Trace semantics via determinization

Alexandra Silva
joint work with Bart Jacobs and Ana Sokolova

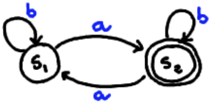
COIN, June 2012

Motivation

- Coalgebras provide an abstract framework to study the behavior of state-based systems.

Motivation

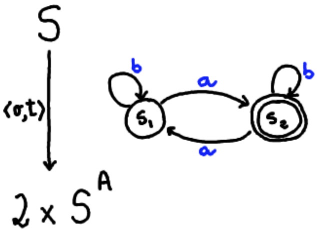
- Coalgebras provide an abstract framework to study the behavior of state-based systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.



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

Motivation

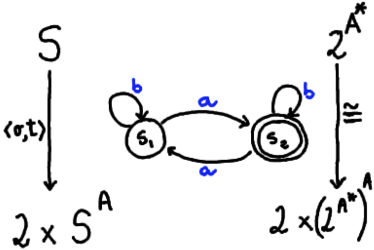
- Coalgebras provide an abstract framework to study the behavior of state-based systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.



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

Motivation

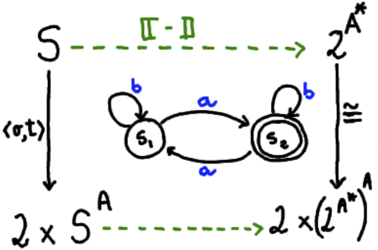
- Coalgebras provide an abstract framework to study the behavior of state-based systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.



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

Motivation

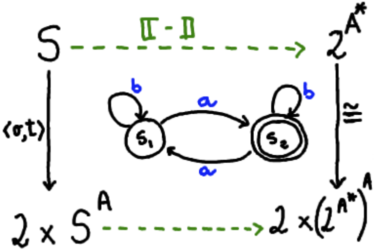
- Coalgebras provide an abstract framework to study the behavior of state-based systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.



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

Motivation

- Coalgebras provide an abstract framework to study the behavior of state-based systems.
- Much of the coalgebraic approach can be nicely illustrated with deterministic automata.



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

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

Motivation

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

- The system type F determines a canonical notion of equivalence: bisimilarity.

Motivation

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

- The system type F determines a canonical notion of equivalence: bisimilarity.

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

Motivation

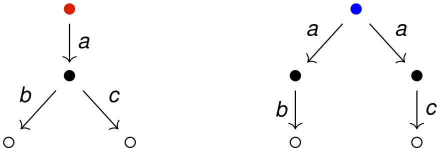
$S \to 2 \times S^A$ $\qquad$ $S \to F(S)$

- The system type F determines a canonical notion of equivalence: bisimilarity.

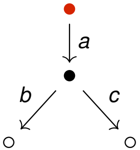
deterministic automata	language equivalence
infinite streams	pointwise equality
labelled transition systems	branching bisimilarity

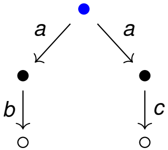
- Sometimes bisimilarity is not what we want…

Example I: NDA



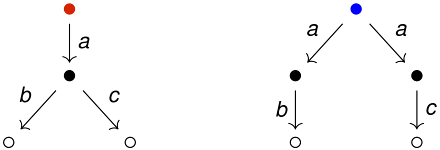
Example I: NDA





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

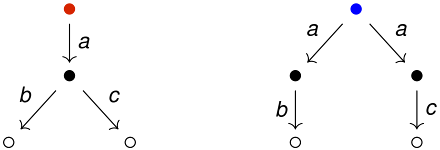
Example I: NDA



- $[\![\bullet]\!] \neq [\![\bullet]\!]$ (branching structure).

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

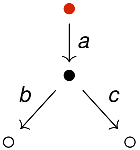
Example I: NDA

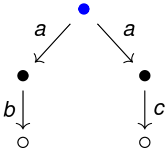


- $[\![\textcolor{red}{\bullet}]\!] \neq [\![\textcolor{blue}{\bullet}]\!]$ (branching structure).
- However... $L(\textcolor{red}{\bullet}) = \{ab, ac\} = L(\textcolor{blue}{\bullet})$.

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

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... $L(\bullet) = \{ab, ac\} = L(\bullet)$.

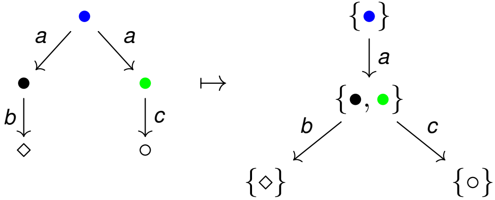
How to model language equivalence coalgebraically?

Trace semantics: take I

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

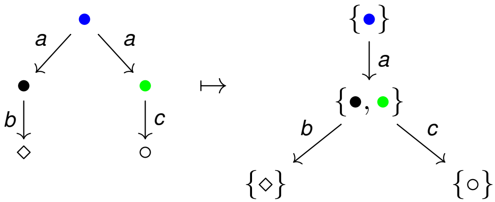
Trace semantics: take I

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



Trace semantics: take I

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



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

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

bisimilarity — language equivalence

Trace semantics: take I, abstractly

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

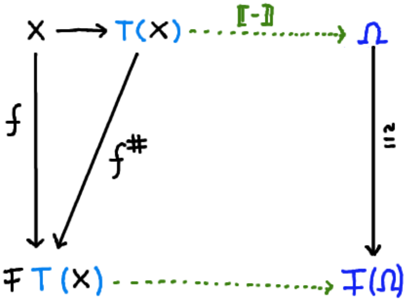
The state space is now *structured* : T monad $(\mathcal{P}, 1+, \ldots)$.

Trace semantics: take I, abstractly

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

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

Trace semantics: take I, abstractly



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

Take I, in a nutshell. . .

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

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;

Take I, in a nutshell. . .

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

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; We can require $FT(X)$ to be a T-algebra; $f^\sharp\colon\ T(X) \to F(T(X))$ is an algebra map in $\mathcal{EM}(T)$.

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

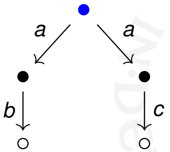
Trace semantics: take II

There is another way of modeling NDA's coalgebraically.

$$S \xrightarrow{\alpha} \mathcal{P}(1 + A \times S)$$

$$s \text{ is final} \iff * \in \alpha(s)$$

$$s \xrightarrow{a} t \iff \langle a, t \rangle \in \alpha(s)$$



Trace semantics: take II, abstractly

$$S \to TFS \quad \text{is an arrow in } \mathcal{K}\ell(T)$$

$$\begin{tikzcd}
S \arrow[r, dashed, "\mathrm{tr}"] \arrow[d] & Z \arrow[d] \\
FS \arrow[r, dashed] & FZ
\end{tikzcd}$$

arrows $X \to Y$ in $\mathcal{K}\ell(T)$ are $X \to TY$ in Set

Trace semantics: take II, abstractly

$S \to TFS$ is an arrow in $\mathcal{K}\ell(T)$

$$\begin{tikzcd} S \arrow[r, dashed, "\mathrm{tr}"] \arrow[d] & Z \arrow[d] \\ FS \arrow[r, dashed] & FZ \end{tikzcd}$$

$$\begin{tikzcd} S \arrow[r, dashed, "\mathrm{tr}"] \arrow[d] & A^* \arrow[d] \\ 1 + A \times S \arrow[r, dashed] & 1 + A \times A^* \end{tikzcd}$$

arrows $X \to Y$ in $\mathcal{K}\ell(T)$ are $X \to TY$ in Set

Trace semantics: take II, abstractly

$$S \to TFS \quad \text{is an arrow in } \mathcal{K}\ell(T)$$

$$\begin{tikzcd} S \arrow[r, dashed, "\mathrm{tr}"] \arrow[d] & Z \arrow[d] \\ FS \arrow[r, dashed] & FZ \end{tikzcd}$$

$$\begin{tikzcd} S \arrow[r, dashed, "\mathrm{tr}"] \arrow[d] & A^* \arrow[d] \\ 1 + A \times S \arrow[r, dashed] & 1 + A \times A^* \end{tikzcd}$$

$$\mathrm{tr}\colon S \to \mathcal{P}(A^*)$$

arrows $X \to Y$ in $\mathcal{K}\ell(T)$ are $X \to TY$ in Set

Trace semantics: take II, abstractly

$S \to \color{blue}{T}\color{red}{F}S$ is an arrow in $\mathcal{K}\ell(\color{blue}{T})$

$$\begin{tikzcd} S \arrow[r, dashed, "\mathrm{tr}"] \arrow[d] & Z \arrow[d] \\ FS \arrow[r, dashed] & FZ \end{tikzcd}$$

$$\begin{tikzcd} S \arrow[r, dashed, "\mathrm{tr}"] \arrow[d] & A^* \arrow[d] \\ 1 + A \times S \arrow[r, dashed] & 1 + A \times A^* \end{tikzcd}$$

$$\begin{tikzcd}[column sep=small] & \color{blue}{\bullet} \arrow[dl, "a"'] \arrow[dr, "a"] & \\ \bullet \arrow[d, "b"'] & & \bullet \arrow[d, "c"] \\ \circ & & \circ \end{tikzcd}$$

$\mathrm{tr}\colon S \to \color{blue}{\mathcal{P}}(A^*)$ $\mathrm{tr}(\color{blue}{\bullet}) = \{ab, ac\}$

arrows $X \to Y$ in $\mathcal{K}\ell(T)$ are $X \to TY$ in Set

Take II, in a nutshell. . .

- Systems are modeled as $S \to TFS$
- Semantics in $\mathcal{K}\ell(T)$
- The catch: for the semantic map tr to exist, we need non-trivial side-conditions (like enrichement in dcpo's), ruling out some interesting examples.

I. Hasuo, B. Jacobs and A. Sokolova. *Generic Trace Semantics via Coinduction.* LMCS 2007.

Our goals

- Understand the connections between the two approaches to trace semantics
- Can we cover more examples in the $\mathcal{K}\ell$ setting?

Observation I

Why do both approaches work for NDA's?

$$X \to 2 \times \mathcal{P}(S)^A \qquad\qquad X \to \mathcal{P}(1 + A \times X)$$

Observation I

Why do both approaches work for NDA's?

$$X \to 2 \times \mathcal{P}(S)^A \qquad\qquad X \to \mathcal{P}(1 + A \times X)$$

Well, they are the same:

$$\mathcal{P}(1 + A \times X) \;\cong\; \mathcal{P}(1) \times \mathcal{P}(A \times X) \;\cong\; 2 \times (\mathcal{P}X)^A.$$

Observation I

Why do both approaches work for NDA's?

$$X \to 2\times\mathcal{P}(S)^A \qquad\qquad X \to \mathcal{P}(1+A\times X)$$

Well, they are the same:

$$\mathcal{P}(1+A\times X) \;\cong\; \mathcal{P}(1)\times\mathcal{P}(A\times X) \;\cong\; 2\times(\mathcal{P}X)^A.$$

And the semantics coincide: $\mathrm{tr}(x) = [\![\, \{x\} \,]\!]$

Yet another example: generative systems

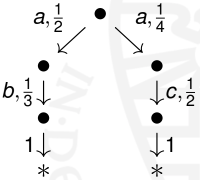
$$(S, \alpha\colon\ X \to \mathcal{D}_\omega(1 + A \times X)\ )$$

$$x \xrightarrow{p} * \quad \text{if} \quad \alpha(x)(*) = p,$$

i.e., x successfully terminates with probability p, and

$$x \xrightarrow{a,p} y \quad \text{if} \quad \alpha(x)(a, y) = p,$$

i.e., if x can make an a-labelled step to y with weight p.



Yet another example: generative systems

$(S, \alpha\colon\ X \to \mathcal{D}_\omega(1 + A \times X)\ )$

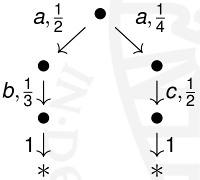
$$x \xrightarrow{p} * \quad \text{if} \quad \alpha(x)(*) = p,$$

i.e., x successfully terminates with probability p, and

$$x \xrightarrow{a,p} y \quad \text{if} \quad \alpha(x)(a, y) = p,$$

i.e., if x can make an a-labelled step to y with weight p.

Even though $X \to TFX$ no trace semantics.



And now what?

Well, for NDA we had

$$\mathcal{P}(1 + A \times X) \;\cong\; \mathcal{P}(1) \times \mathcal{P}(A \times X) \;\cong\; 2 \times (\mathcal{P}X)^A.$$

And now what?

Well, for NDA we had

$$\mathcal{P}(1 + A \times X) \cong \mathcal{P}(1) \times \mathcal{P}(A \times X) \cong 2 \times (\mathcal{P}X)^A.$$

$$\mathcal{D}(1 + A \times X) \cong \mathcal{D}(1) \times \mathcal{D}(A \times X) \cong [0,1] \times (\mathcal{D}X)^A.$$

And now what?

Well, for NDA we had

$$\mathcal{P}(1 + A \times X) \;\cong\; \mathcal{P}(1) \times \mathcal{P}(A \times X) \;\cong\; 2 \times (\mathcal{P}X)^A.$$

$$\mathcal{D}(1 + A \times X) \;\cong\; \mathcal{D}(1) \times \mathcal{D}(A \times X) \;\hookrightarrow\; [0,1] \times (\mathcal{D}X)^A.$$

And now what?

Well, for NDA we had

$$\mathcal{P}(1 + A \times X) \cong \mathcal{P}(1) \times \mathcal{P}(A \times X) \cong 2 \times (\mathcal{P}X)^A.$$

$$\mathcal{D}(1 + A \times X) \cong \mathcal{D}(1) \times \mathcal{D}(A \times X) \hookrightarrow [0,1] \times (\mathcal{D}X)^A.$$

Trace for generative systems

$$\begin{tikzcd}
X \arrow[r, "\eta"] \arrow[d, "\alpha"'] & \mathcal{D}(X) \arrow[r, dashed, "{[\![-]\!]}"] \arrow[ddl, "(\delta\circ\alpha)^\sharp"] & {[0,1]^{A^*}} \arrow[dd] \\
\mathcal{D}(1 + A \times X) \arrow[d, "\delta"'] & & \\
{[0,1] \times (\mathcal{D}X)^A} \arrow[rr, dashed] & & {[0,1] \times ([0,1]^{A^*})^A}
\end{tikzcd}$$

Each state x is assigned a weighted language $L\colon A^* \to [0,1]$

Is there a general result . . .

. . . or was this all ad-hoc?

Is there a general result . . .

. . . or was this all ad-hoc?

Trace semantics for TF-coalgebras via determinization

Given $c\colon X \to T(FX)$ and a nat. transf. $\delta\colon TF \to GT$, one obtains trace semantics in three steps:

1. Transform c into $c^\sharp$;
2. Obtain a map $TX \to Z$ by finality;
3. Get $\mathrm{tr}\colon X \to Z$ by precomposition with η.

$$\begin{tikzcd}
X \arrow[r, "\eta"] \arrow[d, "c"'] \arrow[rr, bend left, "\mathrm{tr}"] & TX \arrow[r, dashed, "[\![-]\!]"] \arrow[ddl, "(\delta\circ c)^\sharp"] & Z \arrow[dd] \\
TFX \arrow[d, "\delta"'] & & \\
GTX \arrow[rr, dashed] & & GZ
\end{tikzcd}$$

Coincidence of semantics

- This method provides trace semantics for TF coalgebras even when the $\mathcal{K}\ell$ semantics does not work.
- But what happens when both methods work?

In $\mathcal{K}\ell(T)$:

$$\begin{tikzcd}
X \arrow[r, dashed, "\mathrm{tr}_{\mathcal{K}\ell}"] \arrow[d, "c"'] & W \arrow[d, "\cong"] \\
FX \arrow[r, dashed] & F(W)
\end{tikzcd}$$

Via the extension:

$$\begin{tikzcd}
X \arrow[r, "\eta"] \arrow[d, "c"'] \arrow[rr, bend left, "\mathrm{tr}"] & TX \arrow[r, dashed, "\llbracket - \rrbracket"] \arrow[ddl, "(\delta \circ c)^{\sharp}"] & Z \arrow[dd] \\
TFX \arrow[d, "\delta"'] & & \\
GTX \arrow[rr, dashed] & & GZ
\end{tikzcd}$$

Coincidence of semantics

Proposition

$$\begin{tikzcd}
X \arrow[r, "\eta"'] \arrow[rr, bend left, "\mathrm{tr}_{\mathcal{K}\ell}"] \arrow[rrr, bend left, "\mathrm{tr}"] & TX \arrow[r] \arrow[d, "(\delta\circ c)^\sharp"] & TW \arrow[r, dashed] \arrow[d, "\cong"] & Z \arrow[d, "\zeta"', "\cong"] \\
& G(TX) \arrow[r] & G(TW) \arrow[r, dashed] & G(Z)
\end{tikzcd}$$

The extension semantics map tr decomposes via the Kleisli trace map $\mathrm{tr}_{\mathcal{K}\ell}$: semantics are compatible.

What I did not show

- We have in the paper a more formal connection between both constructions using liftings.
- We have several examples: different types of probabilistic systems and quantum walks.

Conclusions and Future Work

Conclusions

- Extension of Kleisli trace semantics via determinizations.
- Formal connection between both semantics.

Conclusions and Future Work

Conclusions

- Extension of Kleisli trace semantics via determinizations.
- Formal connection between both semantics.

Future Work

- Coalgebraic account of LTL and CTL.
- General axiomatizations of trace semantics (connections with [Bonsangue,Milius,Silva'12])

Conclusions and Future Work

Conclusions

- Extension of Kleisli trace semantics via determinizations.
- Formal connection between both semantics.

Future Work

- Coalgebraic account of LTL and CTL.
- General axiomatizations of trace semantics (connections with [Bonsangue,Milius,Silva'12])

Thanks!