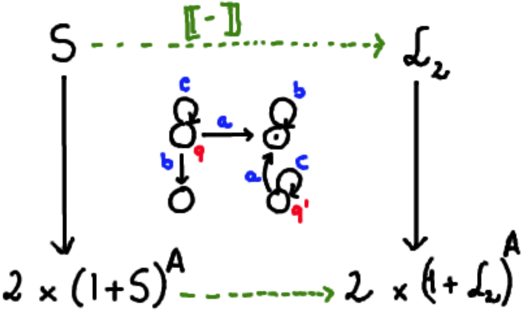
Motivation (by example, cont.)



Motivation (by example, cont.)



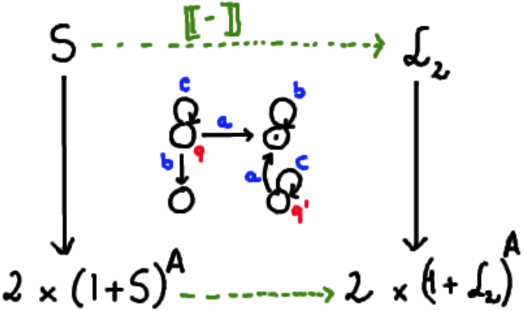
Motivation (by example, cont.)



$\mathcal{L}_2$ are pairs of languages $\langle V, W\rangle$ (<accepted words, domain>)

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

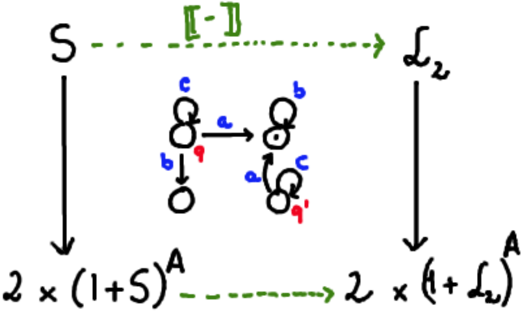
Motivation (by example, cont.)



$\mathcal{L}_2$ 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: $L_q = L_{q'} = c^* ab^*$

Motivation (by example, cont.)

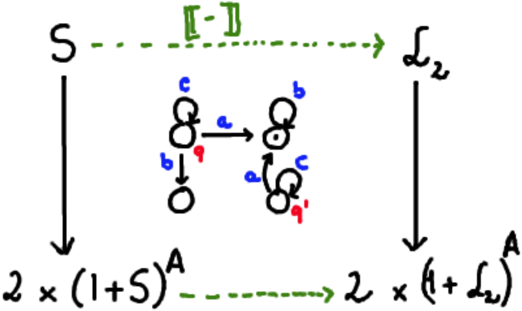


$\mathcal{L}_2$ are pairs of languages $\langle V, W \rangle$ (<accepted words, domain>)

$[\![q]\!] = \langle c^*ab^*, b + c^* + c^*ab^* \rangle \neq \langle c^*ab^*, c^* + c^*ab^* \rangle = [\![q']\!]$
but: $L_q = L_{q'} = c^*ab^*$

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

Motivation (by example, cont.)



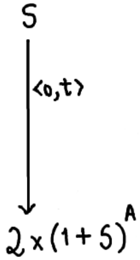
$\mathcal{L}_2$ are pairs of languages $\langle V, W \rangle$ (<accepted words, domain>)

$[\![q]\!] = \langle c^*ab^*, b + c^* + c^*ab^* \rangle \neq \langle c^*ab^*, c^* + c^*ab^* \rangle = [\![q']\!]$
but: $L_q = L_{q'} = c^*ab^*$

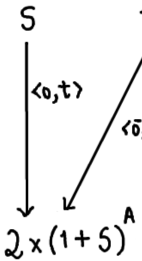
How do we study PA wrt (accepted) language equivalence?

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

Example II: Totalizing (coalgebraically)

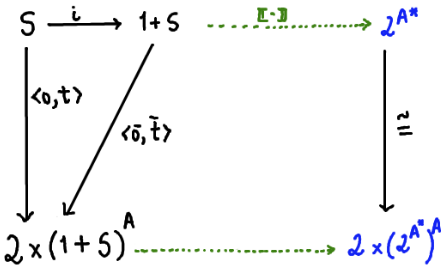


Example II: Totalizing (coalgebraically)



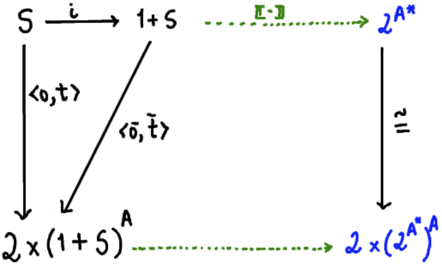
$$\begin{cases} \overline{o}(*) = 0 \\ \overline{o}(s) = o(s) \end{cases} \qquad \begin{cases} \overline{t}(*)(a) = * \\ \overline{t}(s)(a) = t(s)(a) \end{cases}$$

Example II: Totalizing (coalgebraically)



$$\begin{cases} \overline{o}(*) = 0 \\ \overline{o}(s) = o(s) \end{cases} \qquad \begin{cases} \overline{t}(*)(a) = * \\ \overline{t}(s)(a) = t(s)(a) \end{cases}$$

Example II: Totalizing (coalgebraically)



$$\begin{cases} \overline{o}(*) = 0 \\ \overline{o}(s) = o(s) \end{cases} \qquad \begin{cases} \overline{t}(*)(a) = * \\ \overline{t}(s)(a) = t(s)(a) \end{cases}$$

How do we study PA wrt language equivalence?

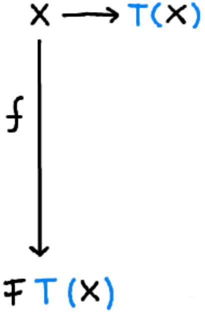
$$L_s = [\![\, i(s) \,]\!]$$

Chasing the pattern. . .

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

Chasing the pattern. . .

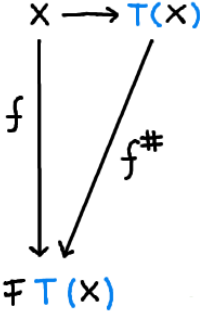
How do we capture both examples (and more) in the same framework?



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

Chasing the pattern. . .

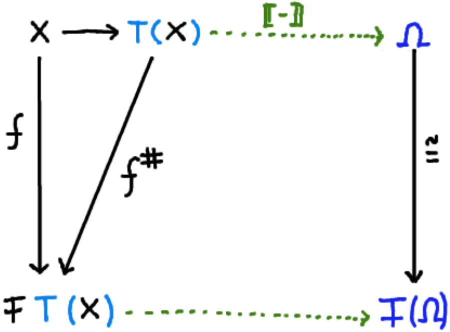
How do we capture both examples (and more) in the same framework?



The state space was *enriched* : T monad ($\mathcal{P}$, $1+$, . . .).
Transform an FT-coalgebra (X,f) into an F-coalgebra ($T(X), f^\sharp$).

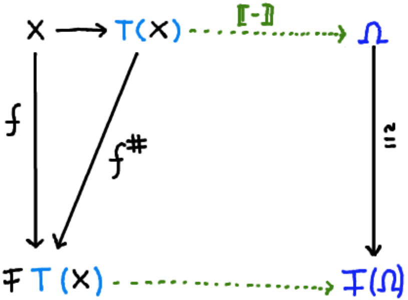
Chasing the pattern...

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



The state space was *enriched* : T monad ($\mathcal{P}$, $1+$, ...).
Transform an FT-coalgebra (X,f) into an F-coalgebra $(T(X), f^{\sharp})$.
If F has final coalgebra: $x_1 \approx_F^T x_2 \Leftrightarrow [\![\eta_X(x_1)]\!] = [\![\eta_X(x_2)]\!]$.

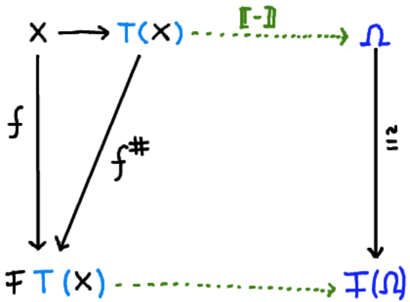
In a nutshell. . .



Ingredients:

- A monad T;
- A final coalgebra for F (for instance, take F to be bounded);
- An extension $f^\sharp$ of f;

In a nutshell...



Ingredients:

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

$$f^\sharp\colon T(X) \xrightarrow{T(f)} T(F(T(X))) \xrightarrow{h} F(T(X))$$

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 $\approx_F^T$ and $\sim_F$?

Examples revisited

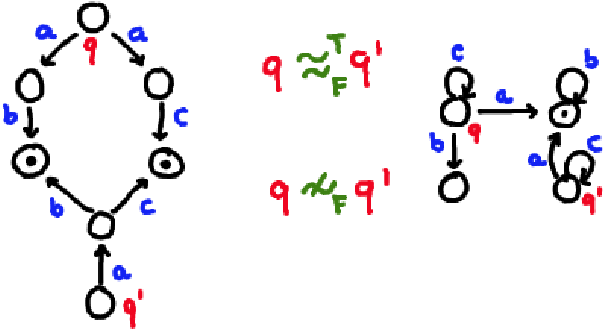
NFA $F(X) = 2 \times X^A$, $T = \mathcal{P}$, $2 \times \mathcal{P}(X)^A$ is a join-semilattice;

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

What is the relation between $\approx_F^T$ and $\sim_F$?

Examples revisited

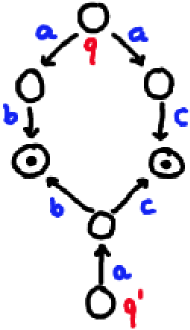
NFA $F(X) = 2 \times X^A$, $T = \mathcal{P}$, $2 \times \mathcal{P}(X)^A$ is a join-semilattice;
PA $F(X) = 2 \times X^A$, $T = 1 + -$, $2 \times (1 + X)^A$ is a pointed set.



What is the relation between $\approx_F^T$ and $\sim_F$?

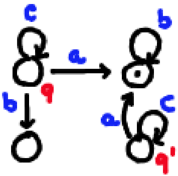
Examples revisited

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.



$q \approx_F^T q'$

$q \not\sim_F q'$



What is the relation between $\approx_F^T$ and $\sim_F$?

Bisimilarity implies linear bisimilarity

Theorem

$$\sim_F \quad \Rightarrow \quad \approx_F^T$$

Bisimilarity implies linear bisimilarity

Theorem

$$\sim_F \quad \Rightarrow \quad \approx_F^T$$

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 ($F(X) = [0,1] \times X^A$, $T = \mathcal{D}_\omega$) that probabilistic bisimilarity implies weighted language equivalence.

Examples, Examples, Examples,...

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

Conclusions

- 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!!

Conclusions

- 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!!