Behavioural differential equations: an introduction

Joost Winter

Centrum Wiskunde & Informatica, Leiden University
Part the NWO project 'CoRE' (Coinductive Calculi of Regular Expressions)

ICT Open, October 23, 2012

Overview

This talk will be about:

1. Finite, nondeterministic, stream and weighted automata (seen as coalgebras)

Overview

This talk will be about:

1. Finite, nondeterministic, stream and weighted automata (seen as coalgebras)
2. Presenting work developed by Brzozowski, Rutten, Bonsangue, Silva, and many others...

Overview

This talk will be about:

1. Finite, nondeterministic, stream and weighted automata (seen as coalgebras)
2. Presenting work developed by Brzozowski, Rutten, Bonsangue, Silva, and many others...
3. ...and if time allows even some of my own research.

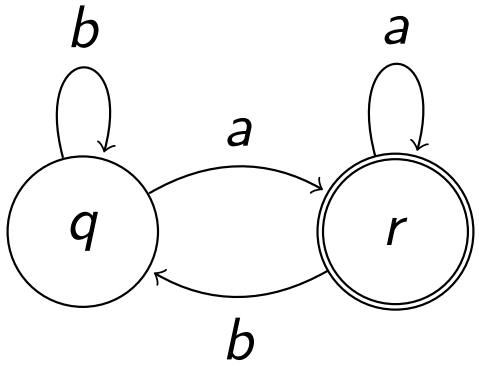
Overview

This talk will be about:

1. Finite, nondeterministic, stream and weighted automata (seen as coalgebras)
2. Presenting work developed by Brzozowski, Rutten, Bonsangue, Silva, and many others...
3. ...and if time allows even some of my own research.
4. (Almost) no category theory.

A fresh look at automata

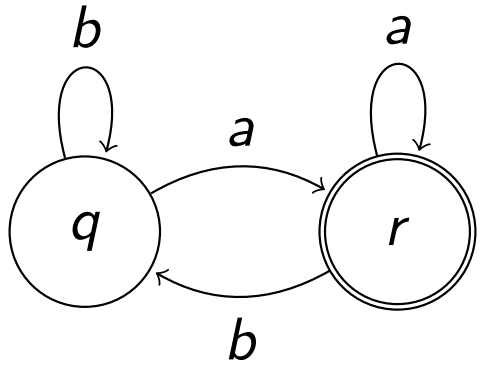
An *automaton* is a triple (Q, o, δ)



$$Q = \{q, r\}$$

A fresh look at automata

An *automaton* is a triple (Q, o, δ)



$$Q = \{q, r\}$$

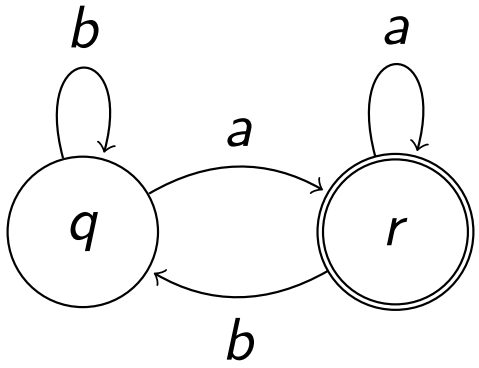
We represent o and δ with a system of *behavioural differential equations*:

$$\begin{array}{lllllllll} o(q) & = & 0 & \quad q_a & = & r & \quad q_b & = & q \\ o(r) & = & 1 & \quad r_a & = & r & \quad r_b & = & q \end{array}$$

Derivatives...

- q_a: shorthand for $\delta(q, a)$.
- Called the *a-derivative* of q...
- ...has some properties similar to the familiar derivative from calculus!

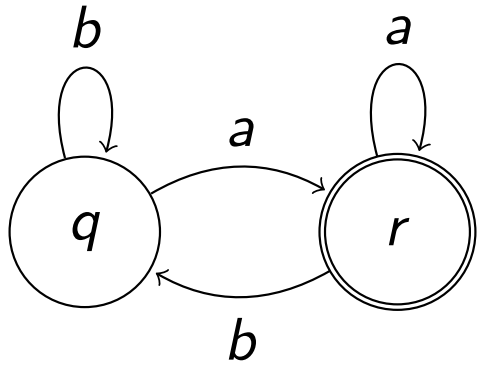
Word derivatives



We can extend the notion of derivatives from alphabet symbols to words inductively (1 denotes the empty word):

$$\begin{aligned} q_1 &= q \\ q_{a \cdot w} &= (q_a)_w \end{aligned}$$

Word derivatives



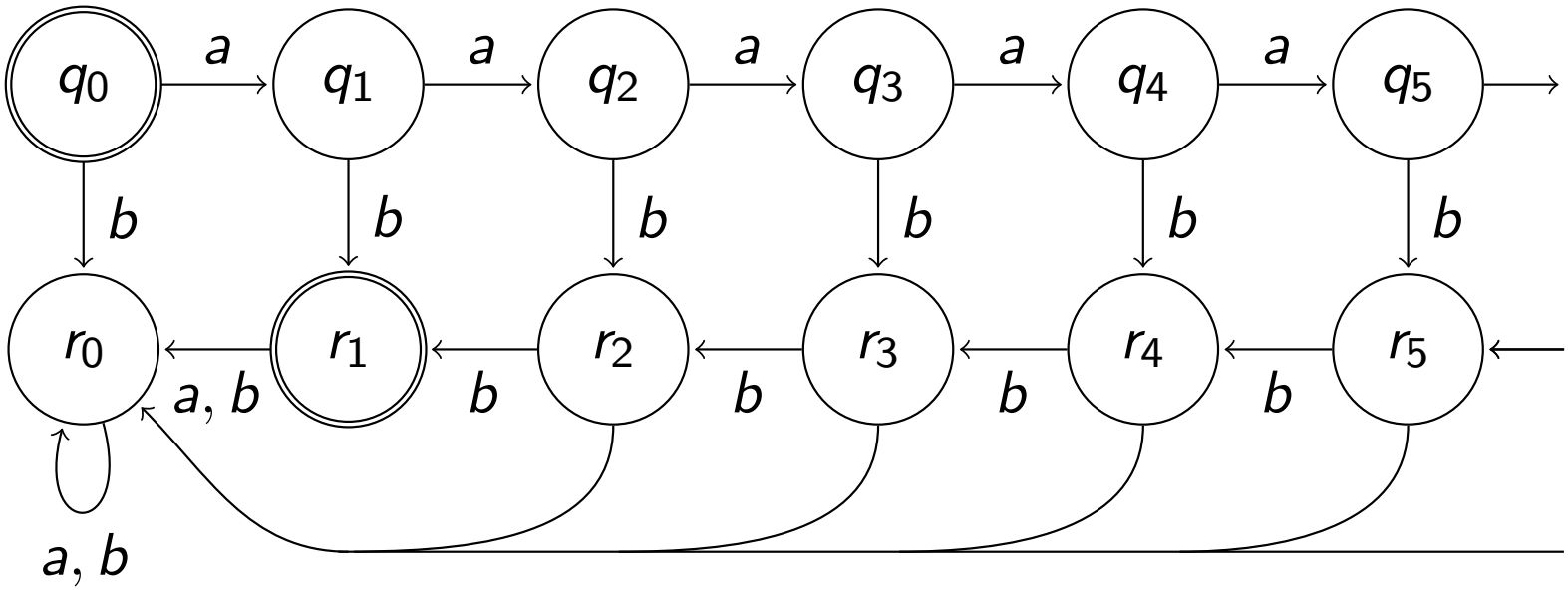
We can extend the notion of derivatives from alphabet symbols to words inductively (1 denotes the empty word):

$$\begin{aligned} q_1 &= q \\ q_{a \cdot w} &= (q_a)_w \end{aligned}$$

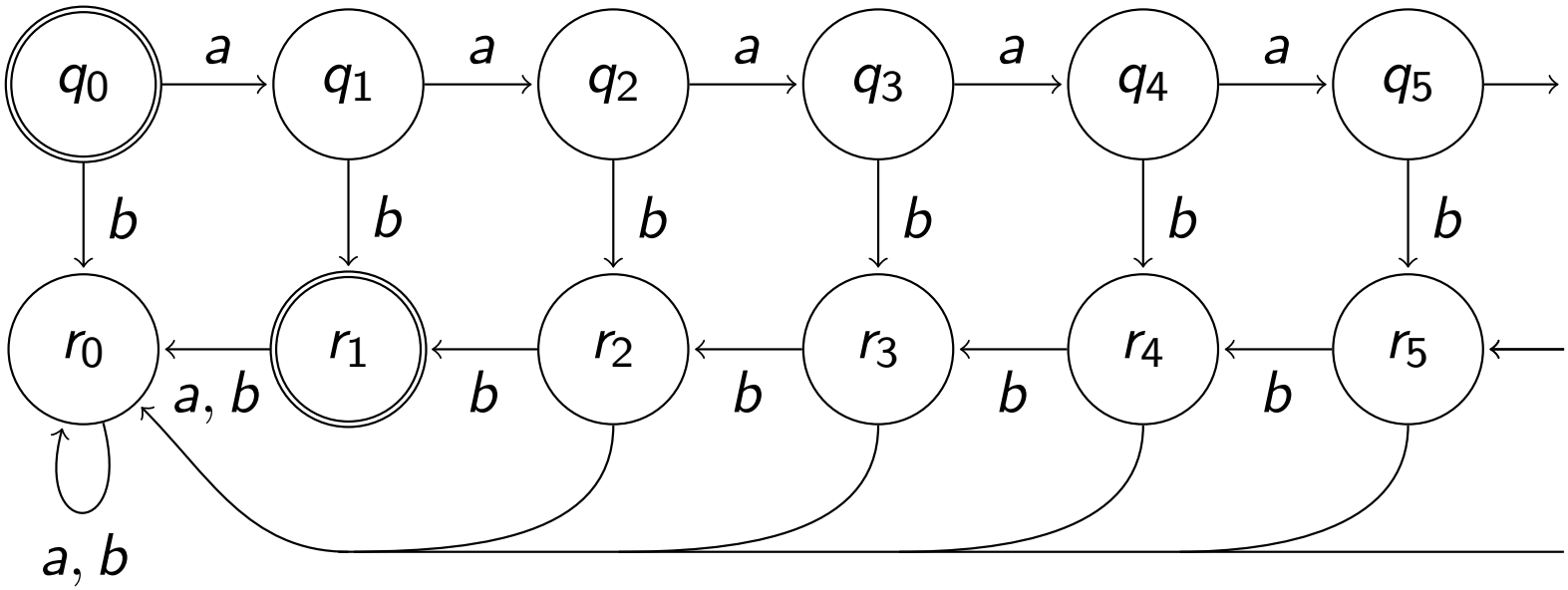
In the above automaton, we have e.g.

$$q_{abb} = q$$

An infinite automaton...



An infinite automaton...



Corresponding system of b.d.e.'s:

$$
\begin{array}{rclcrclcrcl}
o(q_i) & = & 0 & & (q_i)_a & = & q_{i+1} & & (q_i)_b & = & r_i \\
o(r_0) & = & 0 & & (r_0)_a & = & r_0 & & (r_0)_b & = & r_0 \\
o(r_1) & = & 1 & & (r_1)_a & = & r_0 & & (r_1)_b & = & r_0 \\
o(r_{k+2}) & = & 0 & & (r_{k+2})_a & = & r_0 & & (r_{k+2})_b & = & r_{k+1}
\end{array}
$$

A very large automaton

Consider the automaton $(\mathcal{L}, o, \delta)$:

- $\mathcal{L} = \mathcal{P}(\mathcal{P}(A^*))$, the set of all languages over A
- $o(L)$ is 1 if $1 \in L$, and 0 otherwise
- $L_a = \{w \mid aw \in L\}$ for all alphabet symbols a

A very large automaton

Consider the automaton $(\mathcal{L}, o, \delta)$:

- $\mathcal{L} = \mathcal{P}(\mathcal{P}(A^*))$, the set of all languages over A
- $o(L)$ is 1 if $1 \in L$, and 0 otherwise
- $L_a = \{w \mid aw \in L\}$ for all alphabet symbols a

Proposition

For every automaton (Q, o', δ'), there is exactly one mapping $[\![\ ]\!] : Q \to \mathcal{L}$, such that

1. $o'(q) = o([\![q]\!])$ *for all states q, and*
2. $[\![q_a]\!] = [\![q]\!]_a$ *for all states q and alphabet symbols a.*

A very large automaton

Consider the automaton $(\mathcal{L}, o, \delta)$:

- $\mathcal{L} = \mathcal{P}(\mathcal{P}(A^*))$, the set of all languages over A
- $o(L)$ is 1 if $1 \in L$, and 0 otherwise
- $L_a = \{w \mid aw \in L\}$ for all alphabet symbols a

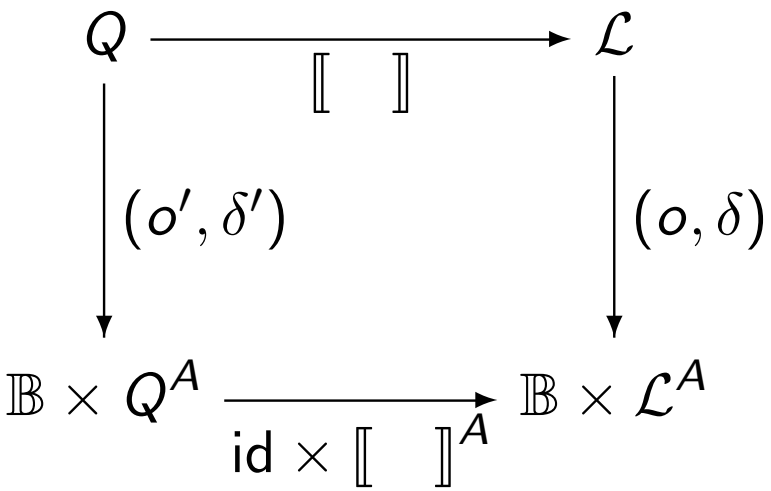
Proposition

For every automaton (Q, o', δ'), there is exactly one mapping $[\![\ ]\!] : Q \to \mathcal{L}$, such that

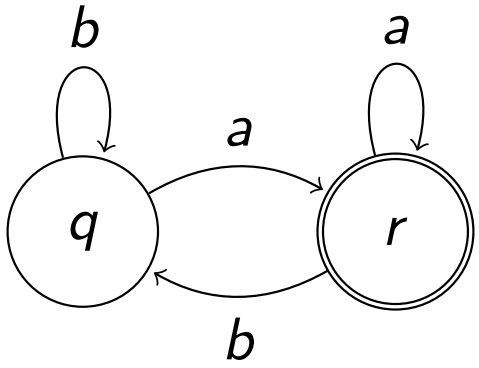
1. $o'(q) = o([\![q]\!])$ *for all states q, and*
2. $[\![q_a]\!] = [\![q]\!]_a$ *for all states q and alphabet symbols a.*

In other words: $\mathcal{L}$ is a *final automaton* or *final coalgebra*.

This is not a categorical diagram (1)



The final homomorphism



$[\![\ ]\!]$ assigns to states the language accepted by those states:

$$\begin{aligned} [\![q]\!] &= \{w \mid w \text{ ends in } a\} \\ [\![r]\!] &= \{w \mid w \text{ ends in } a \text{ or is empty}\} \end{aligned}$$

Bisimulation (1)

A bisimulation between two automata with state spaces Q_0 and $Q_1 \ldots$

Bisimulation (1)

A bisimulation between two automata with state spaces Q_0 and Q_1...

...is a relation $R \subset Q_0 \times Q_1$ such that, whenever $(q, r) \in R$:

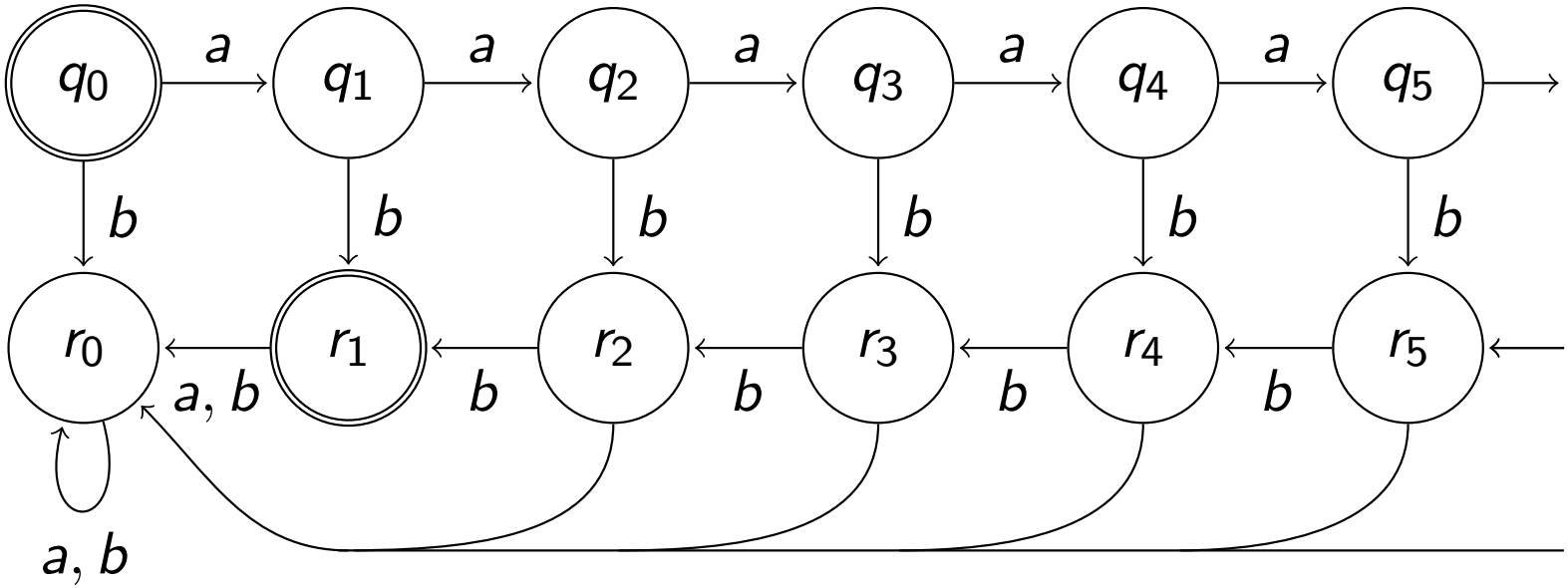
1. $o(q) = o(r)$, and ...
2. $(q_a, r_a) \in R$ for all alphabet symbols a.

Bisimulation (2)

Fact

If $(q, r) \in R$ *for some bisimulation* R*, then* $[\![q]\!] = [\![r]\!]$.

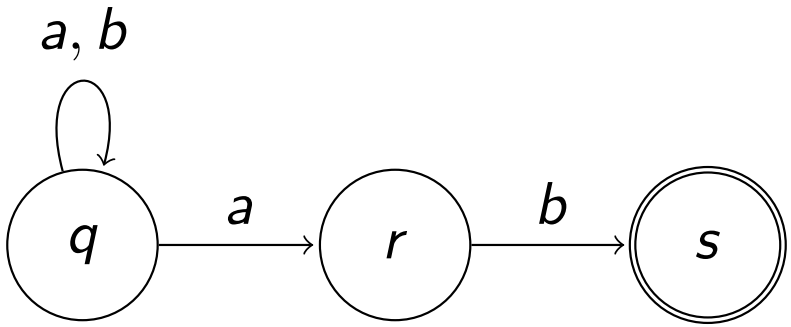
Bisimulation (3)



$$R = \{(q_i, a^n b^{n+i}) \mid i \in \mathbb{N}\} \cup \{(r_0, 0), (r_1, 1)\} \cup \{(r_{i+2}, b^{i+1}) \mid i \in \mathbb{N}\}$$

is a bisimulation. Hence, $[\![q_0]\!] = a^n b^n$, etc. . .

A nondeterministic automaton



$$\begin{array}{lllllllll} o(q) & = & 0 & & q_a & = & q+r & & q_b & = & q \\ o(r) & = & 0 & & r_a & = & 0 & & r_b & = & s \\ o(s) & = & 1 & & s_a & = & 0 & & s_b & = & 0 \end{array}$$

Right hand sides of the equations: sums of states.

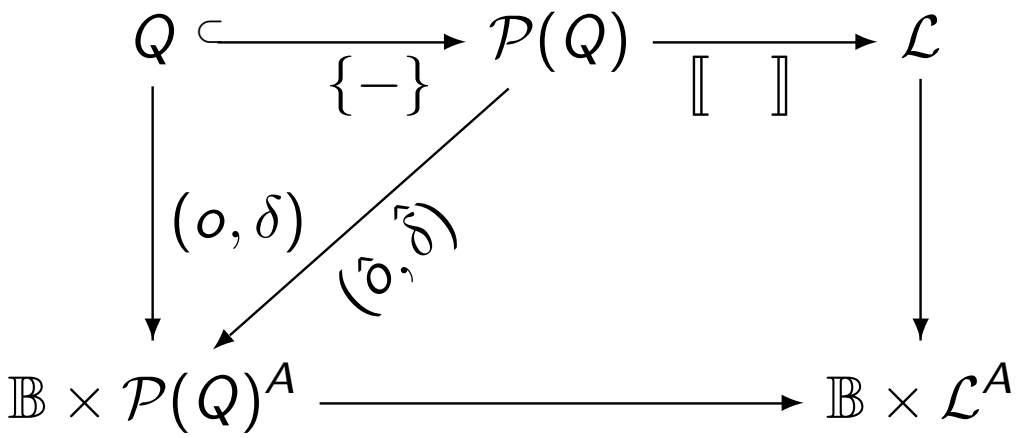
Determinization

We can transform a nondeterministic automaton (Q, o, δ) into a deterministic automaton $(\mathcal{P}(Q), \hat{o}, \hat{\delta})$, by enforcing the rules

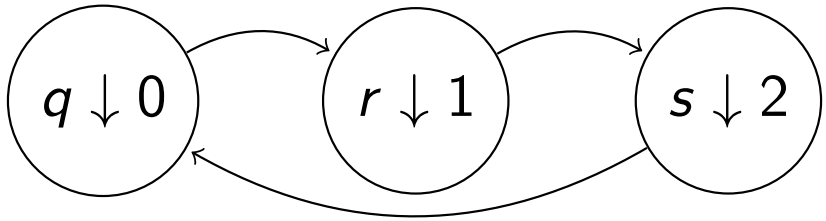
$$\begin{array}{rcl@{\qquad}rcl} \hat{o}(0) & = & 0 & 0_a & = & 0 \\ \hat{o}(s+t) & = & \max(o(s), o(t)) & (s+t)_a & = & s_a + t_a \end{array}$$

(Representing elements $\{q_1, \ldots q_n\} \in \mathcal{P}(Q)$ as sums $\sum_{1 \leq i \leq n} q_i$).

This is not a categorical diagram (2)



A simple stream automaton

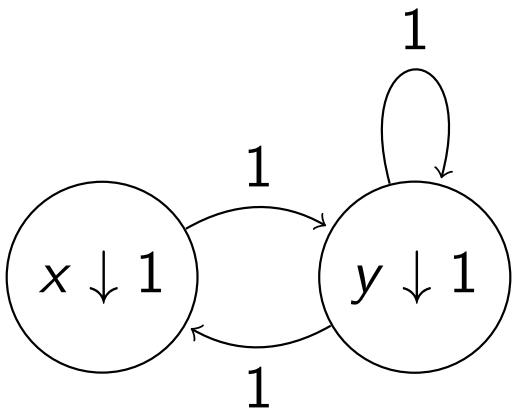


$$\begin{array}{lllllllll} o(q) & = & 0 & q' & = & r & [\![q]\!] & = & (0,1,2,0,1,2,\ldots) \\ o(r) & = & 1 & r' & = & s & [\![r]\!] & = & (1,2,0,1,2,0,\ldots) \\ o(s) & = & 2 & s' & = & q & [\![s]\!] & = & (2,0,1,2,0,1,\ldots) \end{array}$$

(Finite stream automata are not very spectacular)

A weighted automaton (1)

A weighted automaton...



...and the corresponding system of equations ...

$$o(x) = 1 \qquad x' = y$$
$$o(y) = 1 \quad y' = y + x$$

Right hand side: linear combination of states

Determinization (2)

We can transform a weighted automaton (Q, o, δ) into a stream automaton $(\mathrm{span}(Q), \hat{o}, \hat{\delta})$, by enforcing the rule

$$\hat{o}\left(\sum_{0\leq i<k} r_i q_i\right) = \sum_{0\leq i<k} r_i o(q_i) \qquad \left(\sum_{0\leq i<k} r_i q_i\right)' = \sum_{0\leq i<k} r_i q_i'$$

for all linear combinations of elements $q \in Q$.

The Fibonacci sequence

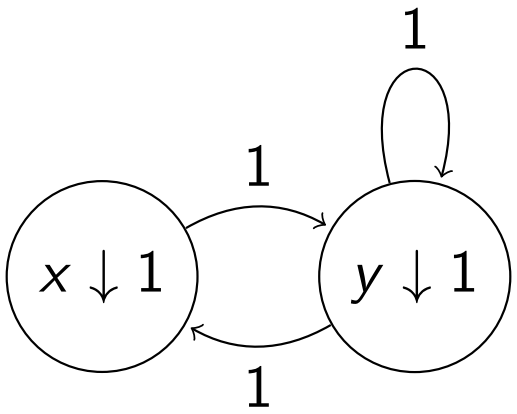
$$\mathbf{fib} = (1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots)$$

Inductive definition:

$$\mathbf{fib}(0) = \mathbf{fib}(1) = 1 \qquad \mathbf{fib}(n+2) = \mathbf{fib}(n+1) + \mathbf{fib}(n)$$

A weighted automaton (2)

A weighted automaton...



...and the corresponding system of equations ...

$$o(x) = 1 \qquad x' = y$$
$$o(y) = 1 \quad y' = y + x$$

Proposition

$[\![x]\!] = \mathbf{fib}$

A proof

The relation

$$R = \{(\mathbf{fib}, x), (\mathbf{fib}', y)\} \cup \{(\mathbf{fib}^{(n+2)}, \mathbf{fib}(n)x + \mathbf{fib}(n+1)y) \mid n \in \mathbb{N}\}$$

is a bisimulation w.r.t. the function **fib** and the system

$$o(x) = 1 \qquad o(y) = 1 \qquad x' = y \qquad y' = x + y.$$

A proof

The relation

$$R = \{(\mathbf{fib}, x), (\mathbf{fib}', y)\} \cup \{(\mathbf{fib}^{(n+2)}, \mathbf{fib}(n)x + \mathbf{fib}(n+1)y) \mid n \in \mathbb{N}\}$$

is a bisimulation w.r.t. the function **fib** and the system

$$o(x) = 1 \qquad o(y) = 1 \qquad x' = y \qquad y' = x + y.$$

E.g.:

$$\begin{aligned} \mathbf{fib}^{(n+2)\prime} &= \mathbf{fib}^{(n+3)} \\ &\overset{R}{} \quad \mathbf{fib}(n+1)x + \mathbf{fib}(n+2)y \\ &= \mathbf{fib}(n+1)x + (\mathbf{fib}(n+1) + \mathbf{fib}(n))y \\ &= (\mathbf{fib}(n)x + \mathbf{fib}(n+1)y)' \end{aligned}$$

More complex systems

We now look at systems where the right hand side of any derivative can be a polynomial over the set X of variables, e.g.

$$x^3y^3 + xy^2 + y$$

We determinize using the product rule

$$o(st) = o(s)o(t) \qquad (st)_a = s_a t + o(s)t_a$$

(in addition to the earlier linearity rules)
Resulting class of languages: the context-free languages!

A context-free system

An example of such a system:

$$\begin{array}{llllll} o(x) = 1 & x_a = xy & x_b = 0 \\ o(y) = 0 & y_a = 0 & y_b = 1 \end{array}$$

A context-free system

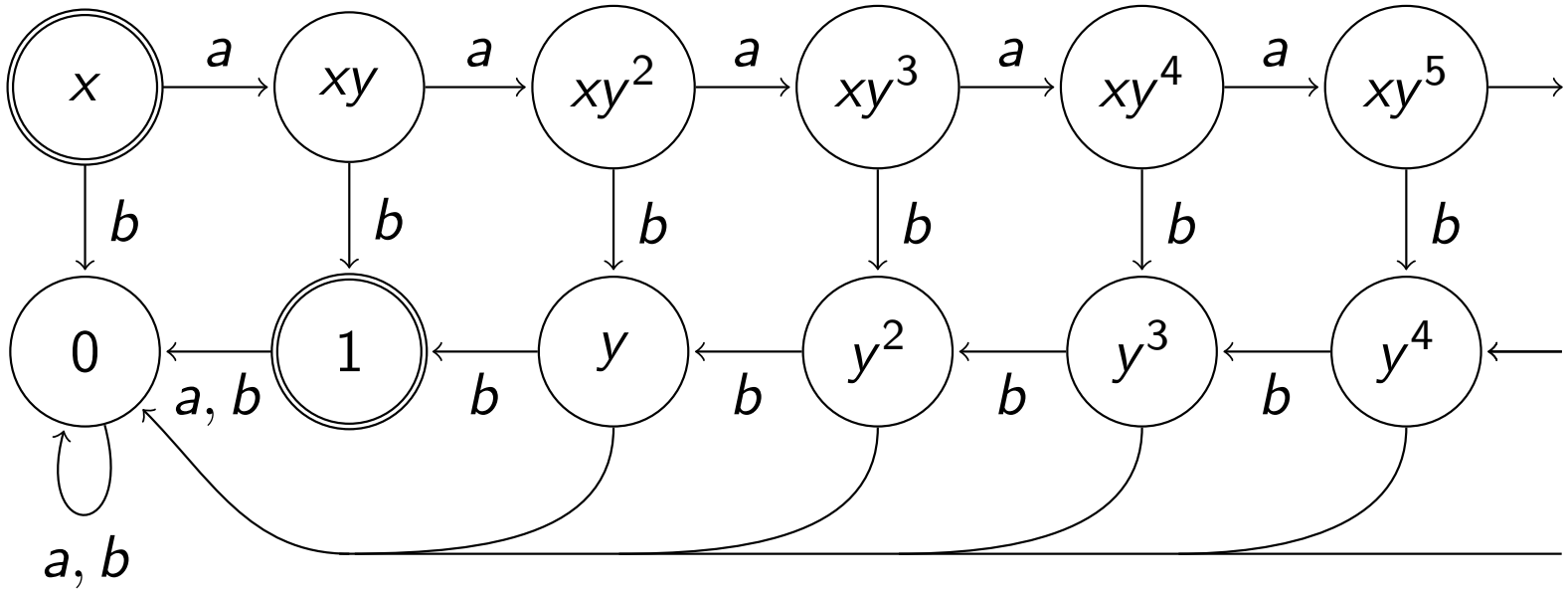
An example of such a system:

$$\begin{array}{lllllllll} o(x) & = & 1 & \quad x_a & = & xy & \quad x_b & = & 0 \\ o(y) & = & 0 & \quad y_a & = & 0 & \quad y_b & = & 1 \end{array}$$

We get

$$[\![x]\!] = a^n b^n$$

Determinization of a context-free system



Classes of b.d.e., their languages and streams

rhs	Class of languages	Class of streams
Single element	Regular	Finitary
Linear combination	(Nondeterministic) regular	Rational
Polynomial	Context-free	Algebraic

Conclusions

- Bisimulation is a very nice coinductive proof technique!
- Behavioural differential equations are an effective way of defining automata-like structures.
- Casting a new light on the existing theory of (weighted) automata.