Stream automata are coalgebras

Vincenzo Ciancia

Joint work with *Yde Venema*

Institute for Logic, Language and Computation
University of Amsterdam

Tallin - March 31, 2012

Motivation

Deterministic finite automata are a prime example of coalgebras:

$$(X, f : X \to X^C \times 2)$$

Standard definitions for bisimilarity, partition refinement and universal model (three views of the same problem!).

Coalgebraic bisimilarity in DFAs obviously corresponds to language equivalence (look at the derivatives!).

[Rutten, CONCUR 98] the equivalence classes of the Myhill-Nerode theorem precisely correspond to the elements of the final coalgebra.

Stream automata

Stream automata, e.g. {Büchi, Muller, parity}-automata, accept languages of infinite words (ω-regular languges).

Closure properties and decision procedures are well-established.

However, procedures and proofs are complex and quite ad-hoc, and there is **no universal model**.

Question:

Can we describe ω-regular languages as elements of the final coalgebra for some functor?

or

Can we describe stream automata as coalgebras, in such a way that bisimilarity coincides with language equivalence?

In this work:

Automata over *infinite words* are coalgebras.

A two-sorted setting is required (just like in Wilke algebras, that provide an algebraic approach).

Closure properties are easy.

There is an universal model.

Canonical representatives of language equivalence classes, minimization, decidability of language equivalence

for free!

Part I

Stream automata are coalgebras

Deterministic Muller automata

Def: A DMA is a structure $(Q, \delta : Q \to Q^C, \mathcal{M} \subseteq \mathcal{P}(Q))$.

where: Q finite set of states, δ deterministic transition function, $\mathcal{M}$ set of sets of states (Muller sets).

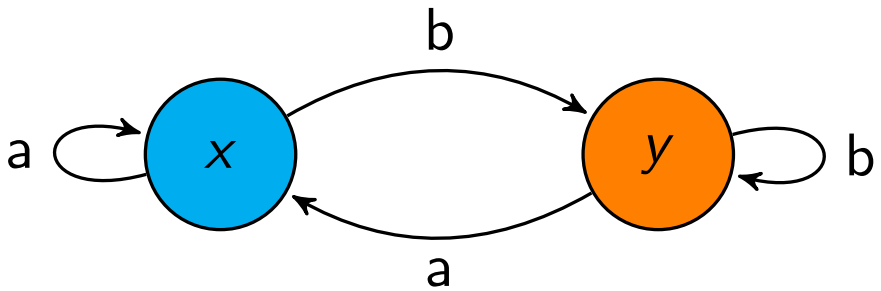
An infinite word (stream) α is in $\mathcal{L}(q)$ iff.

$$Inf(x, q) \in \mathcal{M}$$

(the set of states traversed infinitely often is a Muller set)

Example:

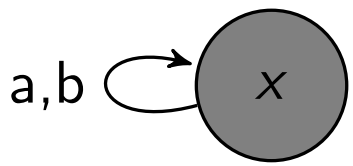
x and y accept the same language:



$$\mathcal{M} = \{\{x\}, \{y\}\} \qquad \mathcal{L}(x) = \mathcal{L}(y) = (C^*)(a^\omega \cup b^\omega)$$

No minimal model:

x and y can not be collapsed: the only possible system with one state is the following and it accepts a different language.



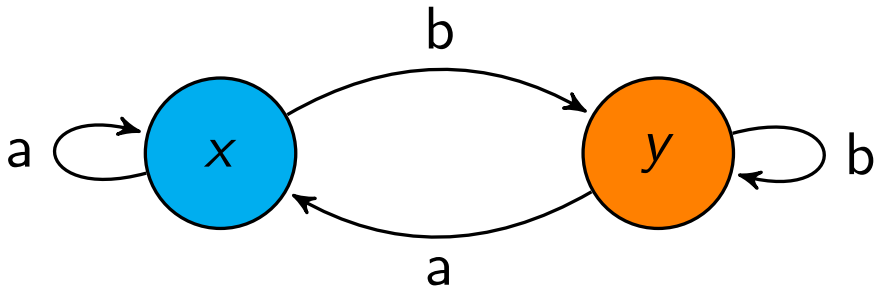
$$\mathcal{M} = \{x\} \qquad \mathcal{L}(x) = C^{\omega}$$

Lasso and looping languages

For $\mathcal{L}$ a set of streams, define:

- $Loop(\mathcal{L}) = \{u \in C^+ | u^\omega \in \mathcal{L}\}$
- $Lasso(\mathcal{L}) = \{(s, l) \in C^* \times C^+ | s\, l^\omega \in \mathcal{L}\}$

Example:



$$Lasso(x) = Lasso(y) = \{(s, l) | s \in C^*, l \in a^* \cup b^*\}$$

$$Loop(x) = Loop(y) = a^+ \cup b^+$$

$\mathcal{L}(x)$ obviously determines $Loop(x)$ and $Lasso(x)$.

$Lasso(\mathcal{L})$ determines $\mathcal{L}$

Fact: If L and L' are ω-regular, $Lasso(L) = Lasso(L') \Rightarrow L = L'$.

If two ω-regular languages are different, there is a distinguishing lasso!

DMAs are coalgebras - the easy way

Coalgebras of the Set functor $\mathtt{T}(X) = X^C \times S$, where $S = \mathcal{P}(C^+)$

deterministic labelled transition systems with observations on states

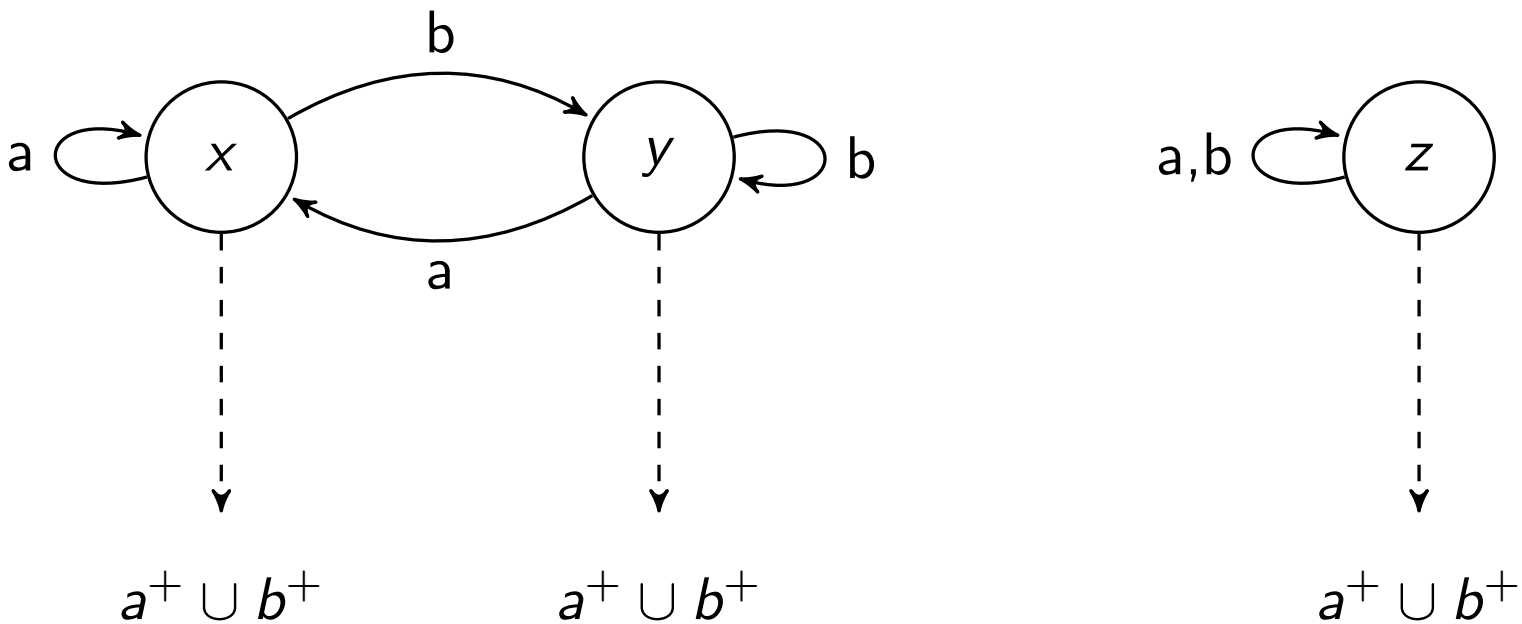
The DMA $(Q, \delta, \mathcal{M})$ is mapped to the coalgebra (Q, f) where

$$f : Q \to Q^C \times \mathcal{P}(C^+)$$

$$f(q) = (\delta(q), Loop(q))$$

Bisimilarity is language equivalence

We are now allowed to collapse x and y:



This looks quite simple!

Characterise *precisely* the ω-regular languages.

(or: don't worry, we can make it more complicated!)

Part II

Finite, circular and coherent coalgebras

Ingredients:

- $Loop(x)$ is regular (and of non-empty words).
- Regular languages of non-empty words are coalgebras.
- Two-sorted coalgebras and dependent bisimilarity.

$Loop(x)$ is regular

Proposition

The language $Loop(x) \subseteq C^+$ is regular.

It is accepted by a parallel automaton that consumes symbols in every state simultaneously, accumulating the set of traversed states.

The accepting states are defined from the Muller sets $\mathcal{M}$ (see the paper for the details of the construction).

Regular languages of non-empty words

DFAs are coalgebras of the functor $\mathtt{T}(X) = X^C \times 2$. But $Loop(x)$ is a language of **non-empty words!**.

Define the functor $\mathtt{D}(X) = (X \times 2)^C$.

Proposition

Finite $\mathtt{D}$*-coalgebras are in one-to-one correspondence with DFAs accepting non-empty words.*

In the final D-coalgebra, all the regular languages of non-empty words are represented as points whose generated sub-coalgebra is finite.

The "derivative" operation computes the standard derivative, minus the empty word.

"Dependent" coalgebras

Notice that:

- We can characterise equality of languages such as $Loop(x)$ using morphisms of $\mathtt{D}$-coalgebras.
- We just saw how to represent DMAs as coalgebras whose bisimilarity *depends* upon equality of $Loop(x)$ for each x.

We define a kind of two-sorted coalgebras that combine these two features.

Two-sorted sets

$\mathtt{Set}^2$ is the category of functors from the discrete category 2 (having two objects and just identity arrows) into $\mathtt{Set}$.

$\mathtt{Set}^2$ is the category of pairs of sets

$$X = (X_1, X_2)$$

and pairs of functions

$$f : X \to Y = (f_1 : X_1 \to Y_1, f_2 : X_2 \to Y_2).$$

Two-sorted sets and "sort-wise" functions!

The functor Ω

Consider the functor $\Omega : \mathtt{Set}^2 \to \mathtt{Set}^2$

$$\Omega(X) = (X_1^C \times X_2, \mathtt{D}(X_2))$$

which is the same as

$$\Omega(X) = (X_1^C \times X_2, (X_2 \times 2)^C)$$

$$h : X \to Y \Rightarrow \Omega h = (h_1^C \times h_2, (h_2 \times 2)^C)$$

Ω-coalgebras

An Ω-coalgebra $f : X \to \Omega(X)$ is a pair $f = (f_1, f_2)$ with

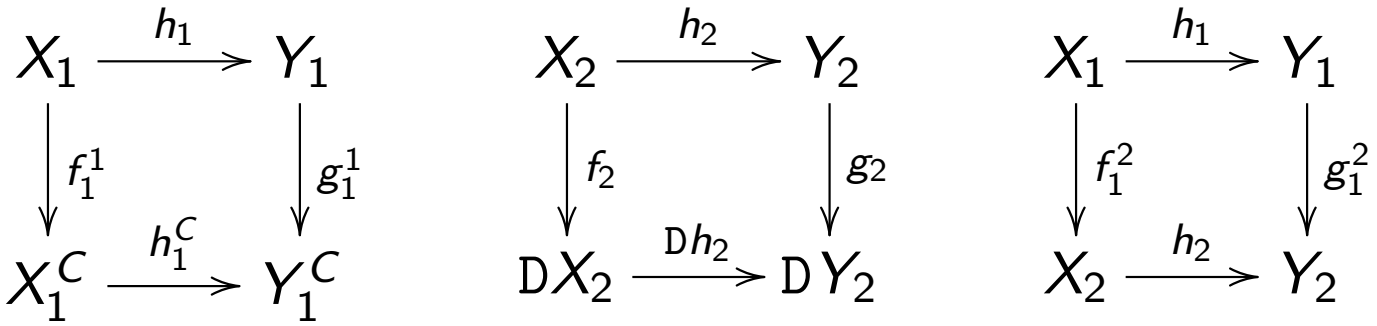
$$f_1 : X_1 \to X_1^C \times X_2 \qquad f_2 : X_2 \to (X_2 \times 2)^C$$

$$f_1 = \langle f_1^1 : X_1 \to X_1^C, f_1^2 : X_1 \to X_2 \rangle$$

f_1^1 is a $(-)^C$ coalgebra $\qquad$ f_2 is a $\mathtt{D}$-coalgebra

f_1^2 maps each state in X_1 to a state in X_2

An arrow $h : X \to Y$ in $\mathtt{Set}^2$ is a coalgebra morphism from (X, f) to (Y, g) iff. the following diagram commutes



h_1 is a morphism of $(-)^C$-coalgebras

h_2 is a morphism of $\mathtt{D}$-coalgebras

the maps from X_1 to X_2 and Y_1 to Y_2 commute with h_1 and h_2

Dependent bisimilarity

$x, y \in X_1$ are identified by a morphism, or "bisimilar" iff.:

1. $\mathcal{L}_{\mathtt{D}}(f_1^2(x)) = \mathcal{L}_{\mathtt{D}}(f_1^2(y))$ (these are DFAs of non-empty words!);

2. For all $c \in C$, $f_1^1 xc$ and $f_1^1 yc$ are bisimilar in turn.

From DMAs to Ω-coalgebras

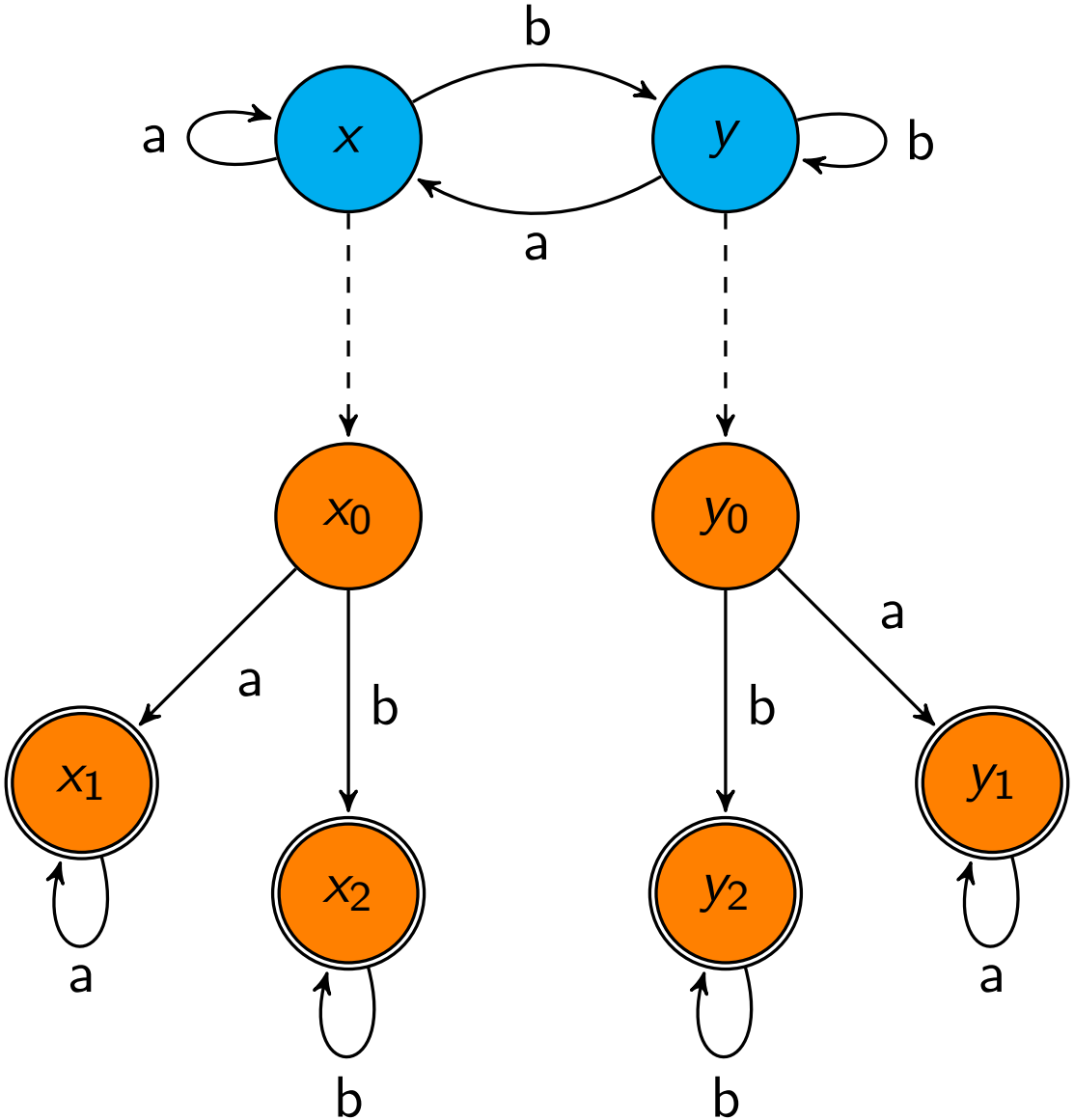
A DMA $(Q, \delta, \mathcal{M})$ is mapped to an Ω-coalgebra (X, f)

$$X_1 = Q \qquad f_1 = \langle \delta, \lambda x.\text{initial state of } \mathit{Loop}(x) \rangle$$

(X, f_2) is the disjoint union of the D-coalgebras for $\mathit{Loop}(x \in Q)$

Theorem

$x, y \in Q$ are bisimilar in the coalgebraic sense if and only if they accept the same stream language.


b
a
x
y
b
a
x0
y0
a
a
b
b
x1
x2
y2
y1
a
b
b
a

Circularity and coherence

A coalgebra accpets lassos (s, l) by following s in the first sort, and then accepting l in the second.

Not all coalgebras accept $Lasso(\mathcal{L})$ for some ω-regular $\mathcal{L}$

the "good coalgebras" are invariant under

$$(s, l) \equiv (s', l') \iff s\, l^{\omega} = s'\, (l')^{\omega}$$

We single out two properties that guarantee the above: circularity and coherence.

Circular:

$$\forall x \in X_1.\forall k > 0.\forall u \in C^+.u \in Loop(x) \iff u^k \in Loop(x)$$

Coherent:

$$\forall x \in X_1.Loop(x) = \{cu \mid uc \in Loop(f_1^1(x)(c))\}$$

Theorem

Each finite, circular and coherent Ω-coalgebras is the image of some DMA.

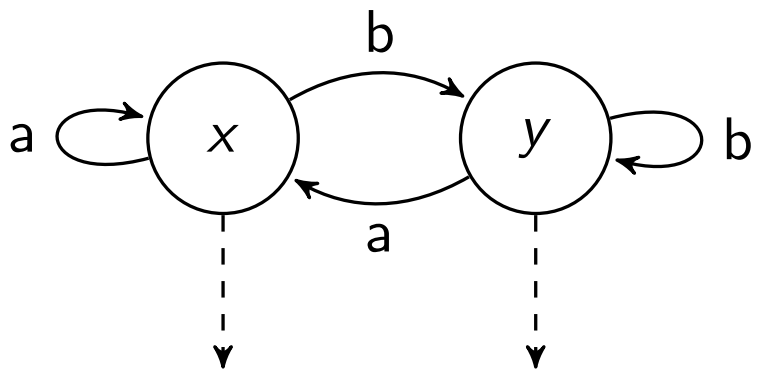
Proposition

Circularity and coherence are preserved by morphisms.

Corollary

The points in the final coalgebra that generate a finite, circular and coherent are in one-to-one correspondence with the ω-regular languages.

This coalgebra is neither circular nor coherent.



$(\epsilon, a) \in Lasso(x)$ $\quad$ $(\epsilon, aaa) \notin Lasso(x)$

$(a, bb) \in Lasso(y)$ $\quad$ $(ab, b) \notin Lasso(y)$

So what?

Boolean operations

Recall union, intersection and complementation of DFAs.

Construct a product automaton (unary product for the complement) and define accepting states using the boolean operation in question.

A very similar definition works for Ω-coalgebras. Proposition: the class of circular and coherent coalgebras is closed under boolean operations.

Simple proof of closure of ω-regular languages under boolean operations.

Minimal models

Coalgebraic partition refinement uses the terminal sequence to compute the image in the final coalgebra of the model.

The final Ω-coalgebra exists; it's the set of languages of lassos over the alphabet. So we have minimal models (not present in DMAs)

On finite-state systems, partition refinement terminates in finite steps.

Simple proof of decidability of language equivalence.

So what?

Future work

Remarkably simple: Ehrenfeucht-Parikh-Rozenberg block cancellation property for lasso languages.

Doable: Myhill-Nerode theorem for lasso languages (we get a dependently typed congruence). Deriving the ω-regular language from the equivalence classes.

- - - - - - - - - - - - up to here: consequence of the framework - - - - - - - - - - - -

Difficult: Coinductive definition of the ω-regular language!

More topics: Modal fixpoint logics on streams; nominal ω-regular languages

The end.