Coalgebraic Determinization of Alternating Automata

M1 Internship Presentation

Meven Bertrand, supervision by Juriaan Rot

September 6 2017

ENS DE LYON

Introduction

Context

Coalgebra: category theory for state based systems

My own motivation

- interest in category theory for some time
- course on coalgebra during the 1st semester
- wanted to go further

Only 1 month of internship due to course schedule

Why this subject?

Determinization: natural problem in coalgebra
Alternating automata: looks like a failure of theory → interesting to study
For me: reasonable background, interesting but not too large question

Introduction

Context

Coalgebra: category theory for state based systems

My own motivation

- interest in category theory for some time
- course on coalgebra during the 1^{st} semester
- wanted to go further

Only 1 month of internship due to course schedule

Why this subject?

Determinization: natural problem in coalgebra
Alternating automata: looks like a failure of theory → interesting to study
For me: reasonable background, interesting but not too large question

Introduction

Context

Coalgebra: category theory for state based systems

My own motivation

- interest in category theory for some time
- course on coalgebra during the 1st semester
- wanted to go further

Only 1 month of internship due to course schedule

Why this subject?

Determinization: natural problem in coalgebra
Alternating automata: looks like a failure of theory → interesting to study
For me: reasonable background, interesting but not too large question

Plan

Plan

ENS DE LYON

Examples

Examples

- stream: $Q \to \Gamma \times Q$
- automaton: $Q \to \mathbf{2} \times Q^{\Sigma}$
- pushdown automaton: $Q \to \mathbf{2} \times ((\Gamma^* \times Q)^{\Gamma^*})^{\Sigma}$
- non-deterministic automaton: $Q \to \mathbf{2} \times \mathcal{P}(Q)^{\Sigma}$
- Moore machine: $Q \to \Gamma \times Q^{\Sigma}$

What Is Really a State-Based System?

Two elements: states, and transitions (maybe with observable output).

Transitions

Simplest: state $\to$ state
Usual ingredients:

- acceptance: $\mathbf{2} \times (-)$
- generic output: $\Gamma \times (-)$
- reading letters: $(-)^{\Sigma}$
- branching: $\mathcal{P}(-)$
- side-effect: $(M \times (-))^{M}$

Many more

Examples

Examples

- stream: output $= Q \to \Gamma \times Q$
- automaton: acceptance + reading letter $= Q \to \mathbf{2} \times Q^{\Sigma}$
- pushdown automaton: acceptance + reading letter + side-effect $=$ $Q \to \mathbf{2} \times ((\Gamma^* \times Q)^{\Gamma^*})^{\Sigma}$
- non-deterministic automaton: acceptance + reading letter + branching $=$ $Q \to \mathbf{2} \times \mathcal{P}(Q)^{\Sigma}$
- Moore machine: output + reading letter $= Q \to B \times Q^{\Sigma}$

Why Category Theory?

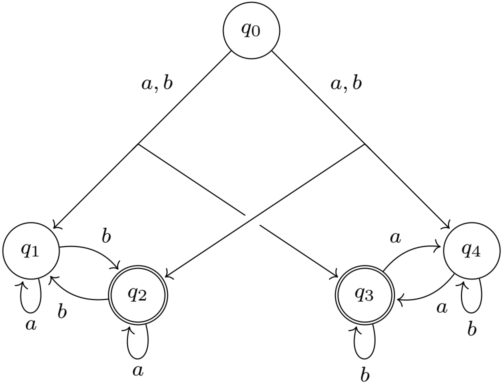
All the ingredients are examples of **functors**.
Functors are a central study subject of category theory, so category theory is a good tool to study state-based systems.

State-based systems	Category theory
Type of system	Functor $\mathcal{F}$
Particular instance	coalgebra $Q \to \mathcal{F}(Q)$
$\vdots$	$\vdots$

ENS DE LYON

Plan

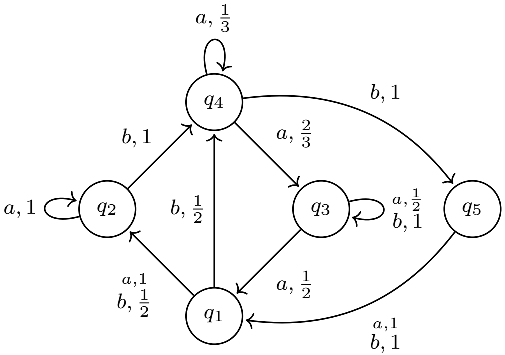
Alternating Automata



Similar to non-deterministic automata, but with existential/universal alternation.
Coalgebra for $\mathbf{2} \times \mathcal{P}(\mathcal{P}(-))^A$.

Other Examples of Non-Determinism

Simplest example: non-deterministic automaton, coalgebra for $\mathbf{2} \times \mathcal{P}(-)^A$.
Another example: probabilistic automaton, coalgebra for $\mathbf{2} \times \mathbb{P}(-)^A$.



Monads For Non-Determinism

There is a common structure: $\underbrace{\mathcal{F}}_{\text{Original functor}} = \underbrace{\mathcal{G}}_{\text{“machine” part}} \circ \underbrace{\mathcal{T}}_{\text{non-determinism}}$

But $\mathcal{T}$ has more structure, it is a monad:

Monad

A monad has three components:

- a functor $\mathcal{T}$
- a unit: collection of $\eta_X : X \to \mathcal{T}(X)$
- a multiplication: collection of $\mu_X : \mathcal{T}(\mathcal{T}(X)) \to \mathcal{T}(X)$

In this setting, there are nice theorems about transforming a coalgebra $X \to \mathcal{G}(\mathcal{T}(X))$ into a coalgebra $\mathcal{T}(X) \to \mathcal{G}(\mathcal{T}(X))$ (“determinization”).

Plan

ENS DE LYON

The Usual Construction

For these theorems to work, we need a monad structure for $\mathcal{P}\mathcal{P}$.
The functor $\mathcal{P}$ is already a monad, so the natural way is to use monad composition.
The main ingredient is a distributive law:

Distributive law

A distributive law $\lambda : \mathcal{T}\ \mathcal{T}' \Rightarrow \mathcal{T}'\mathcal{T}$ is a family of functions $\lambda_X : \mathcal{T}(\mathcal{T}'(X)) \rightarrow \mathcal{T}'(\mathcal{T}(X))$ respecting some axioms.

But for $\mathcal{P}\mathcal{P}$, the natural constructions fail.

The Failure on $\mathcal{P}\mathcal{P}$

The intuitive distribution is

$$(x_1 \vee x_2) \wedge (y_1 \vee y_2 \vee y_3) \mapsto (x_1 \wedge y_1) \vee (x_1 \wedge y_2) \vee (x_1 \wedge y_3) \vee (x_2 \wedge y_1) \vee (x_2 \wedge y_2) \vee (x_2 \vee y_3)$$

as formula:

$$\begin{array}{rccl} \lambda_X : & \mathcal{P}(\mathcal{P}(X)) & \to & \mathcal{P}(\mathcal{P}(X)) \\ & S & \mapsto & \{V \subseteq \cup S \mid \forall\, U \in S,\ \mathrm{Card}(V \cap U) = 1\} \end{array}$$

Changing it to

$$\begin{array}{rccl} \lambda'_X : & \mathcal{P}(\mathcal{P}(X)) & \to & \mathcal{P}(\mathcal{P}(X)) \\ & S & \mapsto & \{V \subseteq \cup S \mid \forall\, U \in S,\ \mathrm{Card}(V \cap U) \geq 1\} \end{array}$$

is better, but still not correct.

A Correct Distributive Law

There is still something worth noting: $\lambda'_X(S) = \uparrow \lambda_X(S)$, so order plays a role.
Hence, change from sets to ordered sets, and from $\mathcal{P}\mathcal{P}$ to $\mathcal{U}p\,\mathcal{D}n$.

With this change in type,

$$\begin{array}{rccl} \lambda''_X : & \mathcal{D}n(\mathcal{U}p(X)) & \to & \mathcal{U}p(\mathcal{D}n(X)) \\ & S & \mapsto & \{V \subseteq X \mid \forall\, U \in S,\ \mathrm{Card}(V \cap U) \geq 1\} \end{array}$$

is a distributive law!

With a little trick to go from sets to ordered sets and back, the problem is solved:

$$\mathbf{Set} \underset{\mathcal{U}}{\overset{\mathcal{O}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathbf{Poset}$$

It Works!

Semantics from this monad for $\langle o, \delta \rangle : Q \to \mathbf{2} \times \mathcal{U}\,\mathcal{U}p\,\mathcal{D}n\,\mathcal{O}(Q)^A$:

- $\text{behaviour}(q)(\varepsilon) = o(q)$
- $\text{behaviour}(q)(a \cdot w) = \mathbf{1} \Leftrightarrow \exists\, F \in \delta(q)(a),\ \forall\, q' \in F,\ \text{behaviour}(q')(w) = \mathbf{1}$

As we want.

Conclusion

Interest

- nice and powerful framework for state-based systems
- showing the value of category theory

To do next?

- Study the possibility to give a distributive law for $\mathcal{P}\mathcal{P}$, more comparison
- Add negation

ENS DE LYON