Two-Way Reversible Transducers as Functors

Ian Price
countingishard.org

Swansea University

British Colloquium for Theoretical Computer Science
April 4, 2024

Outline

Categories

High-level Idea

A Category is an abstract theory of functions.

Definition (Category)

A *category* $\mathcal{C}$ consists of a collection of *objects* $A, B, C, \ldots$ and a collection of *arrows* $f, g, h, \ldots$ such that

- Any arrow $f : \mathrm{dom}(f) \to \mathrm{cod}(f)$ has two objects $\mathrm{dom}(f)$ and $\mathrm{cod}(f)$.
- Given two compatible arrows $f : B \to C$, $g : A \to B$, there is a *composite arrow* $f \circ g : A \to C$.
- There is an *identity arrow* for every object A, $\mathrm{id}_A : A \to A$.
- $\circ$ is associative and id_A are identities for $\circ$.

Categories (Examples)

Example

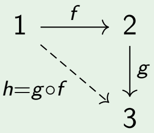
Most mathematical objects can be bundled into categories: **Set**, **Rel**, $\mathbf{Vect}_{\mathbb{K}}$, . . .

Example

Various objects can be thought of as categories: monoids, posets, . . .

Example (3)

You can freely generate categories from graphs



Functors

High-level Idea

Functors are category homomorphisms.

Definition (Functor)

A *functor* $F : \mathcal{C} \to \mathcal{D}$ between categories $\mathcal{C}$ and $\mathcal{D}$ is a pair of mappings on objects and arrows such that

- Given any $\mathcal{C}$-object X, $F(X)$ is a $\mathcal{D}$-object.
- Given any $\mathcal{C}$-arrow $f : A \to B$, $F(f) : F(A) \to F(B)$ is a $\mathcal{D}$-arrow.
- $F(f \circ g) = F(f) \circ F(g)$
- $F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$

Functors (cont.)

Essentially, this entire talk is giving a bunch of examples.

Useful analogy

We will often view a domain category $\mathcal{C}$ as a kind of "syntax" and functors $\mathcal{C} \to \mathcal{D}$ as a kind of "semantics" for it.

The "Syntax" of Automata

Definition (Shape$_\Sigma$)

For any finite alphabet Σ, there is a three object category ***Shape***$_\Sigma$ generated by the following finite graph, where there is one morphism for each letter $a \in \Sigma$.

$$\mathrm{in} \xrightarrow{\triangleright} \mathrm{states} \xrightarrow{\triangleleft} \mathrm{out} \qquad (a : \mathrm{states} \to \mathrm{states})$$

$$\begin{array}{rcl} \text{words over } \Sigma & \cong & \text{morphisms } \mathrm{in} \to \mathrm{out} \\ \text{“abc”} & \mapsto & \triangleright \,;\, a \,;\, b \,;\, c \,;\, \triangleleft \end{array}$$

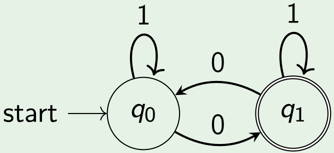
DFAs

Definition (Deterministic Finite Automata)

A *(deterministic) finite automaton* $\mathcal{T}$ is a tuple $(Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set of *states*,
- Σ is a finite alphabet,
- $\delta : Q \times \Sigma \to Q$ is the *transition function*
- $q_0 \in Q$ is the *initial state*,
- $F \subseteq Q$ is the set of *final states*.

Example



DFAs as a Functor

Definition (DFA (revised))

A *(deterministic) finite automaton* with input alphabet Σ is a functor $F : \mathbf{Shape}_\Sigma \to \mathbf{Set}$ such that $F(\mathrm{in}) = \{\bullet\}$, $F(\mathrm{states})$ is non-empty and $F(\mathrm{out}) = \{\mathrm{false}, \mathrm{true}\}$.

We recover the previous definition by setting

- $Q = F(\mathrm{states})$,
- $q_0 = F(\triangleright)(\bullet)$
- $\delta(-, a) = F(a)$
- $x \in F$ iff $F(\triangleleft)(x) = \mathrm{true}$.

$F(\triangleright; w_1; \ldots; w_n; \triangleleft) : \{\bullet\} \to \{\mathrm{false}, \mathrm{true}\}$ is the constantly true function if and only if $w_1 \ldots w_n$ is in the language recognised by the DFA.

Other familiar examples

Definition (NFA)

A *nondeterministic finite automaton* with input alphabet Σ is a functor $F : \mathbf{Shape}_{\Sigma} \to \mathbf{Rel}$ such that $F(\text{in}) = \{\bullet\}$, $F(\text{states})$ is non-empty and $F(\text{out}) = \{\text{false}, \text{true}\}$.

Definition (Weighted automata over $\mathbb{K}$)

A *weighted finite automaton* with input alphabet Σ over the field $\mathbb{K}$ is a functor $F : \mathbf{Shape}_{\Sigma} \to \mathbf{Vect}_{\mathbb{K}}$ such that $F(\text{in}) = F(\text{out}) = \mathbb{K}$ and $F(\text{states})$ is the vector space spanned by the states.

Subsequential Transducers can be defined if I use monads, but I promised the bare minimum of category theory.

2DFTs

Definition (Two-way Deterministic Transducer)

A *two-way deterministic transducer (2DFT)* $\mathcal{T}$ is a tuple $(Q, \rho, \Sigma, \Gamma, \delta, q_0, F)$, where

- Q is a finite set of *states*,
- $\rho : Q \to \{-1, 1\}$ is a *direction map*[a],
- Σ is a finite *input alphabet*,
- Γ is a finite *output alphabet*,
- $\delta : \Sigma \sqcup \{\triangleright, \triangleleft\} \to (Q \rightharpoonup \Gamma^\star \times Q)$ is the *transition function*
- $q_0 \in Q^\rightarrow$ is the *initial state*,
- $F \subseteq Q$ is a set of *final states*.

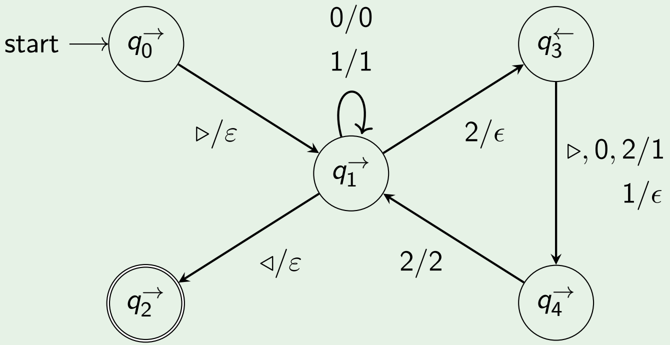
[a] We write $q^\rightarrow$ if $\rho(q) = 1$ and $q^\leftarrow$ if $\rho(q) = -1$

A 2DFT $\mathcal{T}$ defines a partial function $[\![\mathcal{T}]\!] : \Sigma^\star \to \Gamma^\star$ where the input string $\triangleright w_1 \dots w_n \triangleleft$ is sent to a "valid" sequence of configurations by δ.

2DFT Example

Example

The following 2DFT takes any string and ensures that every 2 is preceded by a 1 by adding 1s if necessary.



2RFTs

We will focus on a specific subclass of 2DFTs.

Definition (Two-way Reversible Transducer)

A *two-way reversible transducer (2RFT)* $\mathcal{T}$ with input alphabet Σ and output alphabet Γ is a 2DFT such that

- F is a singleton
- $\delta(a)$ is a partial *injection* for each $a \in \Sigma$.

Transition Diagrams

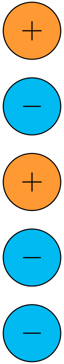
Compared with DFAs we have more structure here

- We can go forwards *and backwards* along the tape
- We need some way to "output" strings
- We require injectivity

This is solved by introducing a new category of "transition diagrams" **TransDiag**.

Objects

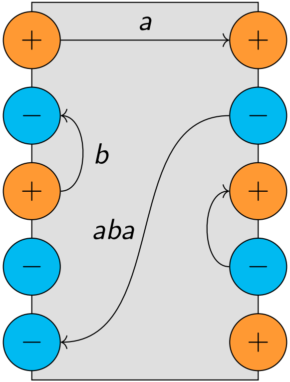
Objects are binary[1] words, which we write vertically top to bottom



[1] We write the objects as + and − rather than 1 and 0

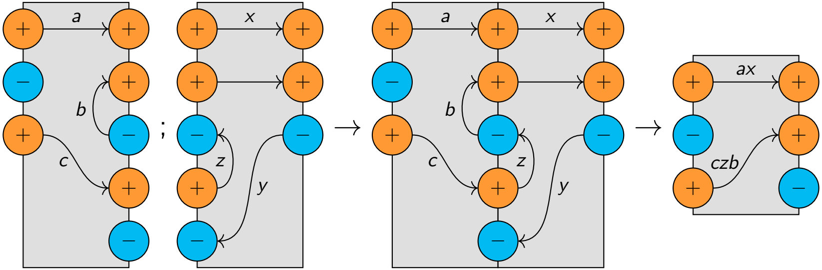
Morphisms

Morphisms are special “diagrams” between these words subject to some restrictions regarding polarity and vertex degree.
Edges are labelled with strings in the output alphabet.

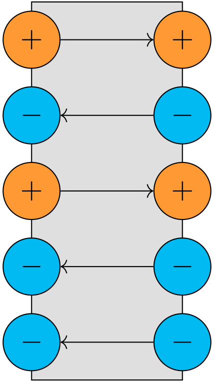


Composition

In a nutshell, glue them together and then concatenate strings along the path.



Identities



2RFTs as a Functor

Definition (Two-way Reversible Transducer)

A *two-way reversible transducer (2RFT)* $\mathcal{T}$ with input alphabet Σ and output alphabet Γ is a functor $F : \mathbf{Shape}_\Sigma \to \mathbf{TransDiag}_\Gamma$ such that

- $F(\mathrm{states})$ is a non-empty binary word,
- $F(\mathrm{in})$ and $F(\mathrm{out})$ are both the binary word $+$.

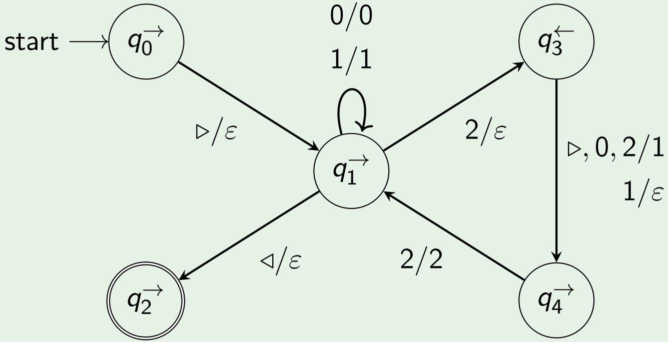
For any word $w \in \Sigma^\star$ we have a planar diagram $F(w) : + \to +$ which is either the empty diagram or a single path with label l. This gives rise to the partial function $[\![\mathcal{T}]\!] : \Sigma^\star \rightharpoonup \Gamma^\star$.

2RFT-Functor Example

Let's take the same example from before and find the equivalent functor.

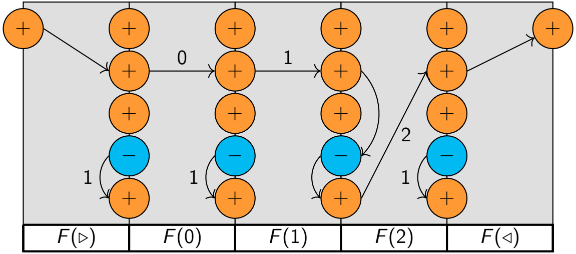
Example

The following 2RFT takes any string and ensures that every 2 is preceded by a 1 by adding 1s if necessary.



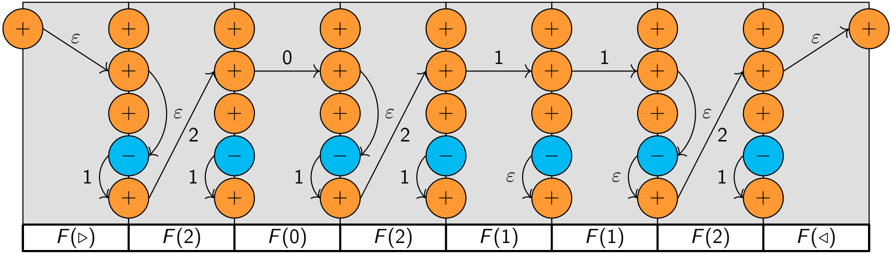
2RFT-Functor Example (cont.)

	$\triangleright$	$\triangleleft$	0	1	2
$\overrightarrow{q_0}$	$\overrightarrow{q_1}/\varepsilon$				
$\overrightarrow{q_1}$		$\overrightarrow{q_2}/\varepsilon$	$\overrightarrow{q_1}/0$	$\overrightarrow{q_1}/1$	$\overleftarrow{q_3}/\varepsilon$
$\overrightarrow{q_2}$					
$\overleftarrow{q_3}$	$\overrightarrow{q_4}/1$		$\overrightarrow{q_4}/1$	$\overrightarrow{q_4}/\varepsilon$	$\overrightarrow{q_4}/1$
$\overrightarrow{q_4}$					$\overrightarrow{q_1}/2$



2RFT-Functor Example (cont.)

To run a transducer on a string, simply paste diagrams together, e.g., take $w = 202112$



$$F(\triangleright; w; \triangleleft) = F(\triangleright); F(2); F(0); F(2); F(1); F(1); F(2); F(\triangleleft) = + \xrightarrow{12012112} +$$

Where do we go from here?

- We are interested not just in these diagrams, but *planar* diagrams.
- Planar diagrams give an autonomous category (not symmetric).
- Links to the non-commutative linear / affine lambda calculus.

Theorem (Pradic & Price, '24)

The following are equivalent:

- *Affine string-to-string* $\lambda\wp$ *definable functions*
- *first-order string transductions*
- *planar reversible two-way finite transducers*

References

- Colcombet and Petrișan, "Automata Minimization: a Functorial Approach"
- Colcombet, Petrișan and Stabile, "Learning automata and transducers: a categorical approach"
- Nguyễn, Noûs, and Pradic, "Two-way automata and transducers with planar behaviours are aperiodic"
- Pradic and Price, "Implicit automata in λ-calculi III: first-order transductions and affine planar λ-terms" (Coming Soon)

Thank You!
Any Questions?