Generalized Determinizations via Coalgebras

Alexandra Silva[3,1] Filippo Bonchi[4] Marcello Bonsangue[1,2]
Jan Rutten[1,3]

[1]Centrum Wiskunde en Informatica
[2]LIACS - Leiden University
[3]Radboud Universiteit Nijmegen
[4]INRIA Saclay - LIX, École Polytechnique

Indiana University, August 2013

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.

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

Motivation (by example)

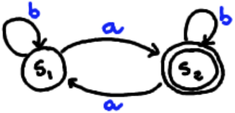
- Coalgebras are a suitable framework to study the behaviour of dynamical systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.

  Jan 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.

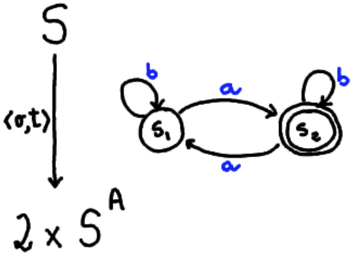
  Jan Rutten. ***Automata and coinduction (an exercise in coalgebra).*** *CONCUR'98*



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.

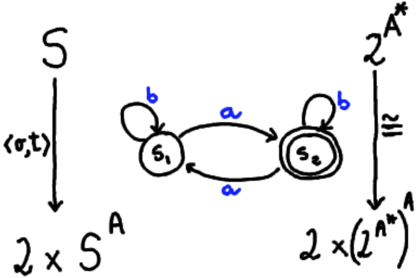
  Jan Rutten. ***Automata and coinduction (an exercise in coalgebra).*** *CONCUR'98*



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.

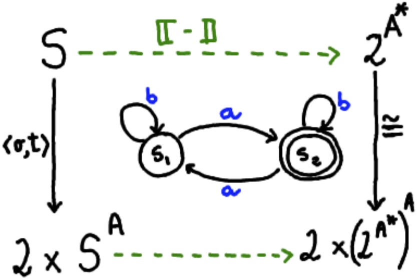
  Jan Rutten. ***Automata and coinduction (an exercise in coalgebra).*** *CONCUR'98*



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.

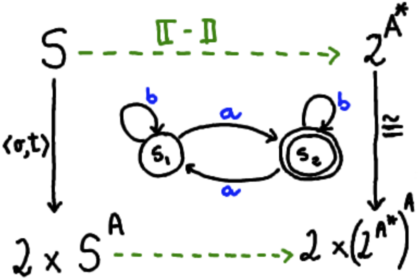
  Jan Rutten. ***Automata and coinduction (an exercise in coalgebra).*** *CONCUR'98*



Motivation (by example)

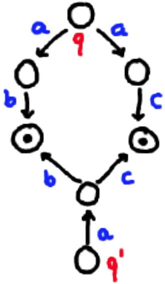
- Coalgebras are a suitable framework to study the behaviour of dynamical systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.

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



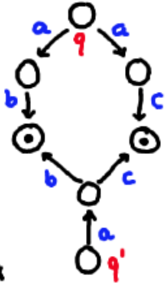
$$[\![s_1]\!] = \{ w \mid |w|_a \text{ is odd} \}$$

Motivation (by example, cont.)

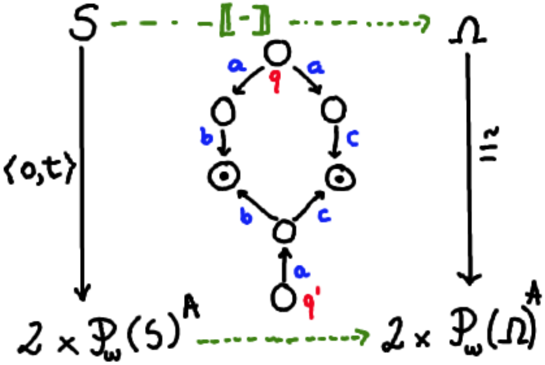


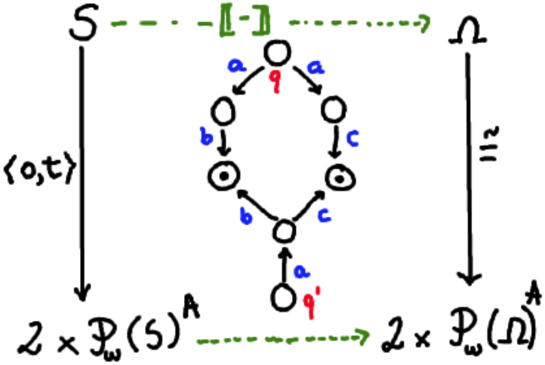
Motivation (by example, cont.)

$$S \xrightarrow{\langle o,t\rangle} 2 \times \mathcal{P}_\omega(S)^A$$



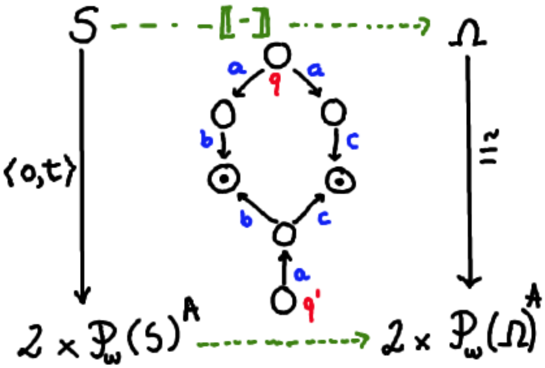
Motivation (by example, cont.)





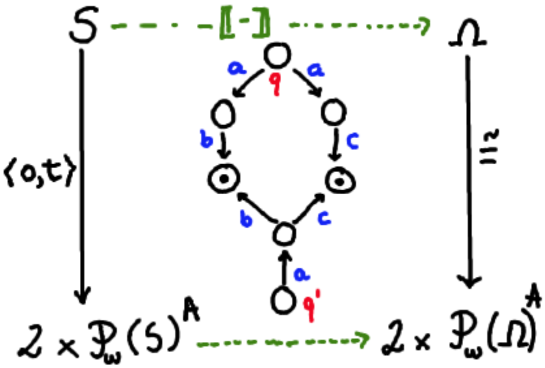
$[\![q]\!] \neq [\![q']\!]$ (different branching structure)

Motivation (by example, cont.)



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

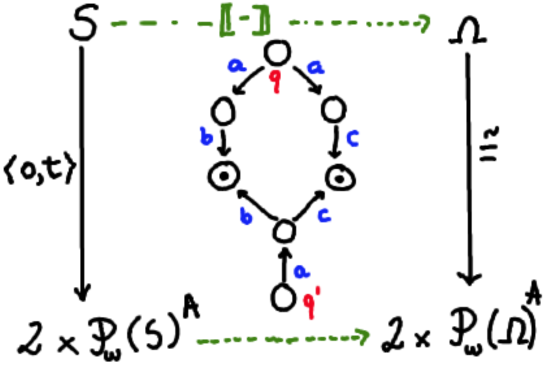
Motivation (by example, cont.)



$[\![\, q \,]\!] \neq [\![\, q' \,]\!]$ (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)

$$\begin{tikzcd} S \arrow[d, "\langle o,t \rangle"] \\ 2 \times \mathcal{P}_\omega(S)^A \end{tikzcd}$$

Example I: Determinizing (coalgebraically)

$$\begin{tikzcd}
S \arrow[d, "\langle o, t \rangle"] & \mathcal{P}_\omega(S) \arrow[dl, "\langle \bar{o}, \bar{t} \rangle"] \\
2 \times \mathcal{P}_\omega(S)^A &
\end{tikzcd}$$

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

Example I: Determinizing (coalgebraically)

$$
\begin{tikzcd}
S \arrow[r, "\{\cdot\}"] \arrow[d, "\langle o,t \rangle"'] & \mathcal{P}_\omega(S) \arrow[rr, dotted, "[\![\cdot]\!]"] \arrow[dl, "\langle \bar{o},\bar{t} \rangle"] & & 2^{A^*} \arrow[d, "\cong"] \\
2 \times \mathcal{P}_\omega(S)^A \arrow[rrr, dotted] & & & 2 \times (2^{A^*})^A
\end{tikzcd}
$$

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

Example I: Determinizing (coalgebraically)

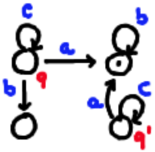
$$
\begin{tikzcd}
S \arrow[r, "\{\cdot\}"] \arrow[d, "\langle o,t \rangle"'] & \mathcal{P}_\omega(S) \arrow[r, dotted, "\llbracket \cdot \rrbracket"] \arrow[dl, "\langle \bar{o},\bar{t} \rangle"] & 2^{A^*} \arrow[d, "\cong"] \\
2 \times \mathcal{P}_\omega(S)^A \arrow[rr, dotted] & & 2 \times (2^{A^*})^A
\end{tikzcd}
$$

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

How do we study NDA wrt language equivalence?

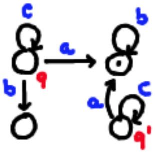
$$L_s = \llbracket \{s\} \rrbracket$$

Motivation (by example, cont.)

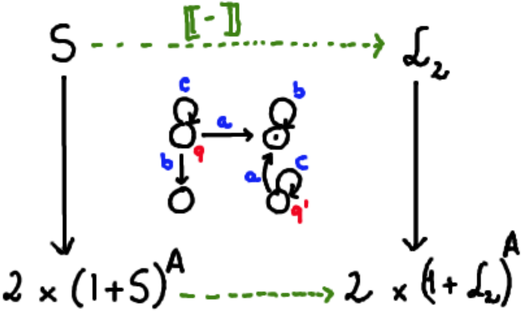


Motivation (by example, cont.)

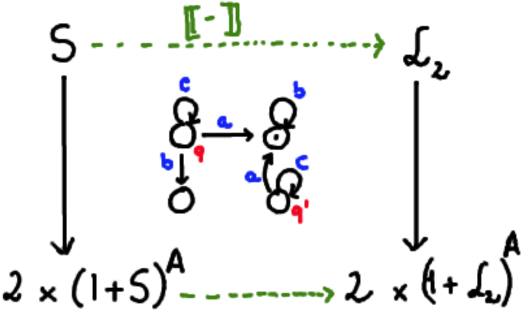
$$
\begin{tikzcd}
S \arrow[d] \\
2 \times (1+S)^A
\end{tikzcd}
$$



Motivation (by example, cont.)



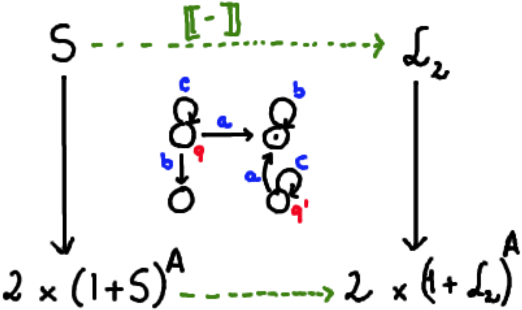
Motivation (by example, cont.)



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

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

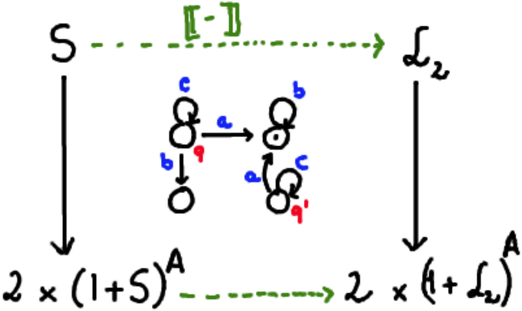
Motivation (by example, cont.)



$\mathcal{L}_2$ 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.)

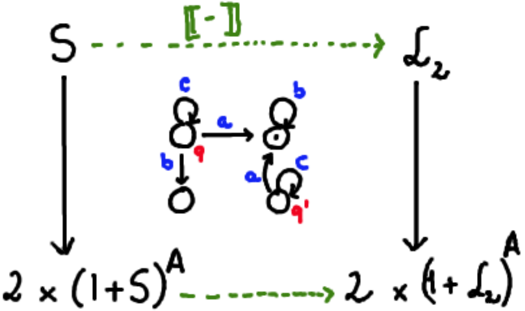


$\mathcal{L}_2$ are pairs of languages $\langle V, W \rangle$ (<accepted words, domain>)

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

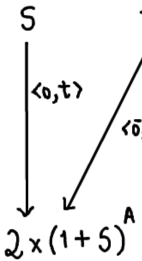
How do we study PA wrt (accepted) language equivalence?

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

Example II: Totalizing (coalgebraically)

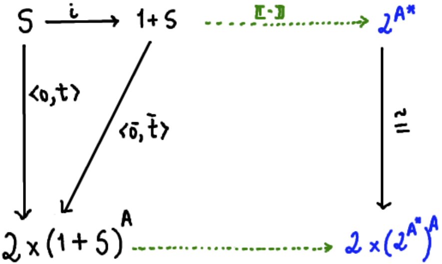
$$
\begin{tikzcd}
S \arrow[d, "{\langle o,t \rangle}"] \\
2 \times (1+S)^A
\end{tikzcd}
$$

Example II: Totalizing (coalgebraically)



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

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)

$$\begin{tikzcd}
S \arrow[r, "i"] \arrow[d, "\langle o,t \rangle"'] & 1+S \arrow[r, dashed, "[\![-]\!]"] \arrow[dl, "\langle \bar{o},\bar{t} \rangle"] & 2^{A^*} \arrow[d, "\cong"] \\
2 \times (1+S)^A \arrow[rr, dashed] & & 2 \times (2^{A^*})^A
\end{tikzcd}$$

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

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?

$$
\begin{tikzcd}
X \arrow[r] \arrow[d, "f"'] & T(X) \\
FT(X) &
\end{tikzcd}
$$

The state space has now *structure* : T monad ($\mathcal{P}$, $1+$, ...).

Chasing the pattern...

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

$$\begin{tikzcd}
X \arrow[r] \arrow[d, "f"'] & T(X) \arrow[dl, "f^{\#}"] \\
FT(X) &
\end{tikzcd}$$

The state space has now *structure* : 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?

$$\begin{tikzcd}
X \arrow[r] \arrow[d, "f"'] & T(X) \arrow[r, dashed, "[\![-]\!]"] \arrow[dl, "f^\sharp"] & \Omega \arrow[d, "\cong"] \\
FT(X) \arrow[rr, dashed] & & F(\Omega)
\end{tikzcd}$$

The state space has now *structure* : 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. . .

$$\begin{tikzcd}
X \arrow[r] \arrow[d, "f"'] & T(X) \arrow[r, dashed, "[\![-]\!]"] \arrow[dl, "f^{\#}"] & \Omega \arrow[d, "\cong"] \\
FT(X) \arrow[rr, dashed] & & F(\Omega)
\end{tikzcd}$$

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. . .

$$\begin{tikzcd}
X \arrow[r] \arrow[d, "f"'] & T(X) \arrow[r, dashed, "{[\![-]\!]}"] \arrow[dl, "f^{\#}"] & \Omega \arrow[d, "\cong"] \\
FT(X) \arrow[rr, dashed] & & F(\Omega)
\end{tikzcd}$$

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

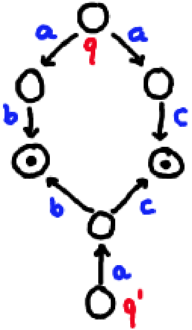
NFA $F(X) = 2 \times X^A$, $T = \mathcal{P}$, $2 \times \mathcal{P}(X)^A$ is a join-semilattice;

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 $\sim_F$?

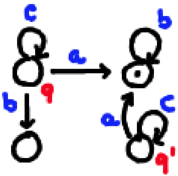
Examples revisited

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.



$q \approx^T_F q'$

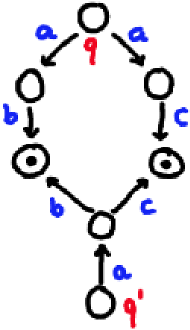
$q \not\sim_F q'$



What is the relation between $\approx^T_F$ and $\sim_F$?

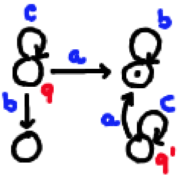
Examples revisited

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.



$q \approx_F^T q'$

$q \not\sim_F q'$



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

Bisimilarity implies linear bisimilarity

Theorem

$$\sim_F \;\Rightarrow\; \approx_F^T$$

Bisimilarity implies linear bisimilarity

Theorem

$$\sim_F \;\Rightarrow\; \approx_F^T$$

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!!