The generalized powerset construction

applications to semantics and concurrency

Alexandra Silva

alexandra@cs.ru.nl

http://www.cs.ru.nl/~alexandra/

Institute for Computing and Information Sciences
Radboud University Nijmegen

MFPS 2014, Cornell University
Ithaca, NY

What is this talk about in one slide

- Coalgebraic techniques in automata and concurrency theory;
- One particular construction: the subset construction;
- Generalizations and applications thereof.

Coalgebraic techniques

- Coalgebras – behaviour of dynamical systems.

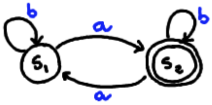
Coalgebraic techniques

- Coalgebras – behaviour of dynamical systems.
- Coalgebraic approach by example.

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

Coalgebraic techniques

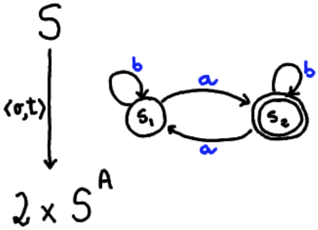
- Coalgebras – behaviour of dynamical systems.
- Coalgebraic approach by example.



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

Coalgebraic techniques

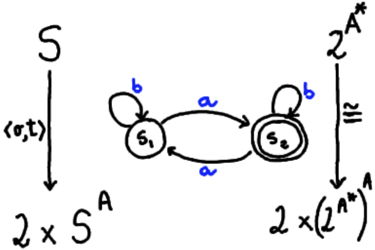
- Coalgebras – behaviour of dynamical systems.
- Coalgebraic approach by example.



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

Coalgebraic techniques

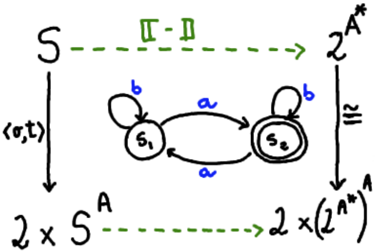
- Coalgebras – behaviour of dynamical systems.
- Coalgebraic approach by example.



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

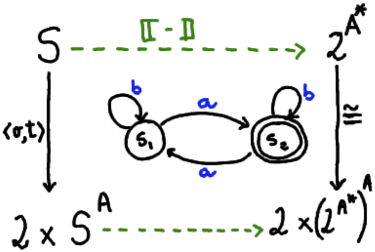
Coalgebraic techniques

- Coalgebras – behaviour of dynamical systems.
- Coalgebraic approach by example.



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

- Coalgebras – behaviour of dynamical systems.
- Coalgebraic approach by example.



$$[\![\, s_1 \,]\!] = \{w \in \{a,b\}^* \mid |w|_a \text{ is odd}\}$$

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

Coalgebraic techniques

$$S \to 2 \times S^A \qquad\qquad S \to F(S)$$

The power of F

- a universe of behaviors (final coalgebra);
- a canonical notion of behavioral equivalence;

Coalgebraic techniques

$$S \to 2 \times S^A \qquad\qquad S \to F(S)$$

The power of F

- a universe of behaviors (final coalgebra);
- a canonical notion of behavioral equivalence;

deterministic automata	language equivalence
infinite streams	pointwise equality
labelled transition systems	branching bisimilarity

Coalgebraic techniques

$$S \to 2 \times S^A \qquad\qquad S \to F(S)$$

The power of F

- a universe of behaviors (final coalgebra);
- a canonical notion of behavioral equivalence;

deterministic automata	language equivalence
infinite streams	pointwise equality
labelled transition systems	branching bisimilarity

- a specification language/logic;
- a sound and complete axiomatization thereof.

Coalgebraic techniques

$$S \to 2 \times S^A \qquad\qquad S \to F(S)$$

The power of F

- a universe of behaviors (final coalgebra);
- a canonical notion of behavioral equivalence;

deterministic automata	language equivalence
infinite streams	pointwise equality
labelled transition systems	branching bisimilarity

- a specification language/logic;
- a sound and complete axiomatization thereof.
- algorithms (wishful thinking: TCS-A meets TCS-B).

Universal coalgebra

Universal coalgebra

=

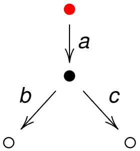
canonical notions/techniques based on the system type

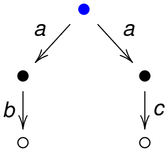
Universal coalgebra

Universal coalgebra
=
canonical notions/techniques based on the system type

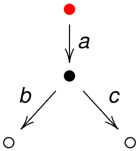
But what happens when canonical is not so canonical?

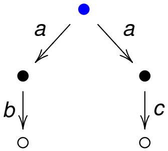
Example I: NDA



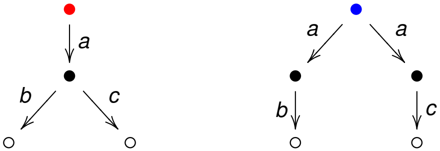


Example I: NDA





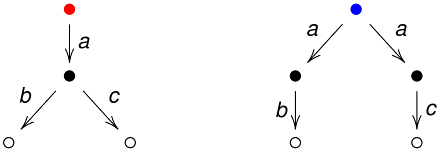
$$\begin{tikzcd}
S \arrow[r, dashed, "{[\![-]\!]}"] \arrow[d, "{\langle o,t\rangle}"'] & \Omega \arrow[d, "\cong"] \\
2 \times \mathcal{P}_\omega(S)^A \arrow[r, dashed] & 2 \times \mathcal{P}_\omega(\Omega)^A
\end{tikzcd}$$



- $[\![\, \textcolor{red}{\bullet} \,]\!] \neq [\![\, \textcolor{blue}{\bullet} \,]\!]$ (branching structure).

$$
\begin{tikzcd}
S \arrow[r, dashed, "{[\![-]\!]}"] \arrow[d, "{\langle o,t \rangle}"'] & \Omega \arrow[d, "\cong"] \\
2 \times \mathcal{P}_\omega(S)^A \arrow[r, dashed] & 2 \times \mathcal{P}_\omega(\Omega)^A
\end{tikzcd}
$$

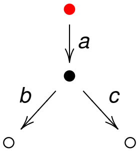
Example I: NDA

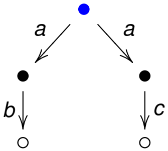


$$\begin{tikzcd}
S \arrow[r, dashed, "{[\![-]\!]}"] \arrow[d, "{\langle o,t\rangle}"'] & \Omega \arrow[d, "\cong"] \\
2 \times \mathcal{P}_\omega(S)^A \arrow[r, dashed] & 2 \times \mathcal{P}_\omega(\Omega)^A
\end{tikzcd}$$

- $[\![\bullet]\!] \neq [\![\bullet]\!]$ (branching structure).
- However... $L(\bullet) = \{ab, ac\} = L(\bullet)$.

Example I: NDA





$$\begin{tikzcd} S \arrow[r, dashed, "[\![-]\!]"] \arrow[d, "\langle o,t \rangle"'] & \Omega \arrow[d, "\cong"] \\ 2 \times \mathcal{P}_\omega(S)^A \arrow[r, dashed] & 2 \times \mathcal{P}_\omega(\Omega)^A \end{tikzcd}$$

- $[\![\bullet]\!] \neq [\![\bullet]\!]$ (branching structure).
- However... $L(\bullet) = \{ab, ac\} = L(\bullet)$.

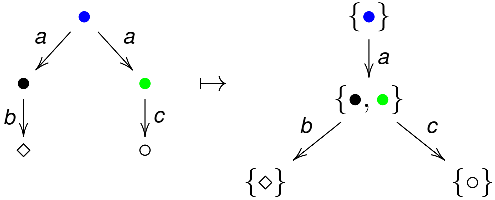
How to model language equivalence coalgebraically?

Classic construction

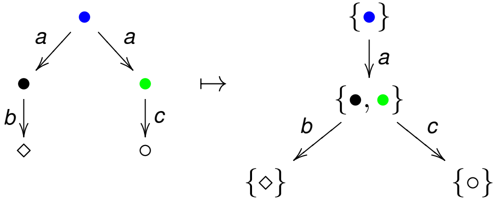
Turn the non-deterministic automaton into a deterministic one via the *powerset construction* and then apply usual semantics.

Classic construction

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



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



$$S \to 2 \times \mathcal{P}(S)^A \qquad\qquad \mathcal{P}(S) \to 2 \times \mathcal{P}(S)^A$$

$$Q \to 2 \times Q^A$$

bisimilarity — language equivalence

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)

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

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

Example I: Determinizing (coalgebraically)

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

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

Example I: Determinizing (coalgebraically)

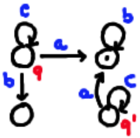
$$\begin{tikzcd}
S \arrow[r, "\{\cdot\}"] \arrow[d, "\langle o,t\rangle"'] & \mathcal{P}_\omega(S) \arrow[r, dashed, "{[\![\cdot]\!]}"] \arrow[dl, "\langle \bar{o},\bar{t}\rangle"] & 2^{A^*} \arrow[d, "\cong"] \\
2 \times \mathcal{P}_\omega(S)^A \arrow[rr, dashed] & & 2 \times (2^{A^*})^A
\end{tikzcd}$$

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

How do we study NDA wrt language equivalence?

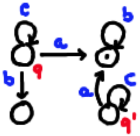
$$L_s = [\![\{s\}]\!]$$

Example II: partial automata

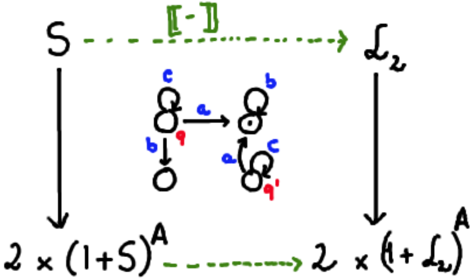


Example II: partial automata

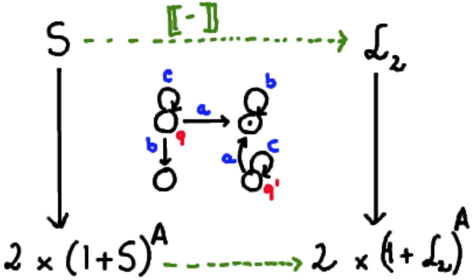
$$\begin{array}{c} S \\ \downarrow \\ 2 \times (1+S)^A \end{array}$$



Example II: partial automata



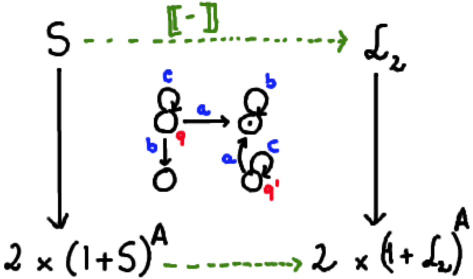
Example II: partial automata



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

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

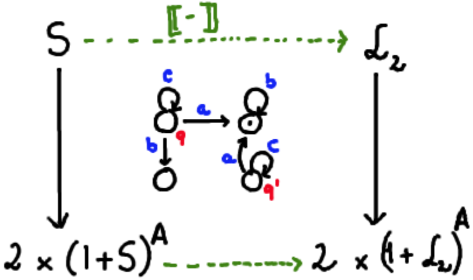
Example II: partial automata



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

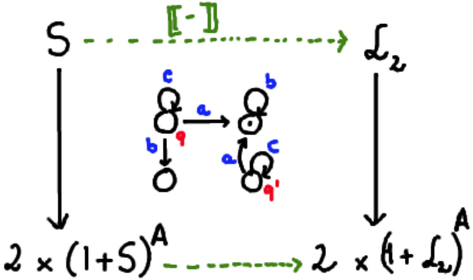


$\mathcal{L}_2$ are pairs of languages $\langle V, W \rangle$ (<accepted words, domain>)

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

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

Example II: partial automata



$\mathcal{L}_2$ 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)

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

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

$$
\begin{tikzcd}
S \arrow[r, "i"] \arrow[d, "\langle o,t \rangle"] & 1+S \arrow[r, dashed, "{[\![-]\!]}"] \arrow[dl, "\langle \bar{o}, \bar{t} \rangle"] & 2^{A^*} \arrow[d, "\cong"] \\
2 \times (1+S)^A \arrow[rr, dashed] & & 2 \times (2^{A^*})^A
\end{tikzcd}
$$

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

Example II: Totalizing (coalgebraically)

$$\begin{tikzcd}
S \arrow[r, "i"] \arrow[d, "\langle o,t\rangle"] & 1+S \arrow[r, dashed, "[\![-]\!]"] \arrow[dl, "\langle \bar{o},\bar{t}\rangle"] & 2^{A^*} \arrow[d, "\cong"] \\
2\times(1+S)^A \arrow[rr, dashed] & & 2\times(2^{A^*})^A
\end{tikzcd}$$

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

How do we study PA wrt language equivalence?

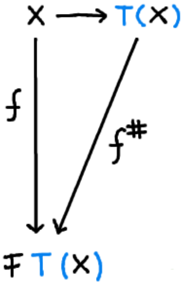
$$L_s = [\![\, i(s) \,]\!]$$

Chasing the pattern. . .

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

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

Chasing the pattern. . .



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

Chasing the pattern...

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

The state space is now *structured* : 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 (intuitively: the structure to hide);
- A final coalgebra for F (for instance, take F to be bounded);
- An extension $f^\sharp$ of f;

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

Ingredients:

- A monad T (intuitively: the structure to hide);
- 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; $f^\sharp\colon T(X) \to F(T(X))$ is an algebra map in $EM(T)$.

Examples revisited

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

Examples revisited

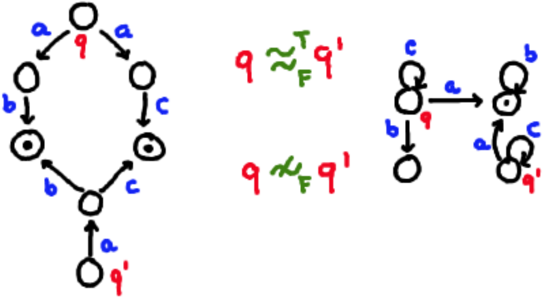
NFA $F(X) = 2 \times X^A$, $T = \mathcal{P}$, $2 \times \mathcal{P}(X)^A$ is a join-semilattice;

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

Examples revisited

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

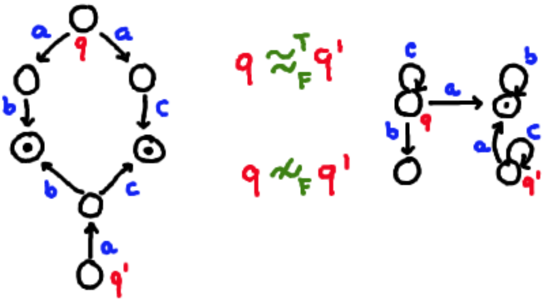
PA $F(X) = 2 \times X^A$, $T = 1 + -$, $2 \times (1 + X)^A$ is a pointed set.



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 $\sim_F$?

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:

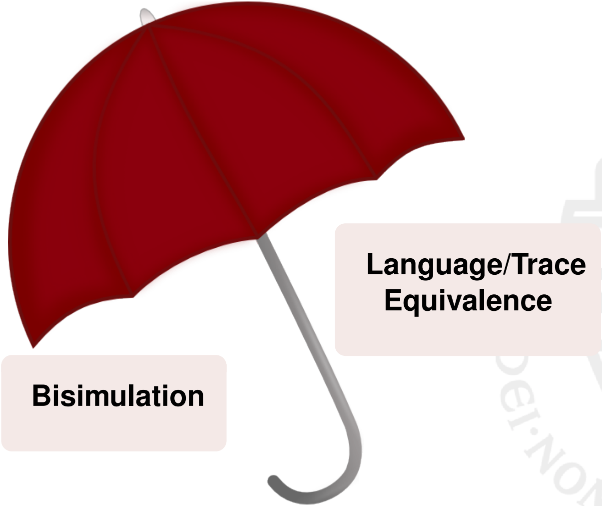
- for NDA ($F(X) = 2 \times X^A$, $T = \mathcal{P}$) that bisimilarity implies language equivalence;
- for PA ($F(X) = 2 \times X^A$, $T = 1 + -$) that equivalences of pair of languages, consisting of defined paths and accepted words, implies equivalence of accepted words;
- for probabilistic automata ($F(X) = [0,1] \times X^A$, $T = \mathcal{D}_\omega$) 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** $S \to \mathbb{R} \times (\mathbb{R}_\omega^X)^A$;
- ...

A. Silva, F. Bonchi, M. Bonsangue and J. Rutten. *Generalizing the powerset construction, coalgebraically*. FSTTCS 2010

And so what?

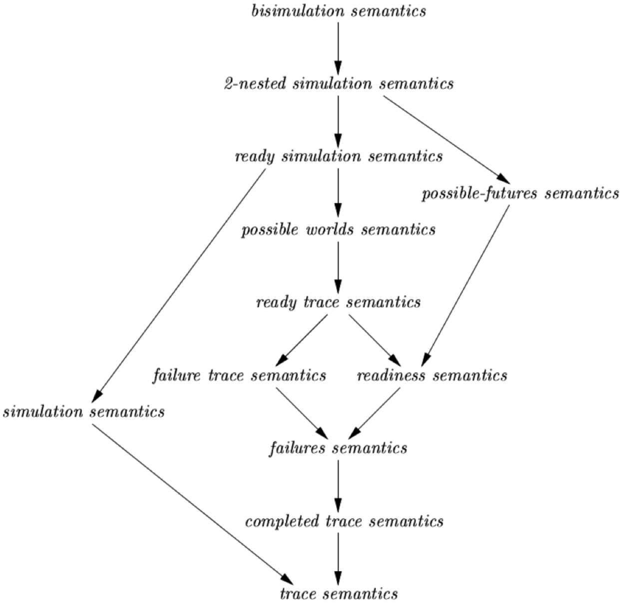


Language/Trace Equivalence

Bisimulation

Is this good enough?

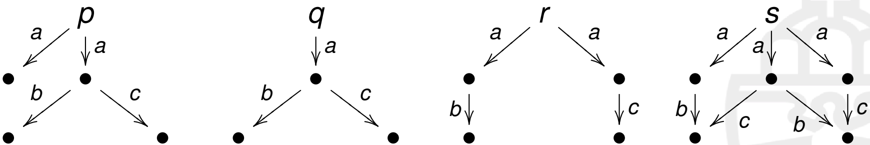
The van Glabbeek spectrum



And now what?

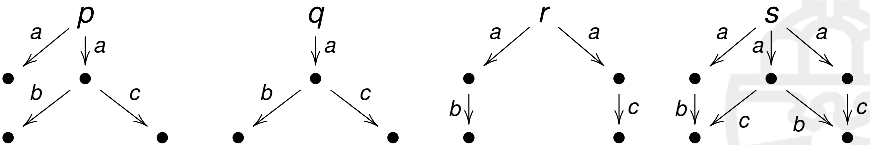
And now what?

Example



- All trace equivalent.

Example



- All trace equivalent.
- *r* and *s* failure equivalent.
- None bisimilar or ready equivalent.

Ready equivalence

$\delta \colon X \to \mathcal{P}(X)^A$ *LTS*

$I(\varphi) = \{a \in A \mid \varphi(a) \neq \emptyset\}$ $\varphi \colon A \to \mathcal{P}(X)$

$I(\delta(x))$ all actions *ready* to be fired in x.

Ready equivalence

$\delta\colon X \to \mathcal{P}(X)^A$ *LTS*

$I(\varphi) = \{a \in A \mid \varphi(a) \neq \emptyset\}$ $\varphi\colon A \to \mathcal{P}(X)$

$I(\delta(x))$ all actions *ready* to be fired in x.

Ready pair of x: $(w, I(\delta(y))) \in A^* \times \mathcal{P}(A)$ s.t. $x \xrightarrow{w} y$.

Ready equivalence

$\delta \colon X \to \mathcal{P}(X)^A$ *LTS*

$I(\varphi) = \{a \in A \mid \varphi(a) \neq \emptyset\}$ $\varphi \colon A \to \mathcal{P}(X)$

$I(\delta(x))$ all actions *ready* to be fired in x.

Ready pair of x: $(w, I(\delta(y))) \in A^* \times \mathcal{P}(A)$ s.t. $x \xrightarrow{w} y$.

x and y are *ready equivalent* $\iff \mathcal{R}(x) = \mathcal{R}(y)$.

Ready equivalence, coalgebraically

The coalgebraic method:

- Semantics: $\mathcal{R}(x) \colon \mathcal{P}(A^* \times \mathcal{P}A)$

Ready equivalence, coalgebraically

The coalgebraic method:

- Semantics: $\mathcal{R}(x) \colon \mathcal{P}(A^* \times \mathcal{P}A)$
- Is $\mathcal{P}(A^* \times \mathcal{P}A)$ the carrier of a final coalgebra?

Ready equivalence, coalgebraically

The coalgebraic method:

- Semantics: $\mathcal{R}(x)\colon \mathcal{P}(A^* \times \mathcal{P}A)$
- Is $\mathcal{P}(A^* \times \mathcal{P}A)$ the carrier of a final coalgebra?
- $\mathcal{P}(A^* \times \mathcal{P}A) \cong \mathcal{P}(\mathcal{P}A)^{A^*}$

The coalgebraic method:

- Semantics: $\mathcal{R}(x)\colon \mathcal{P}(A^* \times \mathcal{P}A)$
- Is $\mathcal{P}(A^* \times \mathcal{P}A)$ the carrier of a final coalgebra?
- $\mathcal{P}(A^* \times \mathcal{P}A) \cong \mathcal{P}(\mathcal{P}A)^{A^*}$

$$\begin{tikzcd}
Q \arrow[r, dashed, "{[\![-]\!]}"] \arrow[d, "t"'] & O^{A^*} \arrow[d, "{\langle \epsilon, (-)_a \rangle}"] \\
O \times Q^A \arrow[r, dashed, "{id \times [\![-]\!]^A}"'] & O \times (O^{A^*})^A
\end{tikzcd}$$

We need a coalgebra $Q \to \mathcal{P}\mathcal{P}(A) \times Q^A$.
But we have $\ldots \delta\colon X \to \mathcal{P}(X)^A$.

Ready equivalence, coalgebraically

$$\delta\colon X \to \mathcal{P}(X)^A \qquad \bar{o}\colon X \to \mathcal{P}(\mathcal{P}(A))$$

Ready equivalence, coalgebraically

$$\delta\colon X \to \mathcal{P}(X)^A \qquad \bar{o}\colon X \to \mathcal{P}(\mathcal{P}(A))$$

$$\overline{o}(x) = \{I(delta(x))\}$$

Ready equivalence, coalgebraically

$$\delta \colon X \to \mathcal{P}(X)^A \qquad \bar{o} \colon X \to \mathcal{P}(\mathcal{P}(A))$$

$$\overline{o}(x) = \{l(delta(x))\}$$

$$\begin{tikzcd}
X \arrow[r, "\{-\}"] \arrow[d, "\langle \overline{o}, \delta \rangle"'] & \mathcal{P}X \arrow[r, dashed, "\llbracket - \rrbracket"] \arrow[dl, "\langle o, t \rangle"] & \mathcal{P}(\mathcal{P}(A))^{A^*} \arrow[d, "\langle \epsilon, (-)_a \rangle"] \\
\mathcal{P}(\mathcal{P}(A)) \times (\mathcal{P}X)^A \arrow[rr, dashed, "id \times \llbracket - \rrbracket^A"'] & & \mathcal{P}(\mathcal{P}(A)) \times (\mathcal{P}(\mathcal{P}(A))^{A^*})^A
\end{tikzcd}$$

$$\llbracket \{x\} \rrbracket = \llbracket \{y\} \rrbracket \iff \mathcal{R}(x) = \mathcal{R}(y)$$

$$\begin{tikzcd}[column sep=large]
X \arrow[r, "\{-\}"] \arrow[d, "\langle \overline{o},\delta\rangle"'] & \mathcal{P}X \arrow[r, dashed, "\llbracket - \rrbracket"] \arrow[dl, "\langle o,t\rangle"] & O^{A^*} \arrow[d, "\langle \epsilon,(-)_a\rangle"] \\
O \times (\mathcal{P}X)^A \arrow[rr, dashed, "id\times\llbracket - \rrbracket^A"'] & & O \times (O^{A^*})^A
\end{tikzcd}$$

- Varying O and $\bar{o}$ yields other equivalences of the spectrum.

The van Glabbeek spectrum coalgebraically

$$\begin{tikzcd}
X \arrow[r, "\{-\}"] \arrow[d, "\langle \bar{o}, \delta \rangle"'] & \mathcal{P}X \arrow[r, dashed, "[\![-]\!]"] \arrow[dl, "\langle o, t \rangle"] & O^{A^*} \arrow[d, "\langle \epsilon, (-)_a \rangle"] \\
O \times (\mathcal{P}X)^A \arrow[rr, dashed, "id \times [\![-]\!]^A"'] & & O \times (O^{A^*})^A
\end{tikzcd}$$

- Varying O and $\bar{o}$ yields other equivalences of the spectrum.
- Even more: must/may semantics.

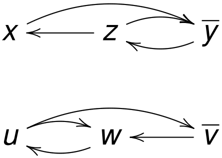
Bonchi, Caltais, Pous, Silva. *Brzozowski's and Up-To Algorithms for Must Testing*. APLAS 2013.

$$\begin{tikzcd}
X \arrow[r, "\{-\}"] \arrow[d, "\langle \overline{o},\delta\rangle"'] & \mathcal{D}X \arrow[r, dashed, "\llbracket - \rrbracket"] \arrow[dl, "\langle o,t\rangle"] & O^{A^*} \arrow[d, "\langle \epsilon,(-)_a\rangle"] \\
O \times (\mathcal{D}X)^A \arrow[rr, dashed, "id\times\llbracket - \rrbracket^A"'] & & O \times (O^{A^*})^A
\end{tikzcd}$$

- LTS's $\delta \colon X \to \mathcal{P}(X)^A$ are replaced by reactive systems $\delta \colon X \to \mathcal{D}(X)^A$.
- All goes through, recovering results by Jou&Smolka 1990.

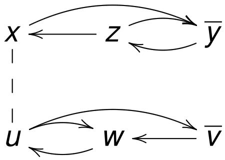
More applications of the powerset construction: up-to

- Often used in process algebra, recent attention in automata theory (Bonchi&Pous POPL 2013).
- Idea: sound enhancements of the proof technique for desired equivalence.
- Simple example: DFA



More applications of the powerset construction: up-to

Use Hopcroft and Karp *on the fly*, through the powerset construction:



Use Hopcroft and Karp *on the fly*, through the powerset construction:

$$
\begin{tikzcd}
x \arrow[rr, bend left, ""] \arrow[d, dash, dashed] & z \arrow[l] \arrow[r, bend left] & \overline{y} \arrow[l, bend left] \\
u \arrow[r, bend left] \arrow[rr, bend left=40] & w \arrow[l, bend left] & \overline{v} \arrow[l]
\end{tikzcd}
$$

$$
\begin{tikzcd}
x \arrow[d, dash, dashed] \\
u
\end{tikzcd}
$$

More applications of the powerset construction: up-to

Use Hopcroft and Karp *on the fly*, through the powerset construction:

$$\begin{tikzcd}
x \arrow[rr, bend left] \arrow[d, dash, dashed] & z \arrow[l] \arrow[r, bend left] & \overline{y} \arrow[l, bend left] \\
u \arrow[r, bend left] \arrow[rr, bend left] & w \arrow[l, bend left] & \overline{v} \arrow[l]
\end{tikzcd}$$

$$\begin{tikzcd}
x \arrow[r] \arrow[d, dash] & \overline{y} \arrow[d, dash, dashed] \\
u \arrow[r] & \overline{v+w}
\end{tikzcd}$$

More applications of the powerset construction: up-to

Use Hopcroft and Karp *on the fly*, through the powerset construction:

$$\begin{tikzcd}
x \arrow[rr, bend left] \arrow[d, dash, dashed] & z \arrow[l] \arrow[r, bend left] & \overline{y} \arrow[l, bend left] \\
u \arrow[r, bend left] \arrow[rr, bend left] & w \arrow[l, bend left] & \overline{v} \arrow[l]
\end{tikzcd}$$

$$\begin{tikzcd}
x \arrow[r] \arrow[d, dash] & \overline{y} \arrow[r] \arrow[d, dash] & z \arrow[d, dash, dashed] \\
u \arrow[r] & \overline{v+w} \arrow[r] & u+w
\end{tikzcd}$$

More applications of the powerset construction: up-to

Use Hopcroft and Karp *on the fly*, through the powerset construction:

$$\begin{tikzcd}
x \arrow[rr, bend left] \arrow[d, dash, dashed] & z \arrow[l] \arrow[r, bend left] & \overline{y} \arrow[l, bend left] \\
u \arrow[r, bend left] \arrow[rr, bend left] & w \arrow[l, bend left] & \overline{v} \arrow[l]
\end{tikzcd}$$

$$\begin{tikzcd}
x \arrow[r] \arrow[d, dash] & \overline{y} \arrow[r] \arrow[d, dash] & z \arrow[r] \arrow[d, dash] & \overline{x+y} \arrow[d, dash, dashed] \\
u \arrow[r] & \overline{v+w} \arrow[r] & u+w \arrow[r] & \overline{u+v+w}
\end{tikzcd}$$

More applications of the powerset construction: up-to

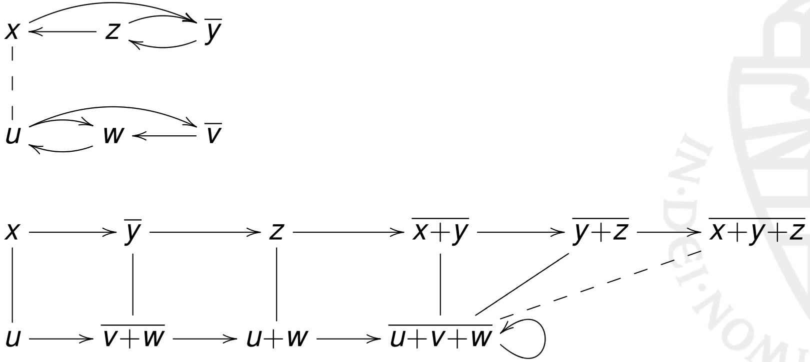
Use Hopcroft and Karp *on the fly*, through the powerset construction:

$$\begin{tikzcd}
x \arrow[rr, bend left] \arrow[dd, dash, dashed] & z \arrow[l] \arrow[r, bend left] & \overline{y} \arrow[l, bend left] \\
\\
u \arrow[r, bend left] \arrow[rr, bend left=40] & w \arrow[l, bend left] & \overline{v} \arrow[l]
\end{tikzcd}$$

$$\begin{tikzcd}
x \arrow[r] \arrow[d, dash] & \overline{y} \arrow[r] \arrow[d, dash] & z \arrow[r] \arrow[d, dash] & \overline{x+y} \arrow[r] \arrow[d, dash] & \overline{y+z} \arrow[dl, dash, dashed] \\
u \arrow[r] & \overline{v+w} \arrow[r] & u+w \arrow[r] & \overline{u+v+w} \arrow[loop right] &
\end{tikzcd}$$

More applications of the powerset construction: up-to

Use Hopcroft and Karp *on the fly*, through the powerset construction:



More applications of the powerset construction: up-to

Use Hopcroft and Karp *on the fly*, through the powerset construction:

$$
\begin{tikzcd}
x \arrow[rr, bend left] \arrow[d, dash, dashed] & z \arrow[l] \arrow[r, bend left] & \overline{y} \arrow[l, bend left] \\
u \arrow[r, bend left] \arrow[rr, bend left=40] & w \arrow[l, bend left] & \overline{v} \arrow[l]
\end{tikzcd}
$$

$$
\begin{tikzcd}
x \arrow[r] \arrow[d, dash] & \overline{y} \arrow[r] \arrow[d, dash] & z \arrow[r] \arrow[d, dash] & \overline{x+y} \arrow[r] \arrow[d, dash] & \overline{y+z} \arrow[r] \arrow[dl, dash] & \overline{x+y+z} \arrow[loop below] \arrow[dll, dash] \arrow[dll, dash, dashed, bend left=5] \\
u \arrow[r] & \overline{v+w} \arrow[r] & u+w \arrow[r] & \overline{u+v+w} \arrow[loop right] & &
\end{tikzcd}
$$

Use Hopcroft and Karp *on the fly*, through the powerset construction:

$$
\begin{tikzcd}
x \arrow[rr, bend left] \arrow[d, dash] & z \arrow[l] \arrow[r, bend left] & \overline{y} \arrow[l, bend left] \\
u \arrow[r, bend left] \arrow[rr, bend left] & w \arrow[l, bend left] & \overline{v} \arrow[l]
\end{tikzcd}
$$

$$
\begin{tikzcd}
x \arrow[r] \arrow[d, dash] & \overline{y} \arrow[r] \arrow[d, dash] & z \arrow[r] \arrow[d, dash] & \overline{x+y} \arrow[r] \arrow[d, dash] & \overline{y+z} \arrow[r] \arrow[dl, dash] & \overline{x+y+z} \arrow[loop below] \arrow[dll, dash] \\
u \arrow[r] & \overline{v+w} \arrow[r] & u+w \arrow[r] & \overline{u+v+w} \arrow[loop right] & &
\end{tikzcd}
$$

More applications of the powerset construction: up-to

One can do better:

$$
\begin{tikzcd}
x \arrow[rr, bend left] \arrow[d, no head] & z \arrow[l] \arrow[r, bend left] & \overline{y} \arrow[l, bend left] \\
u \arrow[r, bend left] \arrow[rr, bend left] & w \arrow[l, bend left] & \overline{v} \arrow[l]
\end{tikzcd}
$$

$$
\begin{tikzcd}
x \arrow[r] \arrow[d, no head] & \overline{y} \arrow[r] \arrow[d, no head] & z \arrow[r] \arrow[d, no head] & \overline{x+y} \arrow[r] \arrow[d, no head, dashed] & \overline{y+z} \arrow[r] & \overline{x+y+z} \arrow[loop below] \\
u \arrow[r] & \overline{v+w} \arrow[r] & u+w \arrow[r] & \overline{u+v+w} \arrow[loop right] & &
\end{tikzcd}
$$

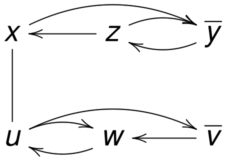
One can do better:

$$\begin{tikzcd}
x \arrow[d, dash] \arrow[rr, bend left, shift left=2] & z \arrow[l] \arrow[r, bend left] & \overline{y} \arrow[l, bend left] \\
u \arrow[r, bend left] \arrow[rr, bend left, shift left=2] & w \arrow[l, bend left] & \overline{v} \arrow[l]
\end{tikzcd}$$

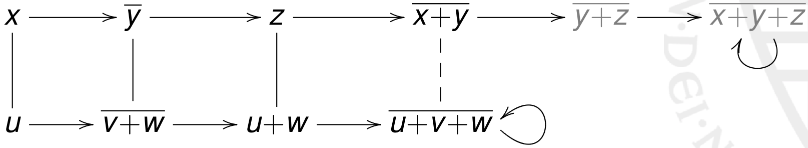
$$\begin{array}{rl}
 & (x,\ u) \\
+ & (y,\ v+w) \\
\hline
= & (x+y,\ u+v+w)
\end{array}$$

$$\begin{tikzcd}
x \arrow[r] \arrow[d, dash] & \overline{y} \arrow[r] \arrow[d, dash] & z \arrow[r] \arrow[d, dash] & \overline{x+y} \arrow[r] \arrow[d, dash, dashed] & \overline{y+z} \arrow[r] & \overline{x+y+z} \arrow[loop below] \\
u \arrow[r] & \overline{v+w} \arrow[r] & u+w \arrow[r] & \overline{u+v+w} \arrow[loop right] & &
\end{tikzcd}$$

One can do better:



$$\begin{array}{rl} & (x,\ u) \\ + & (y,\ v{+}w) \\ \hline = & (x{+}y,\ u{+}v{+}w) \end{array}$$



using bisimulations *up to union*

More applications of the powerset construction: up-to

- up to union – $\mathcal{P}$ monad
- other enhancements: changes in the monad, *determinization* completes a smaller relation into a bisimulation (soundness).

More applications of the powerset construction: axiomatizations

$$\begin{tikzcd}
X \arrow[r, "\{-\}"] \arrow[d, "\langle \overline{o},\delta \rangle"'] & \mathbb{R}^X \arrow[r, dashed, "\llbracket - \rrbracket"] \arrow[dl, "\langle o,t \rangle"] & \mathbb{R}^{A^*} \arrow[d, "\langle \epsilon,(-)_a \rangle"] \\
\mathbb{R} \times (\mathbb{R}^X)^A \arrow[rr, dashed, "id \times \llbracket - \rrbracket^A"'] & & \mathbb{R} \times (\mathbb{R}^{A^*})^A
\end{tikzcd}$$

- First sound and complete axiomatization of weighted language equivalence.
- Usual axioms for regular languages plus vector space (semi-module) axioms.

Bonsangue, Milius, Silva. *Sound and Complete Axiomatizations of Coalgebraic Language Equivalence*. ACM TOCL 2013.

Conclusions

- Lifted *powerset construction* to the more general framework of *FT*-coalgebras;
- Uniform treatment of several types of automata, recovery of known constructions/results;
- Interesting applications in language and concurrency theory;
- Opens the door to the study of *linear equivalences* for many types of automata.

Join for MFPS 2015!

Nijmegen, The Netherlands (Joint with CALCO 2015).