Automata minimization and glueing of categories

13 12 2017 Berkeley

Thomas Colcombet
joint work with Daniela Petrişan

Automata minimization and glueing of categories

[MFCS 2017] & [Informal presentation in SIGLOG column]

13 12 2017 Berkeley

Thomas Colcombet
joint work with Daniela Petrişan

Description of the situation

Automata

Automata

An deterministic automaton is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is a set of states,

$i : 1 \to Q$ is the initial map

$f : Q \to 2$ is the final map

$\delta_a : Q \to Q$ is the transition map

Automata

Rabin & Scott

An deterministic automaton is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is a set of states,

$i : 1 \to Q$ is the initial map

$f : Q \to 2$ is the final map

$\delta_a : Q \to Q$ is the transition map

Automata

Rabin & Scott

An deterministic automaton is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is a set of states,

$i\colon 1 \to Q$ is the initial map

$f\colon Q \to 2$ is the final map

$\delta_a\colon Q \to Q$ is the transition map

It computes the language:

$$\begin{aligned} [\![\mathcal{A}]\!] \colon A^* &\to [1, 2] \\ u &\mapsto f \circ \delta_u \circ i \end{aligned}$$

Automata

Rabin & Scott

An deterministic automaton is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is a set of states,

$i : 1 \to Q$ is the initial map

$f : Q \to 2$ is the final map

$\delta_a : Q \to Q$ is the transition map

It computes the language:

$$\llbracket \mathcal{A} \rrbracket : A^* \to [1, 2] \approx 2$$
$$u \mapsto f \circ \delta_u \circ i$$

Automata

Rabin & Scott

An **deterministic automaton** is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is a set of **states**,

$i : 1 \to Q$ is the **initial map**

$f : Q \to 2$ is the **final map**

$\delta_a : Q \to Q$ is the **transition map**

It computes the **language**:

$$[\![\mathcal{A}]\!] : A^* \to [1, 2] \approx 2$$
$$u \mapsto f \circ \delta_u \circ i$$

A **vector automaton** is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is an $\mathbb{R}$-vector space

$i : \mathbb{R} \to Q$ is a linear map

$f : Q \to \mathbb{R}$ is a linear map

$\delta_a : Q \to Q$ is a linear map

Automata

Rabin & Scott

Schützenberger's automata weighted over a field

An **deterministic automaton** is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

- Q is a set of **states**,
- $i : 1 \to Q$ is the **initial map**
- $f : Q \to 2$ is the **final map**
- $\delta_a : Q \to Q$ is the **transition map**

It computes the **language**:

$$[\![\mathcal{A}]\!] : A^* \to [1, 2] \approx 2$$

$$u \mapsto f \circ \delta_u \circ i$$

A **vector automaton** is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

- Q is an $\mathbb{R}$-vector space
- $i : \mathbb{R} \to Q$ is a linear map
- $f : Q \to \mathbb{R}$ is a linear map
- $\delta_a : Q \to Q$ is a linear map

Automata

Rabin & Scott

Schützenberger's automata weighted over a field

An deterministic automaton is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is a set of states,

$i \colon 1 \to Q$ is the initial map

$f \colon Q \to 2$ is the final map

$\delta_a \colon Q \to Q$ is the transition map

It computes the language:

$$[\![\mathcal{A}]\!] \colon A^* \to [1, 2] \approx 2$$
$$u \mapsto f \circ \delta_u \circ i$$

A vector automaton is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is an $\mathbb{R}$-vector space

$i \colon \mathbb{R} \to Q$ is a linear map

$f \colon Q \to \mathbb{R}$ is a linear map

$\delta_a \colon Q \to Q$ is a linear map

It computes the language:

$$[\![\mathcal{A}]\!] \colon A^* \to [\mathbb{R}, \mathbb{R}]$$
$$u \mapsto f \circ \delta_u \circ i$$

Automata

Rabin & Scott

Schützenberger's automata weighted over a field

An deterministic automaton is

$$\langle Q, i, f, (\delta_a)_{a\in A}\rangle$$

where

- Q is a set of states,
- $i\colon 1 \to Q$ is the initial map
- $f\colon Q \to 2$ is the final map
- $\delta_a\colon Q \to Q$ is the transition map

It computes the language:

$$\begin{aligned}[\![\mathcal{A}]\!]\colon A^* &\to [1,2] \approx 2\\ u &\mapsto f\circ\delta_u\circ i\end{aligned}$$

A vector automaton is

$$\langle Q, i, f, (\delta_a)_{a\in A}\rangle$$

where

- Q is an $\mathbb{R}$-vector space
- $i\colon \mathbb{R} \to Q$ is a linear map
- $f\colon Q \to \mathbb{R}$ is a linear map
- $\delta_a\colon Q \to Q$ is a linear map

It computes the language:

$$\begin{aligned}[\![\mathcal{A}]\!]\colon A^* &\to [\mathbb{R},\mathbb{R}] \approx \mathbb{R}\\ u &\mapsto f\circ\delta_u\circ i\end{aligned}$$

Automata

Rabin & Scott

Schützenberger's automata weighted over a field

An **deterministic automaton** is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is a set of **states**,

$i : 1 \to Q$ is the **initial map**

$f : Q \to 2$ is the **final map**

$\delta_a : Q \to Q$ is the **transition map**

It computes the **language**:

$$[\![\mathcal{A}]\!] : A^* \to [1, 2] \approx 2$$
$$u \mapsto f \circ \delta_u \circ i$$

A **vector automaton** is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is an $\mathbb{R}$-vector space

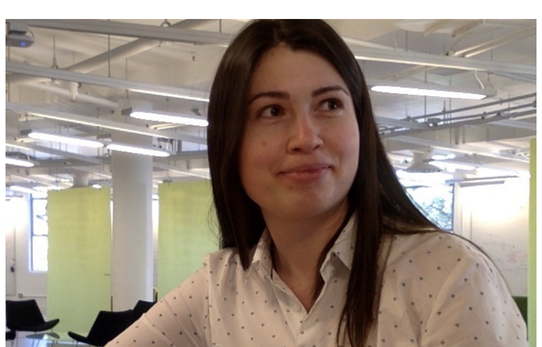
$i : \mathbb{R} \to Q$ is a linear map

$f : Q \to \mathbb{R}$ is a linear map

$\delta_a : Q \to Q$ is a linear map

It computes the **language**:

$$[\![\mathcal{A}]\!] : A^* \to [\mathbb{R}, \mathbb{R}] \approx \mathbb{R}$$
$$u \mapsto f \circ \delta_u \circ i$$



Automata

Rabin & Scott

Schützenberger's automata weighted over a field

An **deterministic automaton** is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

- Q is a set of **states**,
- $i : 1 \to Q$ is the **initial map**
- $f : Q \to 2$ is the **final map**
- $\delta_a : Q \to Q$ is the **transition map**

It computes the **language**:

$$[\![\mathcal{A}]\!] : A^* \to [1, 2] \approx 2$$
$$u \mapsto f \circ \delta_u \circ i$$

A **vector automaton** is

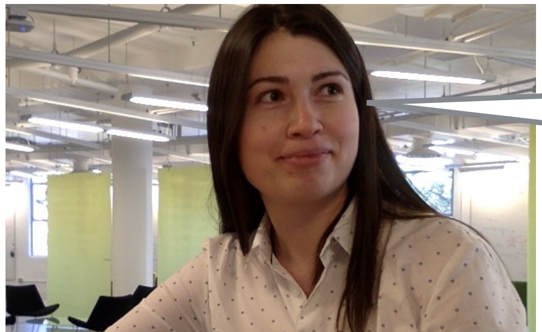
$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

- Q is an $\mathbb{R}$-vector space
- $i : \mathbb{R} \to Q$ is a linear map
- $f : Q \to \mathbb{R}$ is a linear map
- $\delta_a : Q \to Q$ is a linear map

It computes the **language**:

$$[\![\mathcal{A}]\!] : A^* \to [\mathbb{R}, \mathbb{R}] \approx \mathbb{R}$$
$$u \mapsto f \circ \delta_u \circ i$$



These data can modeled as a functor.

Example

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Q is an $\mathbb{R}$-vector space

$i\colon\ \mathbb{R} \to Q$ is a linear map

$f\colon Q \to \mathbb{R}$ is a linear map

$\delta_a\colon Q \to Q$ is a linear map

Example

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Q is an $\mathbb{R}$-vector space

$Q = \mathbb{R}^2$

$i\colon \mathbb{R} \to Q$ is a linear map

$f\colon Q \to \mathbb{R}$ is a linear map

$\delta_a\colon Q \to Q$ is a linear map

Example

$$L_{\mathtt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Q is an $\mathbb{R}$-vector space

$i\colon \mathbb{R} \to Q$ is a linear map

$f\colon Q \to \mathbb{R}$ is a linear map

$\delta_a\colon Q \to Q$ is a linear map

$Q = \mathbb{R}^2$

$i(x) = (x, 0)$

Example

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Q is an $\mathbb{R}$-vector space

$i\colon \mathbb{R} \to Q$ is a linear map

$f\colon Q \to \mathbb{R}$ is a linear map

$\delta_a\colon Q \to Q$ is a linear map

$Q = \mathbb{R}^2$

$i(x) = (x, 0)$

$f(x, y) = x$

Example

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Q is an $\mathbb{R}$-vector space — $Q = \mathbb{R}^2$

$i\colon \mathbb{R} \to Q$ is a linear map — $i(x) = (x, 0)$

$f\colon Q \to \mathbb{R}$ is a linear map — $f(x, y) = x$

$\delta_a\colon Q \to Q$ is a linear map

$$\delta_a(x, y) = (2x, 2y)$$
$$\delta_b(x, y) = (y, x)$$
$$\delta_c(x, y) = (0, 0)$$

Example

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Q is an $\mathbb{R}$-vector space

$i\colon \mathbb{R} \to Q$ is a linear map

$f\colon Q \to \mathbb{R}$ is a linear map

$\delta_a\colon Q \to Q$ is a linear map

$Q = \mathbb{R}^2$

$i(x) = (x, 0)$

$f(x, y) = x$

$\delta_a(x, y) = (2x, 2y)$

$\delta_b(x, y) = (y, x)$

$\delta_c(x, y) = (0, 0)$

Is it possible to do better?

A better implementation

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Solution in vector spaces

$$Q = \mathbb{R}^2$$

$$i(x) = (x, 0)$$

$$f(x, y) = x$$

$$\delta_a(x, y) = (2x, 2y)$$
$$\delta_b(x, y) = (y, x)$$
$$\delta_c(x, y) = (0, 0)$$

A better implementation

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Informally: use one bit for the parity to the number of b's.

Solution in vector spaces

$$Q = \mathbb{R}^2$$

$$i(x) = (x, 0)$$

$$f(x, y) = x$$

$$\delta_a(x, y) = (2x, 2y)$$
$$\delta_b(x, y) = (y, x)$$
$$\delta_c(x, y) = (0, 0)$$

A better implementation

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Informally: use one bit for the parity to the number of b's.

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

Solution in vector spaces

$$Q = \mathbb{R}^2$$

$$i(x) = (x, 0)$$

$$f(x, y) = x$$

$$\delta_a(x, y) = (2x, 2y)$$
$$\delta_b(x, y) = (y, x)$$
$$\delta_c(x, y) = (0, 0)$$

A better implementation

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Informally: use one bit for the parity to the number of b's.

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$
$$i(x) = (\texttt{even}, x)$$

Solution in vector spaces

$$Q = \mathbb{R}^2$$
$$i(x) = (x, 0)$$
$$f(x, y) = x$$
$$\delta_a(x, y) = (2x, 2y)$$
$$\delta_b(x, y) = (y, x)$$
$$\delta_c(x, y) = (0, 0)$$

A better implementation

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Informally: use one bit for the parity to the number of b's.

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

$$i(x) = (\texttt{even}, x)$$

$$f(\texttt{even}, x) = x$$
$$f(\texttt{odd}, x) = 0$$

Solution in vector spaces

$$Q = \mathbb{R}^2$$

$$i(x) = (x, 0)$$

$$f(x, y) = x$$

$$\delta_a(x, y) = (2x, 2y)$$
$$\delta_b(x, y) = (y, x)$$
$$\delta_c(x, y) = (0, 0)$$

A better implementation

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Informally: use one bit for the parity to the number of b's.

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

$$i(x) = (\texttt{even}, x)$$

$$f(\texttt{even}, x) = x$$
$$f(\texttt{odd}, x) = 0$$

$$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$$
$$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$$

Solution in vector spaces

$$Q = \mathbb{R}^2$$

$$i(x) = (x, 0)$$

$$f(x, y) = x$$

$$\delta_a(x, y) = (2x, 2y)$$
$$\delta_b(x, y) = (y, x)$$
$$\delta_c(x, y) = (0, 0)$$

A better implementation

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Informally: use one bit for the parity to the number of b's.

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

$$i(x) = (\texttt{even}, x)$$

$$f(\texttt{even}, x) = x$$
$$f(\texttt{odd}, x) = 0$$

$$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$$
$$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$$
$$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$$
$$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$$

Solution in vector spaces

$$Q = \mathbb{R}^2$$

$$i(x) = (x, 0)$$

$$f(x, y) = x$$

$$\delta_a(x, y) = (2x, 2y)$$
$$\delta_b(x, y) = (y, x)$$
$$\delta_c(x, y) = (0, 0)$$

A better implementation

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Informally: use one bit for the parity to the number of b's.

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

$$i(x) = (\texttt{even}, x)$$

$$f(\texttt{even}, x) = x$$
$$f(\texttt{odd}, x) = 0$$

$$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$$
$$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$$
$$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$$
$$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$$
$$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$$
$$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$$

Solution in vector spaces

$$Q = \mathbb{R}^2$$

$$i(x) = (x, 0)$$

$$f(x, y) = x$$

$$\delta_a(x, y) = (2x, 2y)$$
$$\delta_b(x, y) = (y, x)$$
$$\delta_c(x, y) = (0, 0)$$

A better implementation

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Informally: use one bit for the parity to the number of b's.

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

$$i(x) = (\texttt{even}, x)$$

$$f(\texttt{even}, x) = x$$
$$f(\texttt{odd}, x) = 0$$

$$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$$
$$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$$
$$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$$
$$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$$
$$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$$
$$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$$

Solution in vector spaces

$$Q = \mathbb{R}^2$$

$$i(x) = (x, 0)$$

$$f(x, y) = x$$

$$\delta_a(x, y) = (2x, 2y)$$
$$\delta_b(x, y) = (y, x)$$
$$\delta_c(x, y) = (0, 0)$$

Why is it a better implementation?

Is there a good notion of such automata?

What are their properties (e.g. minimization) ?

A definition via categories

Automata in a category

Automata in a category

A (C,I,F)-automaton is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is a object of states,

$i\colon I \to Q$ is the initial arrow

$f\colon Q \to F$ is the final arrow

$\delta_a\colon Q \to Q$ is the transition arrow for the letter a.

Automata in a category

A (C,I,F)-automaton is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is a object of states,

$i \colon I \to Q$ is the initial arrow

$f \colon Q \to F$ is the final arrow

$\delta_a \colon Q \to Q$ is the transition arrow
for the letter a.

The (C,I,F)-language computed is:

$$\overbrace{[\![\mathcal{A}]\!] \colon A^* \to [I, F]}$$
$$u \mapsto f \circ \delta_u \circ i$$

Automata in a category

A (C,I,F)-automaton is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is a object of states,

$i : I \to Q$ is the initial arrow

$f : Q \to F$ is the final arrow

$\delta_a : Q \to Q$ is the transition arrow for the letter a.

The (C,I,F)-language computed is:

$$\overbrace{[\![\mathcal{A}]\!] : A^* \to [I, F]}$$

$$u \mapsto f \circ \delta_u \circ i$$

Auto(L) is the category of (C,I,F)-automata for the (C,I,F)-language L.

Automata in a category

A (C,I,F)-automaton is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is a object of states,

$i \colon I \to Q$ is the initial arrow

$f \colon Q \to F$ is the final arrow

$\delta_a \colon Q \to Q$ is the transition arrow for the letter a.

The (C,I,F)-language computed is:

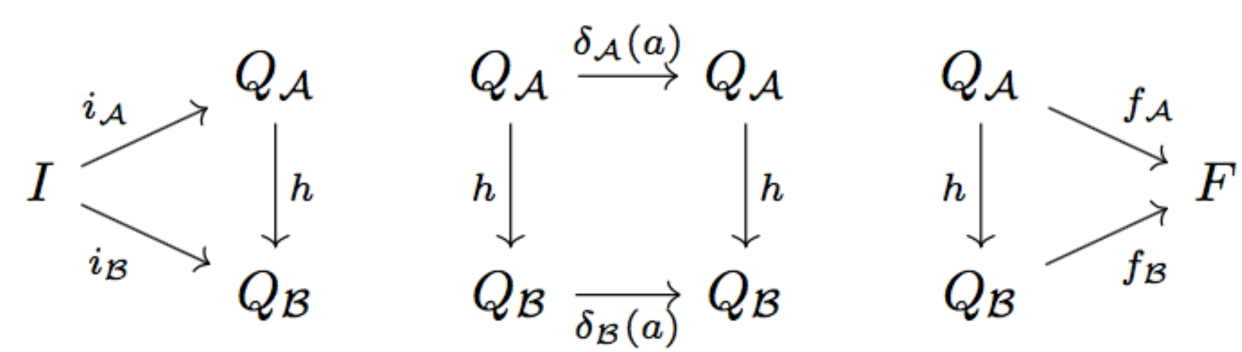
$$[\![\mathcal{A}]\!] \colon A^* \to [I, F]$$
$$u \mapsto f \circ \delta_u \circ i$$

Auto(L) is the category of (C,I,F)-automata for the (C,I,F)-language L.

A morphism is an arrow

$$h \colon Q_{\mathcal{A}} \to Q_{\mathcal{B}}$$

such that tfdc:



Automata in a category

A (C,I,F)-automaton is

$$\langle Q, i, f, (\delta_a)_{a \in A} \rangle$$

where

Q is a object of states,

$i \colon I \to Q$ is the initial arrow

$f \colon Q \to F$ is the final arrow

$\delta_a \colon Q \to Q$ is the transition arrow for the letter a.

The (C,I,F)-language computed is:

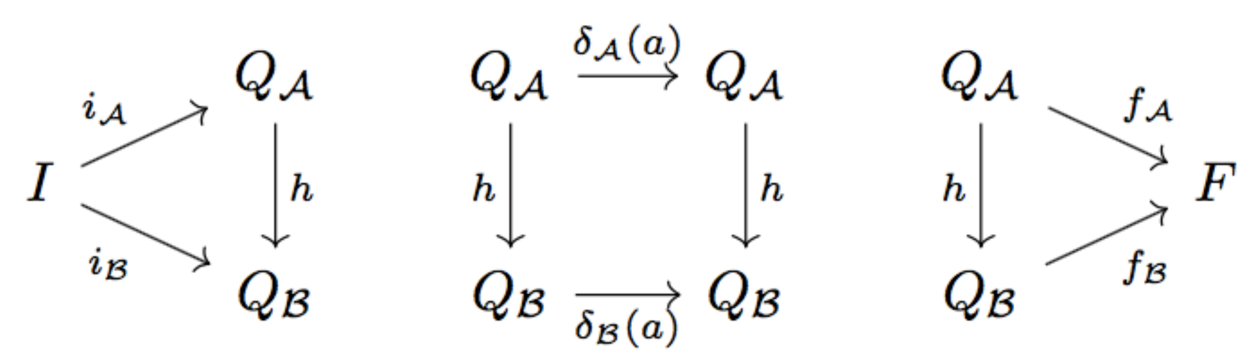
$$\llbracket \mathcal{A} \rrbracket \colon A^* \to [I, F]$$
$$u \mapsto f \circ \delta_u \circ i$$

Auto(L) is the category of (C,I,F)-automata for the (C,I,F)-language L.

A morphism is an arrow

$$h \colon Q_{\mathcal{A}} \to Q_{\mathcal{B}}$$

such that tfdc:



Rk: Morphisms preserve the language.

Automata in a category

A (C,I,F)-automaton is

$$\langle Q, i, f, (\delta_a)_{a\in A}\rangle$$

where

Q is a object of states,

$i\colon I \to Q$ is the initial arrow

$f\colon Q \to F$ is the final arrow

$\delta_a\colon Q \to Q$ is the transition arrow for the letter a.

- (Set,1,2)-automata are deterministic automata
- (Rel,1,1)-automata are non-deterministic automata
- (Vec,K,K)-automata are automata weighted over a field K. (more generally semi-modules)
- …

The (C,I,F)-language computed is:

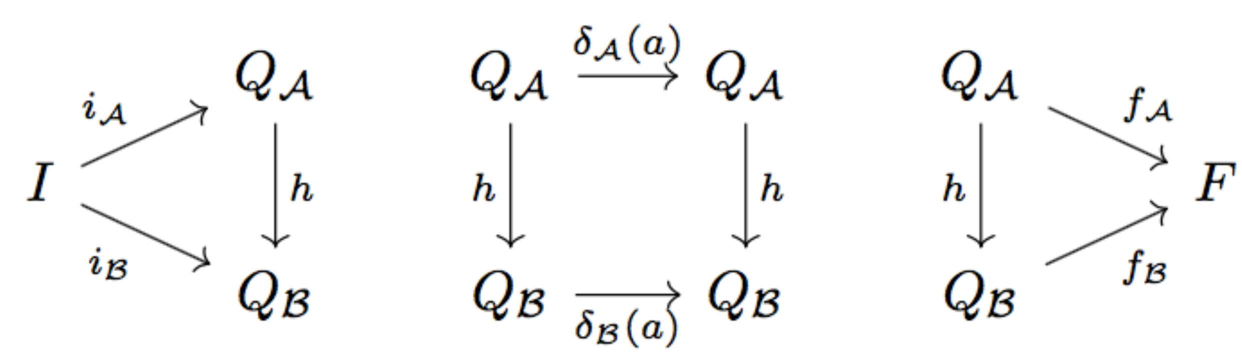
$$\begin{aligned}[\![\mathcal{A}]\!]\colon A^* &\to [I, F]\\ u &\mapsto f\circ\delta_u\circ i\end{aligned}$$

Auto(L) is the category of (C,I,F)-automata for the (C,I,F)-language L.

A morphism is an arrow

$$h\colon Q_{\mathcal{A}} \to Q_{\mathcal{B}}$$

such that tfdc:



Rk: Morphisms preserve the language.

Category of disjoint unions of vector spaces

(free co-product completion of **Vec**)

Category of disjoint unions of vector spaces

(free co-product completion of **Vec**)

A disjoint union of vector space is an ordered pair

$$(I, (V_i)_{i \in I})$$

where I is a set of indices, and V_i is a vector space for all $i \in I$.

Category of disjoint unions of vector spaces

(free co-product completion of **Vec**)

A disjoint union of vector space is an ordered pair

$$(I, (V_i)_{i \in I})$$

where I is a set of indices, and V_i is a vector space for all $i \in I$.

Let Duvs be the category with
- as objects the finite unions of vector spaces
- as arrows the morphisms of finite unions of vector spaces.

Category of disjoint unions of vector spaces

(free co-product completion of **Vec**)

A disjoint union of vector space is an ordered pair

$$(I, (V_i)_{i \in I})$$

where I is a set of indices, and V_i is a vector space for all $i \in I$.

Let Duvs be the category with
- as objects the finite unions of vector spaces
- as arrows the morphisms of finite unions of vector spaces.

A morphism from $(I, (V_i)_{i \in I})$ to $(J, (W_i)_{i \in J})$ is the pair of:

Category of disjoint unions of vector spaces

(free co-product completion of **Vec**)

A disjoint union of vector space is an ordered pair

$$(I, (V_i)_{i \in I})$$

where I is a set of indices, and V_i is a vector space for all $i \in I$.

Let Duvs be the category with
- as objects the finite unions of vector spaces
- as arrows the morphisms of finite unions of vector spaces.

A morphism from $(I, (V_i)_{i \in I})$ to $(J, (W_i)_{i \in J})$ is the pair of:
- a map f from I to J

Category of disjoint unions of vector spaces

(free co-product completion of **Vec**)

A disjoint union of vector space is an ordered pair

$$(I, (V_i)_{i \in I})$$

where I is a set of indices, and V_i is a vector space for all $i \in I$.

Let Duvs be the category with
- as objects the finite unions of vector spaces
- as arrows the morphisms of finite unions of vector spaces.

A morphism from $(I, (V_i)_{i \in I})$ to $(J, (W_i)_{i \in J})$ is the pair of:
- a map f from I to J
- a linear map g_i from V_i to $W_{f(i)}$ for all $i \in I$.

Category of disjoint unions of vector spaces

(free co-product completion of **Vec**)

A disjoint union of vector space is an ordered pair

$$(I, (V_i)_{i \in I})$$

where I is a set of indices, and V_i is a vector space for all $i \in I$.

Let Duvs be the category with
- as objects the finite unions of vector spaces
- as arrows the morphisms of finite unions of vector spaces.

A morphism from $(I, (V_i)_{i \in I})$ to $(J, (W_i)_{i \in J})$ is the pair of:
- a map f from I to J
- a linear map g_i from V_i to $W_{f(i)}$ for all $i \in I$.

Remark: Vec is a subcategory of Duvs.

Duvs-automata

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

$$i(x) = (\texttt{even}, x)$$

$$f(\texttt{even}, x) = x$$
$$f(\texttt{odd}, x) = 0$$

$$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$$
$$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$$
$$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$$
$$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$$
$$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$$
$$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$$

Duvs-automata

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Indices = $\{\texttt{odd}, \texttt{even}\}$

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

$$i(x) = (\texttt{even}, x)$$

$$f(\texttt{even}, x) = x$$
$$f(\texttt{odd}, x) = 0$$

$$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$$
$$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$$
$$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$$
$$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$$
$$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$$
$$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$$

Duvs-automata

$$L_{\mathtt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Indices = $\{\mathtt{odd}, \mathtt{even}\}$

$Q = (\{\mathtt{odd}\} \times \mathbb{R}) \cup (\{\mathtt{even}\} \times \mathbb{R})$

$V_{\mathtt{odd}} = V_{\mathtt{even}} = \mathbb{R}$

$i(x) = (\mathtt{even}, x)$

$f(\mathtt{even}, x) = x$
$f(\mathtt{odd}, x) = 0$

$\delta_a(\mathtt{even}, x) = (\mathtt{even}, 2x)$
$\delta_a(\mathtt{odd}, x) = (\mathtt{odd}, 2x)$
$\delta_b(\mathtt{even}, x) = (\mathtt{odd}, x)$
$\delta_b(\mathtt{odd}, x) = (\mathtt{even}, x)$
$\delta_c(\mathtt{even}, x) = (\mathtt{even}, 0)$
$\delta_c(\mathtt{odd}, x) = (\mathtt{odd}, 0)$

Duvs-automata

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Indices = $\{\texttt{odd}, \texttt{even}\}$

$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$

$V_{\texttt{odd}} = V_{\texttt{even}} = \mathbb{R}$

$i(x) = (\texttt{even}, x)$

$f(\texttt{even}, x) = x$
$f(\texttt{odd}, x) = 0$

Is it minimal ?

$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$
$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$
$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$
$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$
$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$
$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$

Duvs-automata

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Indices = $\{\texttt{odd}, \texttt{even}\}$

$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$

$V_{\texttt{odd}} = V_{\texttt{even}} = \mathbb{R}$

$i(x) = (\texttt{even}, x)$

$f(\texttt{even}, x) = x$
$f(\texttt{odd}, x) = 0$

Is it minimal ? No…

$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$
$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$
$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$
$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$
$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$
$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$

Duvs-automata

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Indices = $\{\texttt{odd}, \texttt{even}\}$

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

$V_{\texttt{odd}} = V_{\texttt{even}} = \mathbb{R}$

$$i(x) = (\texttt{even}, x)$$

$$f(\texttt{even}, x) = x$$
$$f(\texttt{odd}, x) = 0$$

$$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$$
$$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$$
$$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$$
$$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$$
$$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$$
$$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$$

Is it minimal ? No…

$(\texttt{odd}, 0)$ and $(\texttt{even}, 0)$ are observationally equivalent

Duvs-automata

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

Indices = $\{\texttt{odd}, \texttt{even}\}$

$V_{\texttt{odd}} = V_{\texttt{even}} = \mathbb{R}$

$$i(x) = (\texttt{even}, x)$$

$$f(\texttt{even}, x) = x$$
$$f(\texttt{odd}, x) = 0$$

$$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$$
$$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$$
$$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$$
$$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$$
$$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$$
$$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$$

Is it minimal ? No…

$(\texttt{odd}, 0)$ and $(\texttt{even}, 0)$ are observationally equivalent

But the implementation is arbitrary.

Duvs-automata

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

Indices = $\{\texttt{odd}, \texttt{even}\}$

$V_{\texttt{odd}} = V_{\texttt{even}} = \mathbb{R}$

$$i(x) = (\texttt{even}, x)$$

$$f(\texttt{even}, x) = x$$
$$f(\texttt{odd}, x) = 0$$

$$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$$
$$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$$
$$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$$
$$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$$
$$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$$
$$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$$

Is it minimal ? No…

$(\texttt{odd}, 0)$ and $(\texttt{even}, 0)$ are observationally equivalent

But the implementation is arbitrary.

Can it be made minimal?

Duvs-automata

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Indices = $\{\texttt{odd}, \texttt{even}\}$

$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$

$V_{\texttt{odd}} = V_{\texttt{even}} = \mathbb{R}$

$i(x) = (\texttt{even}, x)$

$f(\texttt{even}, x) = x$
$f(\texttt{odd}, x) = 0$

$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$
$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$
$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$
$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$
$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$
$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$

Is it minimal ? No…
$(\texttt{odd}, 0)$ and $(\texttt{even}, 0)$ are observationally equivalent
But the implementation is arbitrary.

Can it be made minimal? No…

Duvs-automata

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Indices = $\{\texttt{odd}, \texttt{even}\}$

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

$V_{\texttt{odd}} = V_{\texttt{even}} = \mathbb{R}$

$$i(x) = (\texttt{even}, x)$$

$$f(\texttt{even}, x) = x$$
$$f(\texttt{odd}, x) = 0$$

$$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$$
$$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$$
$$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$$
$$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$$
$$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$$
$$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$$

Is it minimal ? No…
$(\texttt{odd}, 0)$ and $(\texttt{even}, 0)$ are observationally equivalent
But the implementation is arbitrary.

Can it be made minimal? No…
Well, in fact Yes… but would be larger…

Duvs-automata

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

Indices = $\{\texttt{odd}, \texttt{even}\}$

$V_{\texttt{odd}} = V_{\texttt{even}} = \mathbb{R}$

$$i(x) = (\texttt{even}, x)$$

$$f(\texttt{even}, x) = x$$
$$f(\texttt{odd}, x) = 0$$

$$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$$
$$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$$
$$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$$
$$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$$
$$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$$
$$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$$

Is it minimal ? No…
$(\texttt{odd}, 0)$ and $(\texttt{even}, 0)$ are observationally equivalent
But the implementation is arbitrary.

Can it be made minimal? No…
Well, in fact Yes… but would be larger…
What can be done?

Minimizing automata via categories

Ingredients for the existence of a minimal automaton

Questions:

Given a (C,I,F)-automaton,

- what does it mean to be minimal?
- at what condition there exists a minimal automaton for a language?
- what do we need to effectively compute it?

Ingredients for the existence of a minimal automaton

Questions:

Given a (C,I,F)-automaton,

- what does it mean to be minimal?
- at what condition there exists a minimal automaton for a language?
- what do we need to effectively compute it?

Minimal? « A DFA is minimal if it divides any other automaton for the same language. »

Ingredients for the existence of a minimal automaton

Questions:

Given a (C,I,F)-automaton,

- what does it mean to be minimal?
- at what condition there exists a minimal automaton for a language?
- what do we need to effectively compute it?

Minimal? « A DFA is minimal if it divides any other automaton for the same language. »

it is the quotient of a subautomaton.

Ingredients for the existence of a minimal automaton

Questions:

Given a (C,I,F)-automaton,

- what does it mean to be minimal?
- at what condition there exists a minimal automaton for a language?
- what do we need to effectively compute it?

Minimal? « A DFA is minimal if it divides any other automaton for the same language. »

it is the quotient of a subautomaton.

notion of « surjection » notion of « injection »

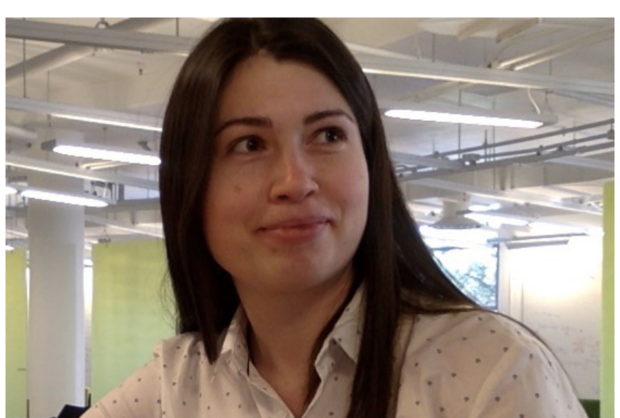
Ingredients for the existence of a minimal automaton

Questions:
Given a (C,I,F)-automaton,
- what does it mean to be minimal?
- at what condition there exists a minimal automaton for a language?
- what do we need to effectively compute it?

Minimal? « A DFA is minimal if it divides any other automaton for the same language. »

it is the quotient of a subautomaton.

notion of « surjection » (quotient) — notion of « injection » (subautomaton)



It suffices to have
1. an initial automaton
2. a final automaton
3. a factorization system

Factorization systems

A pair of families of arrows $(\mathcal{E}, \mathcal{M})$ is a factorization system if:

Factorization systems

« epimorphisms »
« surjections »

« monomorphisms »
« injections »

A pair of families of arrows $(\mathcal{E}, \mathcal{M})$ is a **factorization system** if:

Factorization systems

« epimorphisms »
« surjections »

« monomorphisms »
« injections »

A pair of families of arrows $(\mathcal{E}, \mathcal{M})$ is a **factorization system** if:

- arrows in $\mathcal{E}$ are closed under composition
- arrows in $\mathcal{M}$ are closed under composition

Factorization systems

« epimorphisms » « monomorphisms »
« surjections » « injections »

A pair of families of arrows $(\mathcal{E}, \mathcal{M})$ is a factorization system if:

- arrows in $\mathcal{E}$ are closed under composition
- arrows in $\mathcal{M}$ are closed under composition
- arrows that are both in $\mathcal{E}$ and in $\mathcal{M}$ are isomorphisms,

Factorization systems

« epimorphisms » « monomorphisms »

« surjections » « injections »

A pair of families of arrows $(\mathcal{E}, \mathcal{M})$ is a factorization system if:

- arrows in $\mathcal{E}$ are closed under composition
- arrows in $\mathcal{M}$ are closed under composition
- arrows that are both in $\mathcal{E}$ and in $\mathcal{M}$ are isomorphisms,
- all arrows $f\colon X \to Y$ can be written

$$f = m \circ e$$

for some $e\colon X \to Z$ in $\mathcal{E}$ and $m\colon Z \to Y$ in $\mathcal{M}$.

Factorization systems

« epimorphisms » « monomorphisms »
« surjections » « injections »

A pair of families of arrows $(\mathcal{E}, \mathcal{M})$ is a factorization system if:

- arrows in $\mathcal{E}$ are closed under composition
- arrows in $\mathcal{M}$ are closed under composition
- arrows that are both in $\mathcal{E}$ and in $\mathcal{M}$ are isomorphisms,
- all arrows $f : X \to Y$ can be written

$$f = m \circ e$$

for some $e\colon X \to Z$ in $\mathcal{E}$ and $m\colon Z \to Y$ in $\mathcal{M}$.

the factorization of f .

Factorization systems

« epimorphisms »
« surjections »

« monomorphisms »
« injections »

A pair of families of arrows $(\mathcal{E}, \mathcal{M})$ is a factorization system if:

- arrows in $\mathcal{E}$ are closed under composition
- arrows in $\mathcal{M}$ are closed under composition
- arrows that are both in $\mathcal{E}$ and in $\mathcal{M}$ are isomorphisms,
- all arrows $f : X \to Y$ can be written

$$f = m \circ e$$

the factorization of f.

for some $e : X \to Z$ in $\mathcal{E}$ and $m : Z \to Y$ in $\mathcal{M}$.

- furthermore, this decomposition is unique up to isomorphism (it has in fact the stronger « diagonal property »).

Factorization systems

« epimorphisms » « surjections »

« monomorphisms » « injections »

A pair of families of arrows $(\mathcal{E}, \mathcal{M})$ is a factorization system if:

- arrows in $\mathcal{E}$ are closed under composition
- arrows in $\mathcal{M}$ are closed under composition
- arrows that are both in $\mathcal{E}$ and in $\mathcal{M}$ are isomorphisms,
- all arrows $f\colon X \to Y$ can be written

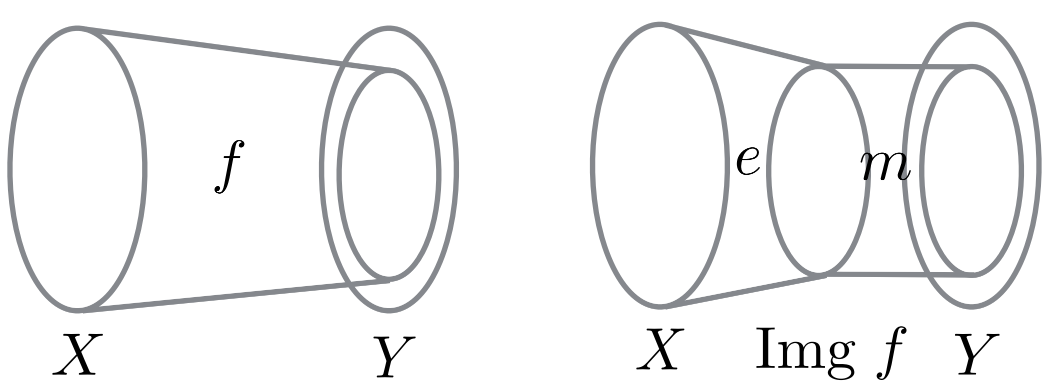
$$f = m \circ e$$

the factorization of f.

for some $e\colon X \to Z$ in $\mathcal{E}$ and $m\colon Z \to Y$ in $\mathcal{M}$.

- furthermore, this decomposition is unique up to isomorphism (it has in fact the stronger « diagonal property »).

In Set:



Factorization systems

« epimorphisms » « surjections »

« monomorphisms » « injections »

A pair of families of arrows $(\mathcal{E}, \mathcal{M})$ is a factorization system if:

- arrows in $\mathcal{E}$ are closed under composition
- arrows in $\mathcal{M}$ are closed under composition
- arrows that are both in $\mathcal{E}$ and in $\mathcal{M}$ are isomorphisms,
- all arrows $f : X \to Y$ can be written

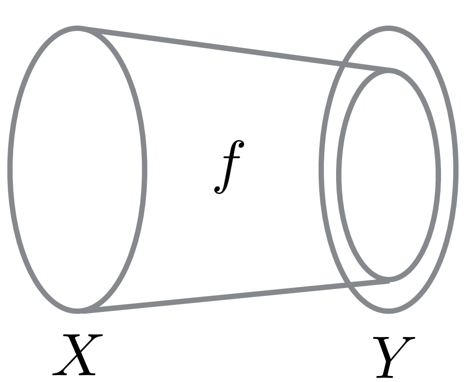
$$f = m \circ e$$

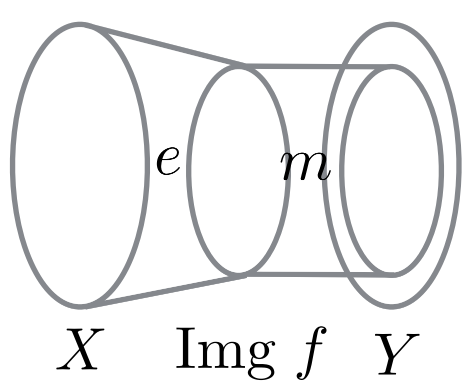
the factorization of f.

for some $e : X \to Z$ in $\mathcal{E}$ and $m : Z \to Y$ in $\mathcal{M}$.

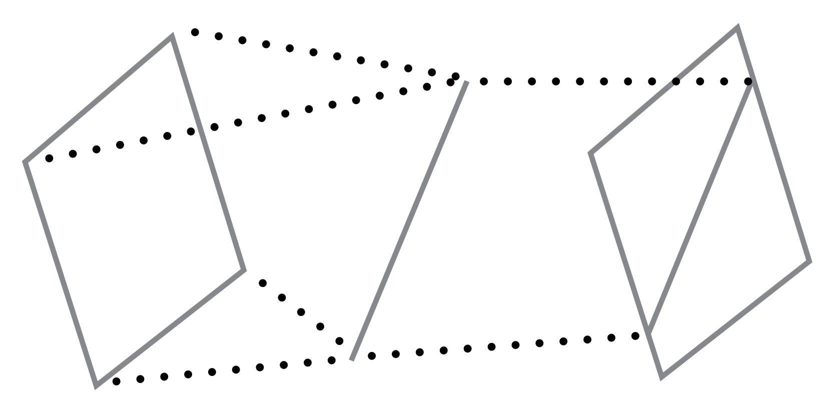
- furthermore, this decomposition is unique up to isomorphism (it has in fact the stronger « diagonal property »).

In Set:





In Vec:



Factorization systems

« epimorphisms » « monomorphisms »

« surjections » « injections »

A pair of families of arrows $(\mathcal{E}, \mathcal{M})$ is a factorization system if:

- arrows in $\mathcal{E}$ are closed under composition
- arrows in $\mathcal{M}$ are closed under composition
- arrows that are both in $\mathcal{E}$ and in $\mathcal{M}$ are isomorphisms,
- all arrows $f \colon X \to Y$ can be written

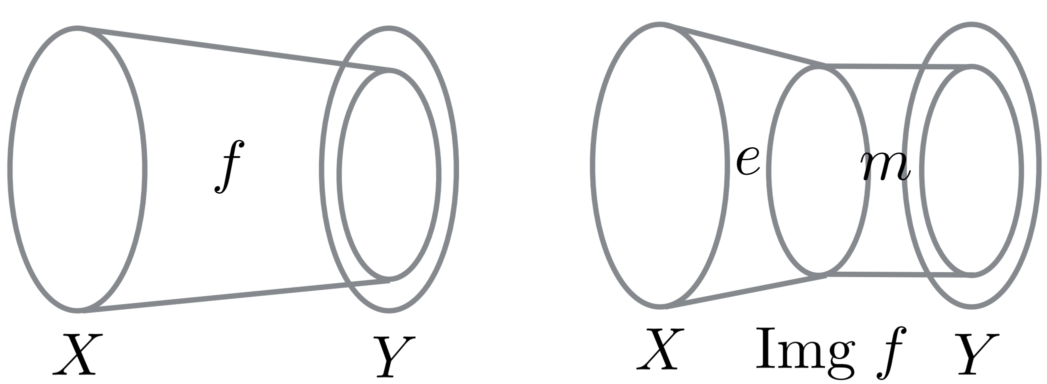
$$f = m \circ e$$

the factorization of f.

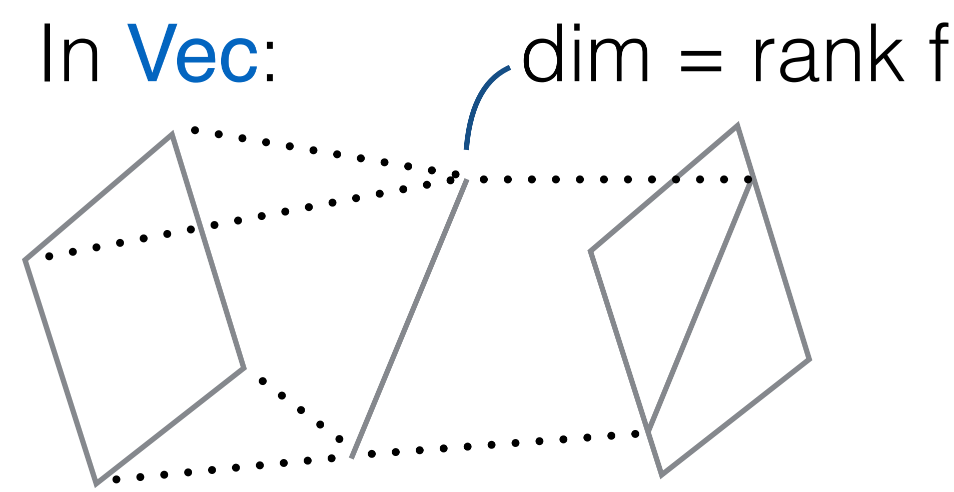
for some $e \colon X \to Z$ in $\mathcal{E}$ and $m \colon Z \to Y$ in $\mathcal{M}$.

- furthermore, this decomposition is unique up to isomorphism (it has in fact the stronger « diagonal property »).

In Set:



In Vec:



Factorization system for automata

Factorization system for automata

Lemma: If there is a factorization system $(\mathcal{E}, \mathcal{M})$ in a category $\mathcal{C}$ then it can be lifted to the category of $\mathcal{C}$-automata for a language: these automata morphisms that belong to $\mathcal{E}$ (resp. $\mathcal{M}$) as arrows in $\mathcal{C}$.

Factorization system for automata

Lemma: If there is a factorization system $(\mathcal{E}, \mathcal{M})$ in a category $\mathcal{C}$ then it can be lifted to the category of $\mathcal{C}$-automata for a language: these automata morphisms that belong to $\mathcal{E}$ (resp. $\mathcal{M}$) as arrows in $\mathcal{C}$.

Hence (Set,1,2)-automata (i.e. DFA) have a factorization system (surjective morphisms,injective morphisms).

Factorization system for automata

Lemma: If there is a factorization system $(\mathcal{E}, \mathcal{M})$ in a category $\mathcal{C}$ then it can be lifted to the category of $\mathcal{C}$-automata for a language: these automata morphisms that belong to $\mathcal{E}$ (resp. $\mathcal{M}$) as arrows in $\mathcal{C}$.

Hence (Set,1,2)-automata (i.e. DFA) have a factorization system (surjective morphisms,injective morphisms).

Similarly (Vec,K,K)-automata (i.e., automata weighted over a field) possess factorization system (surjective morphisms,injective morphisms).

Factorization system for automata

Lemma: If there is a factorization system $(\mathcal{E}, \mathcal{M})$ in a category $\mathcal{C}$ then it can be lifted to the category of $\mathcal{C}$-automata for a language: these automata morphisms that belong to $\mathcal{E}$ (resp. $\mathcal{M}$) as arrows in $\mathcal{C}$.

Hence (Set,1,2)-automata (i.e. DFA) have a factorization system (surjective morphisms,injective morphisms).

Similarly (Vec,K,K)-automata (i.e., automata weighted over a field) possess factorization system (surjective morphisms,injective morphisms).

Definition:

- an $\mathcal{M}$-subobject X of Y is such that there is an $\mathcal{M}$-arrow $m\colon X \to Y$,
- an $\mathcal{E}$-quotient X of Y is such that there is an $\mathcal{E}$-arrow $e\colon Y \to X$,
- X $(\mathcal{E}, \mathcal{M})$-divides Y if it is a $\mathcal{E}$-quotient of an $\mathcal{M}$-subobject of Y.

Minimization !

Minimization !

Lemma: In a category with initial object, final object, and a factorization system $(\mathcal{E},\mathcal{M})$ then:

- there exists an object $\mathtt{Min}$ that $(\mathcal{E},\mathcal{M})$-divides all objects,
- furthermore $\mathtt{Min} \approx \mathtt{Obs}(\mathtt{Reach}(X)) \approx \mathtt{Reach}(\mathtt{Obs}(X))$ for all X,

where

- $\mathtt{Reach}(X)$ is the factorization of the only arrow from I to X, and
- $\mathtt{Obs}(X)$ is the factorization of the only arrow from X to F.

Minimization !

Lemma: In a category with initial object, final object, and a factorization system $(\mathcal{E}, \mathcal{M})$ then:

- there exists an object $\mathtt{Min}$ that $(\mathcal{E}, \mathcal{M})$-divides all objects,
- furthermore $\mathtt{Min} \approx \mathtt{Obs}(\mathtt{Reach}(X)) \approx \mathtt{Reach}(\mathtt{Obs}(X))$ for all X,

where

- $\mathtt{Reach}(X)$ is the factorization of the only arrow from I to X, and
- $\mathtt{Obs}(X)$ is the factorization of the only arrow from X to F.

Proof: $\mathtt{Min}$ is the factorization of the only arrow from I to F. And…

Minimization !

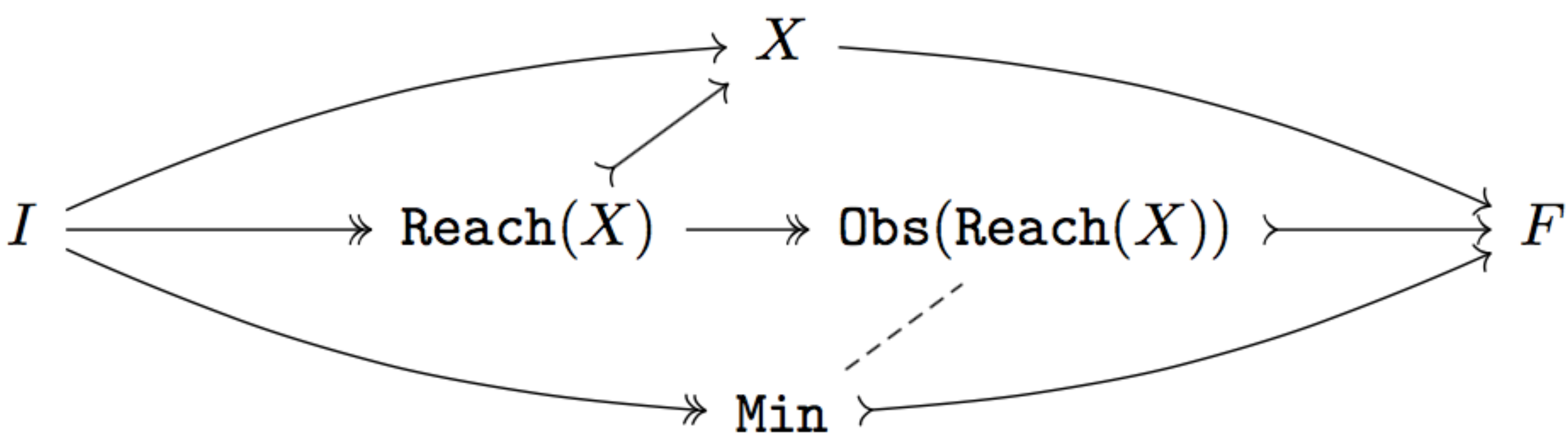
Lemma: In a category with initial object, final object, and a factorization system $(\mathcal{E}, \mathcal{M})$ then:

- there exists an object $\texttt{Min}$ that $(\mathcal{E}, \mathcal{M})$-divides all objects,
- furthermore $\texttt{Min} \approx \texttt{Obs}(\texttt{Reach}(X)) \approx \texttt{Reach}(\texttt{Obs}(X))$ for all X,

where

- $\texttt{Reach}(X)$ is the factorization of the only arrow from I to X, and
- $\texttt{Obs}(X)$ is the factorization of the only arrow from X to F.

Proof: $\texttt{Min}$ is the factorization of the only arrow from I to F. And…



At this point…

We know that:

- C-automata and C-languages can be defined generally in a category C, yielding a

  category Auto(L) of « C-automata for the language L »

- for having a minimal object in a category, it is sufficient to have:
  1) an initial and a final object in the category for the language,
  2) a factorization system in C,
- that the existence of initial and final automata arise from simple assumptions on C
- that the factorization system for automata is inherited from C,
- that standard minimization for DFA and field weighted automata are obtained this way.

At this point…

We know that:

- C-automata and C-languages can be defined generally in a category C, yielding a

  category Auto(L) of « C-automata for the language L »

- for having a minimal object in a category, it is sufficient to have:
  1) an initial and a final object in the category for the language,
  2) a factorization system in C,
- that the existence of initial and final automata arise from simple assumptions on C
- that the factorization system for automata is inherited from C,
- that standard minimization for DFA and field weighted automata are obtained this way.

But, what about minimizing duvs-automata?

Minimization of Duvs automata (wrong version)

Minimization of Duvs automata is possible (all the ingredient are there).

Minimization of Duvs automata (wrong version)

Minimization of Duvs automata is possible (all the ingredient are there).

However, for the definition of factorization system that works (epi,mono), the minimal automaton for

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

has state space

$$Q = \mathbb{R}^2$$

and not

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

Glueings

Glueings

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Glueings

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

Vec-automaton

$Q = \mathbb{R}^2$

$i(x) = (x, 0)$

$f(x, y) = x$

$\delta_a(x, y) = (2x, 2y)$
$\delta_b(x, y) = (y, x)$
$\delta_c(x, y) = (0, 0)$

Glueings

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

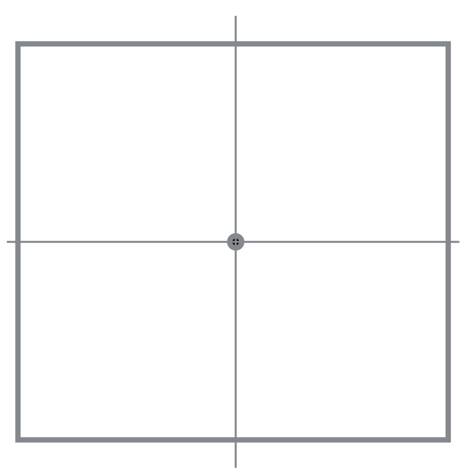
Vec-automaton

$Q = \mathbb{R}^2$

$i(x) = (x, 0)$

$f(x, y) = x$

$\delta_a(x, y) = (2x, 2y)$
$\delta_b(x, y) = (y, x)$
$\delta_c(x, y) = (0, 0)$



Glueings

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

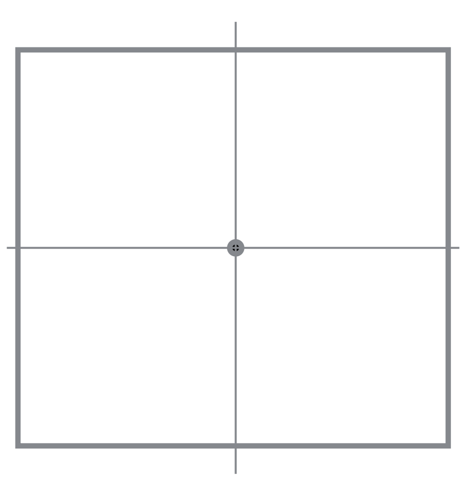
Vec-automaton

$Q = \mathbb{R}^2$

$i(x) = (x, 0)$

$f(x, y) = x$

$\delta_a(x, y) = (2x, 2y)$
$\delta_b(x, y) = (y, x)$
$\delta_c(x, y) = (0, 0)$



Duvs-automaton

$Q = \{\texttt{odd}, \texttt{even}\} \times \mathbb{R}$

$i(x) = (\texttt{even}, x)$

$f(\texttt{even}, x) = x$
$f(\texttt{odd}, x) = 0$

$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$
$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$

$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$
$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$

$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$
$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$

Glueings

$$L_{\mathtt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

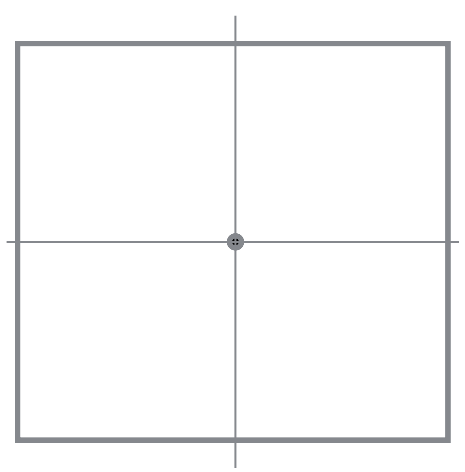
Vec-automaton

$Q = \mathbb{R}^2$

$i(x) = (x, 0)$

$f(x, y) = x$

$\delta_a(x, y) = (2x, 2y)$
$\delta_b(x, y) = (y, x)$
$\delta_c(x, y) = (0, 0)$



Duvs-automaton

$Q = \{\mathtt{odd}, \mathtt{even}\} \times \mathbb{R}$

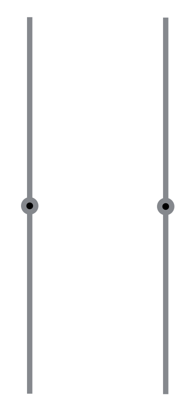
$i(x) = (\mathtt{even}, x)$

$f(\mathtt{even}, x) = x$
$f(\mathtt{odd}, x) = 0$

$\delta_a(\mathtt{even}, x) = (\mathtt{even}, 2x)$
$\delta_a(\mathtt{odd}, x) = (\mathtt{odd}, 2x)$

$\delta_b(\mathtt{even}, x) = (\mathtt{odd}, x)$
$\delta_b(\mathtt{odd}, x) = (\mathtt{even}, x)$

$\delta_c(\mathtt{even}, x) = (\mathtt{even}, 0)$
$\delta_c(\mathtt{odd}, x) = (\mathtt{odd}, 0)$



Glueings

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

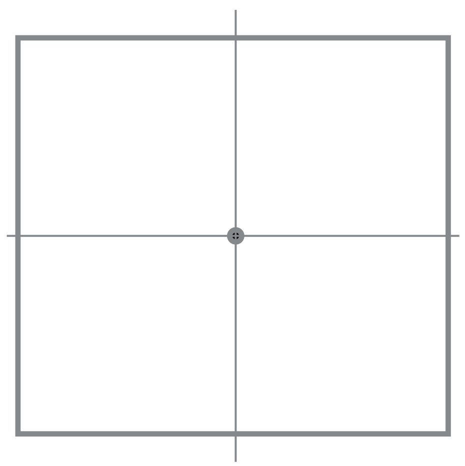
Vec-automaton

$Q = \mathbb{R}^2$

$i(x) = (x, 0)$

$f(x, y) = x$

$\delta_a(x, y) = (2x, 2y)$
$\delta_b(x, y) = (y, x)$
$\delta_c(x, y) = (0, 0)$



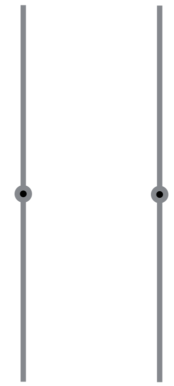
Duvs-automaton

$Q = \{\texttt{odd}, \texttt{even}\} \times \mathbb{R}$

$i(x) = (\texttt{even}, x)$

$f(\texttt{even}, x) = x$
$f(\texttt{odd}, x) = 0$

$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$
$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$
$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$
$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$
$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$
$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$



Glue(Vec)-automaton

Glueings

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

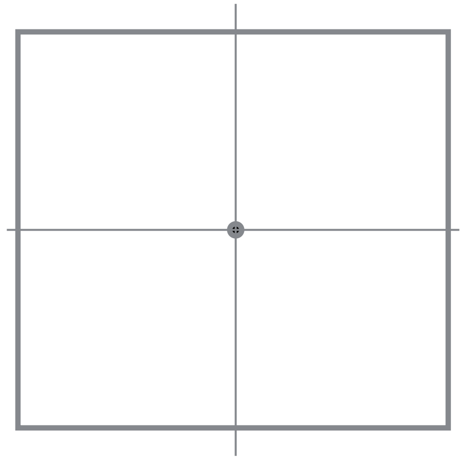
Vec-automaton

$Q = \mathbb{R}^2$

$i(x) = (x, 0)$

$f(x, y) = x$

$\delta_a(x, y) = (2x, 2y)$
$\delta_b(x, y) = (y, x)$
$\delta_c(x, y) = (0, 0)$



Duvs-automaton

$Q = \{\texttt{odd}, \texttt{even}\} \times \mathbb{R}$

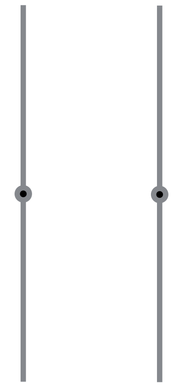
$i(x) = (\texttt{even}, x)$

$f(\texttt{even}, x) = x$
$f(\texttt{odd}, x) = 0$

$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$
$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$

$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$
$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$

$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$
$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$



Glue(Vec)-automaton

Glueings

$$L_{\texttt{Vec}}(u) = \begin{cases} 2^{|u|_a} & \text{if } |u|_b \text{ is even, and } |u|_c = 0 \\ 0 & \text{otherwise} \end{cases}$$

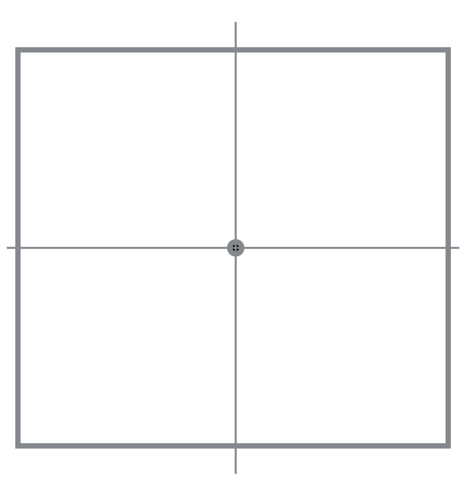
Vec-automaton

$$Q = \mathbb{R}^2$$

$$i(x) = (x, 0)$$

$$f(x, y) = x$$

$$\delta_a(x, y) = (2x, 2y)$$
$$\delta_b(x, y) = (y, x)$$
$$\delta_c(x, y) = (0, 0)$$



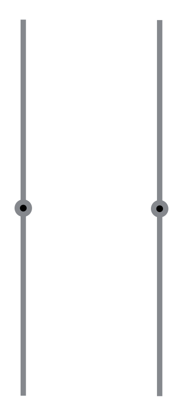
Duvs-automaton

$$Q = \{\texttt{odd}, \texttt{even}\} \times \mathbb{R}$$

$$i(x) = (\texttt{even}, x)$$

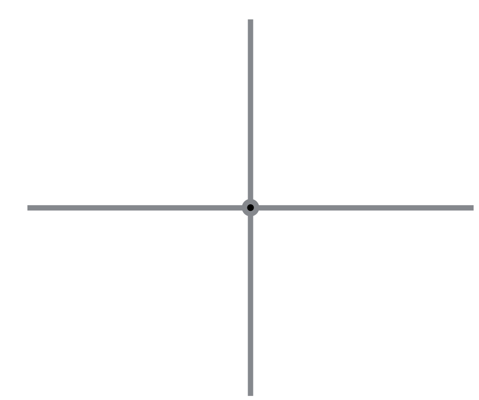
$$f(\texttt{even}, x) = x$$
$$f(\texttt{odd}, x) = 0$$

$$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$$
$$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$$
$$\delta_b(\texttt{even}, x) = (\texttt{odd}, x)$$
$$\delta_b(\texttt{odd}, x) = (\texttt{even}, x)$$
$$\delta_c(\texttt{even}, x) = (\texttt{even}, 0)$$
$$\delta_c(\texttt{odd}, x) = (\texttt{odd}, 0)$$



Glue(Vec)-automaton

?



Defining Glue(Vec)

Defining Glue(Vec)

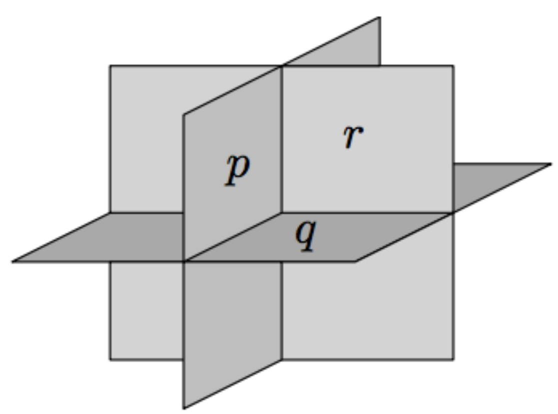
A glueing of vector space is

- a disjoint union of vector spaces
- together with an equivalence relation which:
  1) is trivial over each base space
  2) defines linear bijections between subspaces when restricted to pairs of base spaces.

Defining Glue(Vec)

A glueing of vector space is
- a disjoint union of vector spaces
- together with an equivalence relation which:
  1) is trivial over each base space
  2) defines linear bijections between subspaces when restricted to pairs of base spaces.

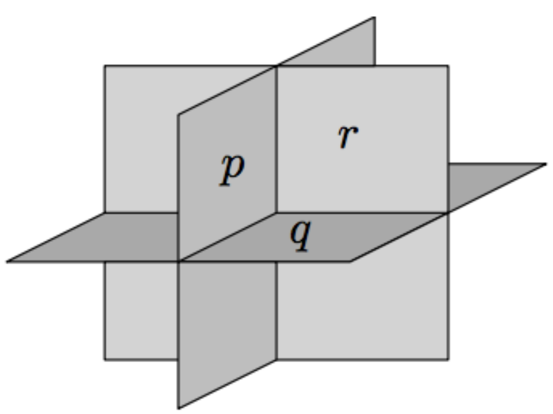
$$(p, x, 0) \sim_{\text{glue}} (q, 0, x)$$
$$(q, x, 0) \sim_{\text{glue}} (r, 0, x)$$
$$(r, x, 0) \sim_{\text{glue}} (p, 0, x)$$



Defining Glue(Vec)

A glueing of vector space is
- a disjoint union of vector spaces
- together with an equivalence relation which:
  1) is trivial over each base space
  2) defines linear bijections between subspaces when restricted to pairs of base spaces.

$$(p, x, 0) \sim_{\text{glue}} (q, 0, x)$$
$$(q, x, 0) \sim_{\text{glue}} (r, 0, x)$$
$$(r, x, 0) \sim_{\text{glue}} (p, 0, x)$$

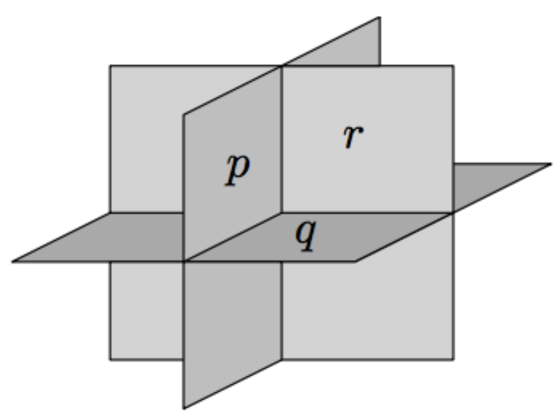


Morphisms are…
complicated to describe…

Defining Glue(Vec)

A glueing of vector space is
- a disjoint union of vector spaces
- together with an equivalence relation which:
  1) is trivial over each base space
  2) defines linear bijections between subspaces when restricted to pairs of base spaces.

$$(p, x, 0) \sim_{\text{glue}} (q, 0, x)$$
$$(q, x, 0) \sim_{\text{glue}} (r, 0, x)$$
$$(r, x, 0) \sim_{\text{glue}} (p, 0, x)$$



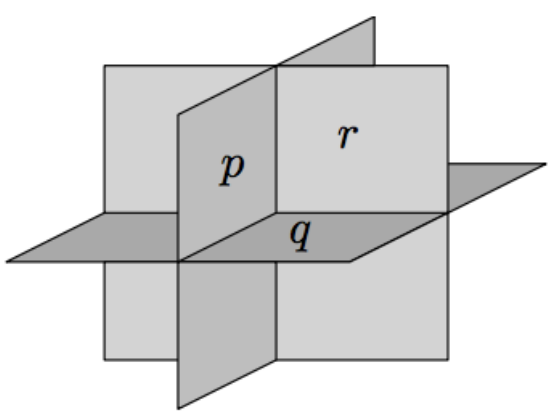
Morphisms are…
complicated to describe…

Aggregating objects from a category is a well known task in category theory: this is obtained by freely adding colimits.

Defining Glue(Vec)

A glueing of vector space is
- a disjoint union of vector spaces
- together with an equivalence relation which:
  1) is trivial over each base space
  2) defines linear bijections between subspaces when restricted to pairs of base spaces.

$$(p, x, 0) \sim_{\text{glue}} (q, 0, x)$$
$$(q, x, 0) \sim_{\text{glue}} (r, 0, x)$$
$$(r, x, 0) \sim_{\text{glue}} (p, 0, x)$$



Morphisms are…
complicated to describe…

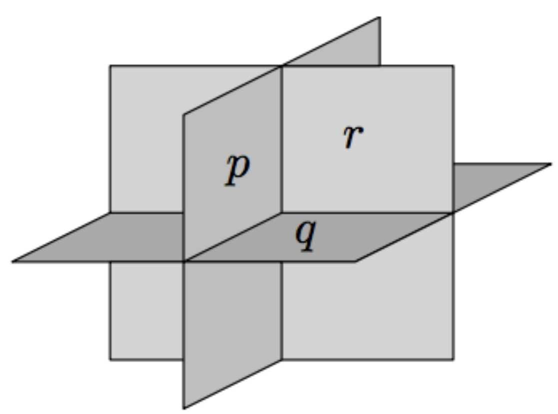
Aggregating objects from a category is a well known task in category theory: this is obtained by freely adding colimits.

The category of glueings of vector spaces is the restriction of the co-completion of Vec to some specific colimits: mono-colimits.

Defining Glue(Vec)

A glueing of vector space is
- a disjoint union of vector spaces
- together with an equivalence relation which:
  1) is trivial over each base space
  2) defines linear bijections between subspaces when restricted to pairs of base spaces.

$$(p, x, 0) \sim_{\text{glue}} (q, 0, x)$$
$$(q, x, 0) \sim_{\text{glue}} (r, 0, x)$$
$$(r, x, 0) \sim_{\text{glue}} (p, 0, x)$$



Morphisms are…
complicated to describe…

Aggregating objects from a category is a well known task in category theory: this is obtained by freely adding colimits.

The category of glueings of vector spaces is the restriction of the co-completion of Vec to some specific colimits: mono-colimits.

The advantage is that the concepts are well known, definition properly stated, and this can be applied to other categories than Vec.

Defining **Glue(Vec)** in categorical terms

Consider a category that already has colimits (for instance Vec)

Defining **Glue(Vec)** in categorical terms

Consider a category that already has colimits (for instance Vec)

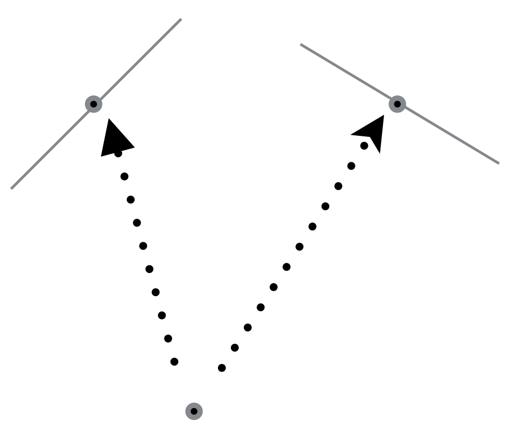
A mono-co-limit diagram is a diagram such that the universal cocone consists only of monos.

Defining **Glue(Vec)** in categorical terms

Consider a category that already has colimits (for instance Vec)

A mono-co-limit diagram is a diagram such that the universal cocone consists only of monos.

For instance in Vec/Set:

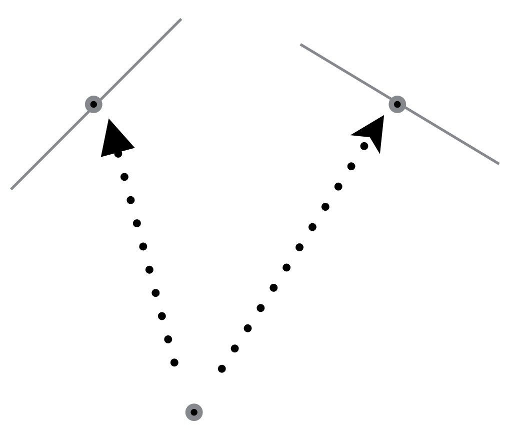


Defining **Glue(Vec)** in categorical terms

Consider a category that already has colimits (for instance Vec)

A mono-co-limit diagram is a diagram such that the universal cocone consists only of monos.

For instance in Vec/Set:



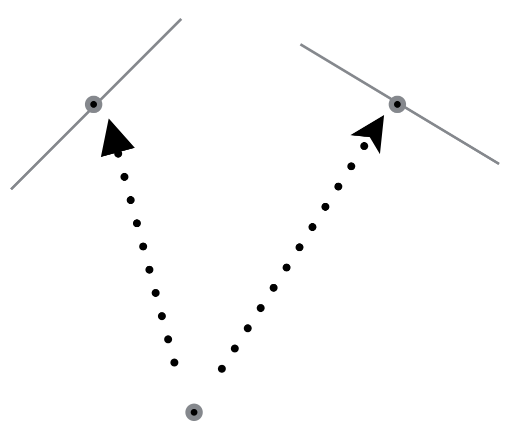
Yes!

Defining **Glue(Vec)** in categorical terms

Consider a category that already has colimits (for instance Vec)

A mono-co-limit diagram is a diagram such that the universal cocone consists only of monos.

For instance in Vec/Set:



Yes!

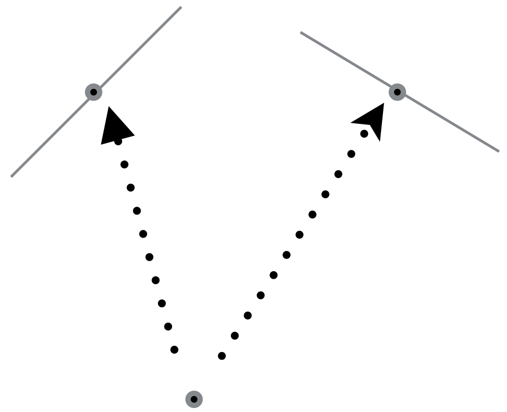
coproducts
are
mono-colimits

Defining **Glue(Vec)** in categorical terms

Consider a category that already has colimits (for instance Vec)

A mono-co-limit diagram is a diagram such that the universal cocone consists only of monos.

For instance in Vec/Set:



Yes!

coproducts
are
mono-colimits

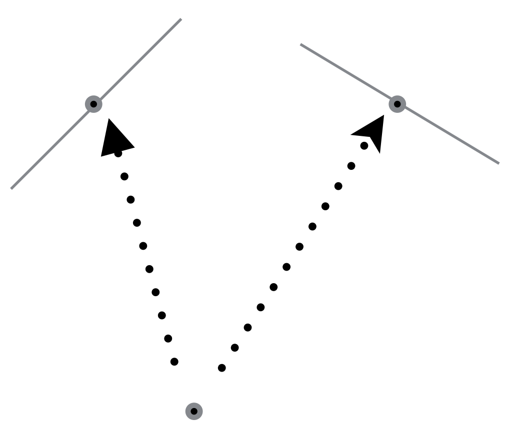
Yes!

Defining **Glue(Vec)** in categorical terms

Consider a category that already has colimits (for instance Vec)

A mono-co-limit diagram is a diagram such that the universal cocone consists only of monos.

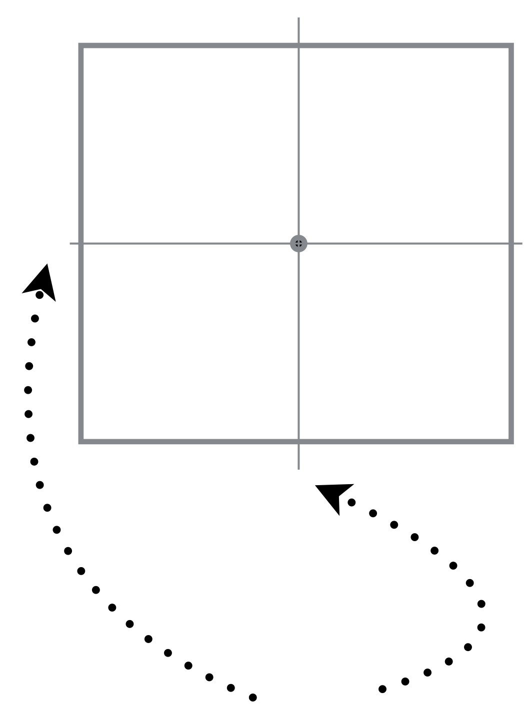
For instance in Vec/Set:



Yes!

coproducts
are
mono-colimits

Yes!

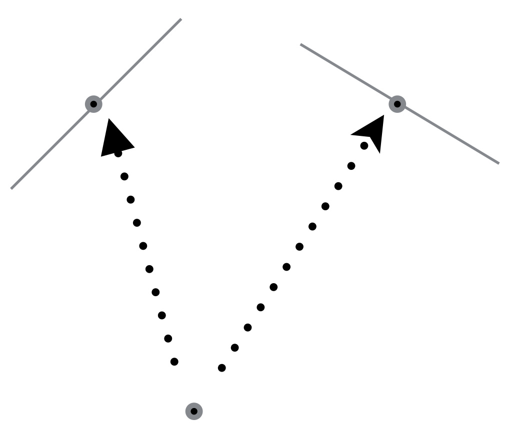


Defining **Glue(Vec)** in categorical terms

Consider a category that already has colimits (for instance Vec)

A mono-co-limit diagram is a diagram such that the universal cocone consists only of monos.

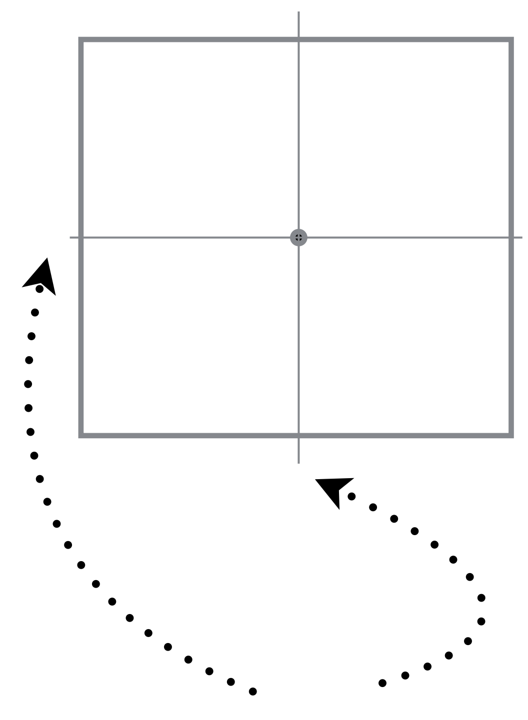
For instance in Vec/Set:



Yes!

coproducts
are
mono-colimits

Yes!



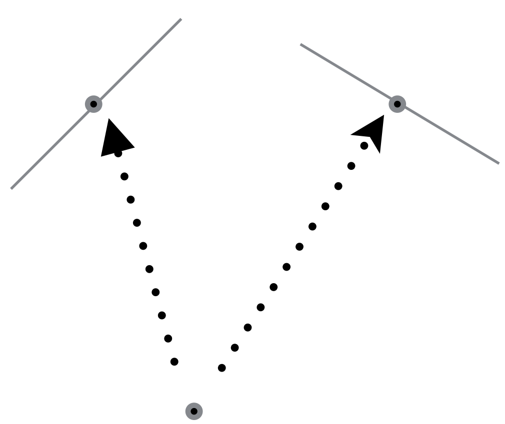
No!

Defining **Glue(Vec)** in categorical terms

Consider a category that already has colimits (for instance Vec)

A mono-co-limit diagram is a diagram such that the universal cocone consists only of monos.

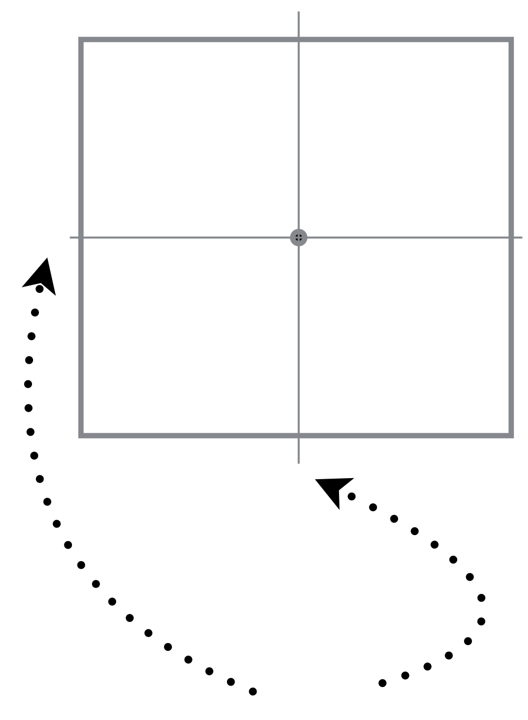
For instance in Vec/Set:



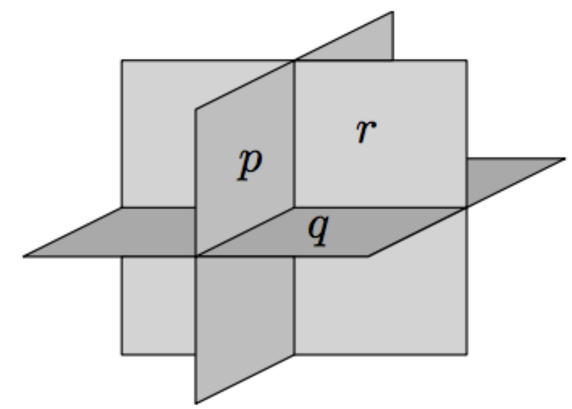
Yes!

coproducts
are
mono-colimits

Yes!



No!

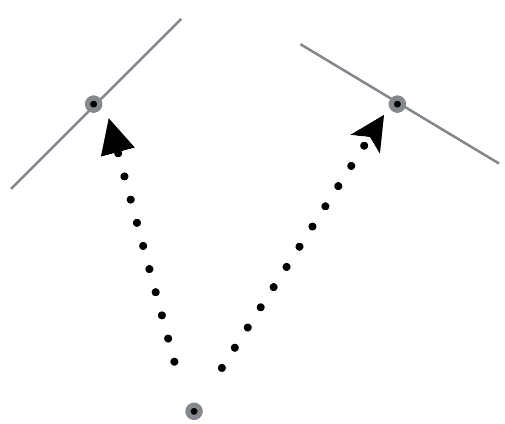


Defining **Glue(Vec)** in categorical terms

Consider a category that already has colimits (for instance Vec)

A mono-co-limit diagram is a diagram such that the universal cocone consists only of monos.

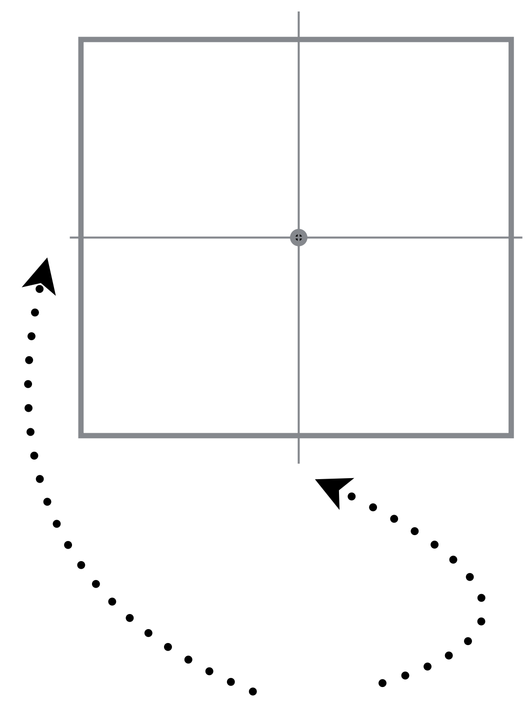
For instance in Vec/Set:



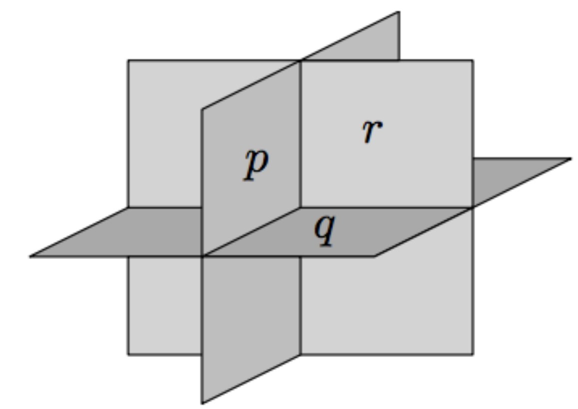
Yes!

coproducts
are
mono-colimits

Yes!



No!



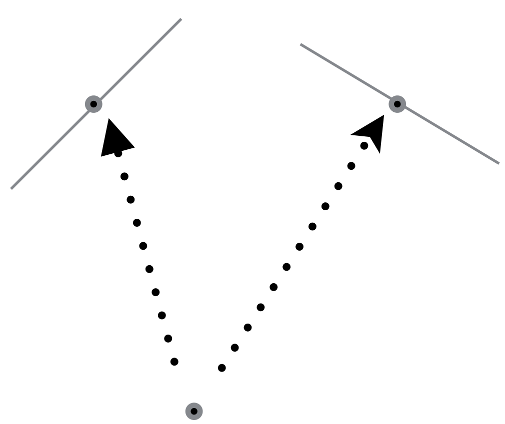
Yes!

Defining **Glue(Vec)** in categorical terms

Consider a category that already has colimits (for instance Vec)

A mono-co-limit diagram is a diagram such that the universal cocone consists only of monos.

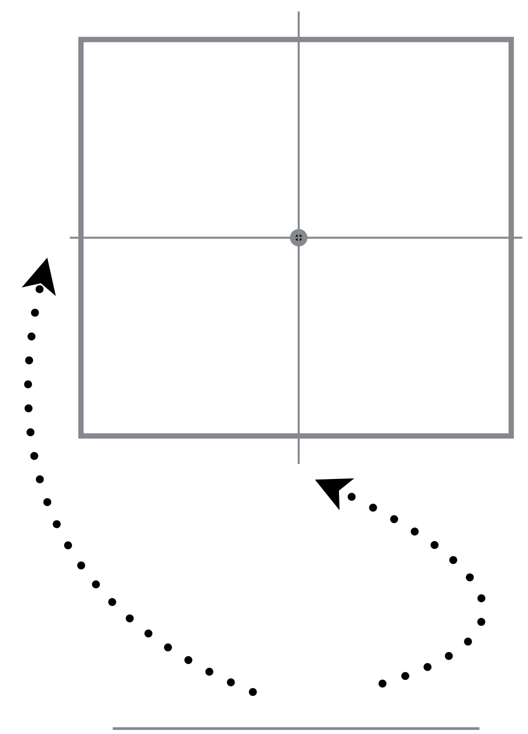
For instance in Vec/Set:



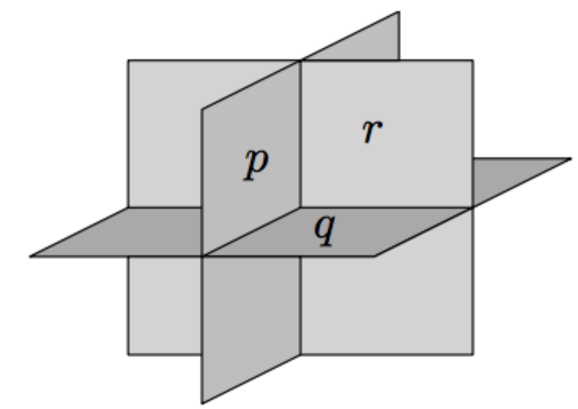
Yes!

coproducts
are
mono-colimits

Yes!



No!



Yes!

Definition:
The glueings of a category is its free completion under mono-co-limits

Example: continued

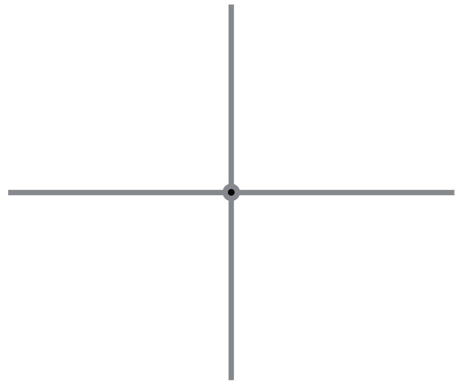
The minimal automaton for our example is:

Example: continued

The minimal automaton for our example is:

$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$

with $(\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$



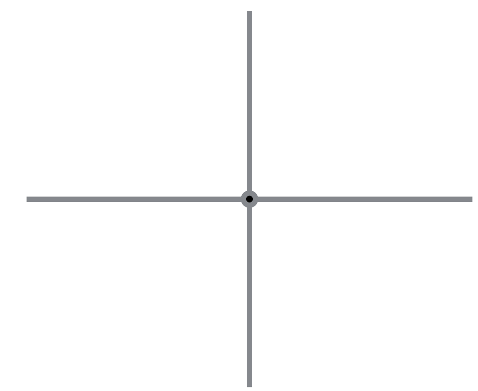
Example: continued

The minimal automaton for our example is:

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

with $(\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$

$$i(x) = (\texttt{even}, x)$$

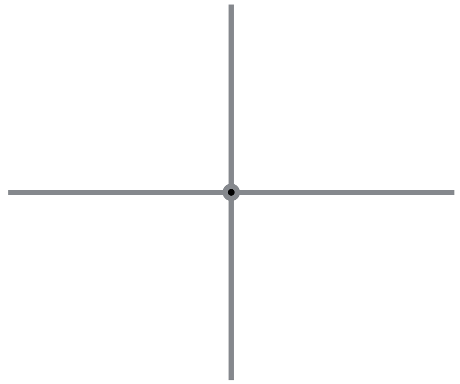


Example: continued

The minimal automaton for our example is:

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

with $(\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$

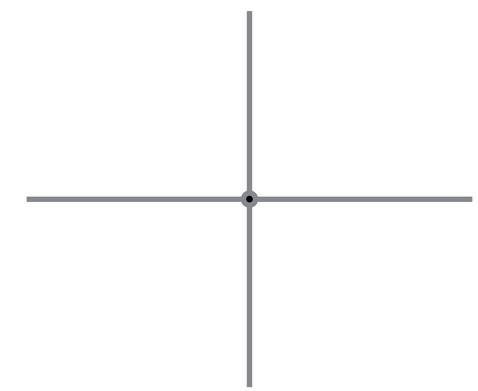


$$i(x) = (\texttt{even}, x)$$

$$f(\texttt{even}, x) = x$$
$$f(\texttt{odd}, x) = 0$$

Example: continued

The minimal automaton for our example is:



$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

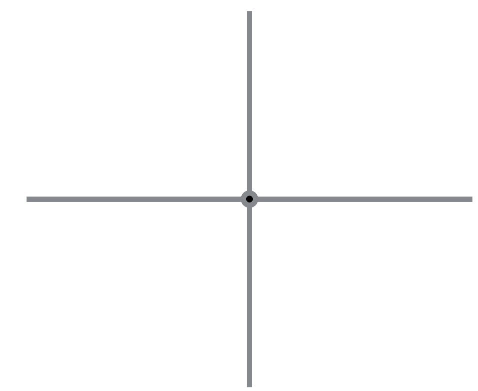
with $(\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$

$$i(x) = (\texttt{even}, x)$$

$$\left.\begin{array}{l} f(\texttt{even}, x) = x \\ f(\texttt{odd}, x) = 0 \end{array}\right\}$$ agrees on $(\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$

Example: continued

The minimal automaton for our example is:



$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

with $(\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$

$$i(x) = (\texttt{even}, x)$$

$$\left.\begin{array}{l} f(\texttt{even}, x) = x \\ f(\texttt{odd}, x) = 0 \end{array}\right\} \quad \text{agrees on } (\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$$

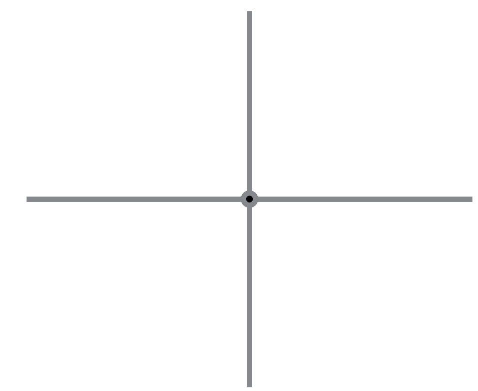
$$\delta_a(\texttt{even}, x) = (\texttt{even}, 2x)$$
$$\delta_a(\texttt{odd}, x) = (\texttt{odd}, 2x)$$

Example: continued

The minimal automaton for our example is:

$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

with $(\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$



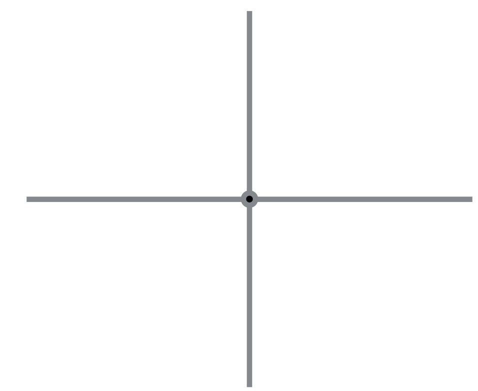
$$i(x) = (\texttt{even}, x)$$

$$\left.\begin{aligned} f(\texttt{even}, x) &= x \\ f(\texttt{odd}, x) &= 0 \end{aligned}\right\} \quad \text{agrees on } (\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$$

$$\left.\begin{aligned} \delta_a(\texttt{even}, x) &= (\texttt{even}, 2x) \\ \delta_a(\texttt{odd}, x) &= (\texttt{odd}, 2x) \end{aligned}\right\} \quad \text{agrees on } (\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$$

Example: continued

The minimal automaton for our example is:



$$Q = (\{\texttt{odd}\} \times \mathbb{R}) \cup (\{\texttt{even}\} \times \mathbb{R})$$

with $(\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$

$$i(x) = (\texttt{even}, x)$$

$$\left.\begin{aligned} f(\texttt{even}, x) &= x \\ f(\texttt{odd}, x) &= 0 \end{aligned}\right\}$$ agrees on $(\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$

$$\left.\begin{aligned} \delta_a(\texttt{even}, x) &= (\texttt{even}, 2x) \\ \delta_a(\texttt{odd}, x) &= (\texttt{odd}, 2x) \end{aligned}\right\}$$ agrees on $(\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$

$$\left.\begin{aligned} \delta_b(\texttt{even}, x) &= (\texttt{odd}, x) \\ \delta_b(\texttt{odd}, x) &= (\texttt{even}, x) \end{aligned}\right\}$$ agrees on $(\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$

$$\left.\begin{aligned} \delta_c(\texttt{even}, x) &= (\texttt{even}, 0) \\ \delta_c(\texttt{odd}, x) &= (\texttt{odd}, 0) \end{aligned}\right\}$$ agrees on $(\texttt{even}, 0) \sim_{\texttt{glue}} (\texttt{odd}, 0)$

Properties of automata one glueings of vector spaces

Properties of automata one glueings of vector spaces

There exists an initial and a final automaton for a Glue(Vec)-language.

Properties of automata one glueings of vector spaces

There exists an initial and a final automaton for a Glue(Vec)-language.

There is a natural factorization system « (surjection like,injection like) ».

Properties of automata one glueings of vector spaces

There exists an initial and a final automaton for a Glue(Vec)-language.

There is a natural factorization system « (surjection like,injection like) ».

However, this yields wrong minimal automata:

Properties of automata one glueings of vector spaces

There exists an initial and a final automaton for a Glue(Vec)-language.

There is a natural factorization system « (surjection like,injection like) ».

However, this yields wrong minimal automata:

$$L(a^n)(x) = x\cos(n\alpha)$$

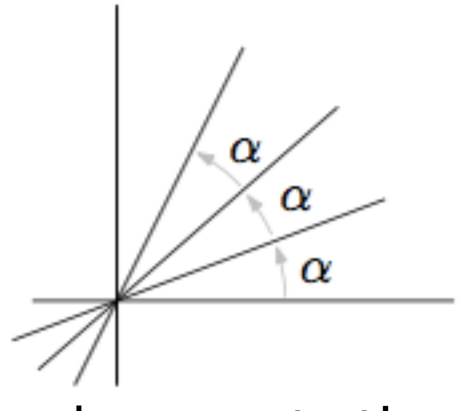
Properties of automata one glueings of vector spaces

There exists an initial and a final automaton for a Glue(Vec)-language.

There is a natural factorization system « (surjection like,injection like) ».

However, this yields wrong minimal automata:

$$L(a^n)(x) = x\cos(n\alpha)$$



« implementation »

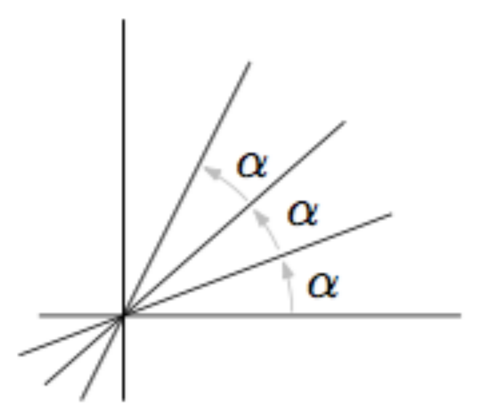
Properties of automata one glueings of vector spaces

There exists an initial and a final automaton for a Glue(Vec)-language.

There is a natural factorization system « (surjection like,injection like) ».

However, this yields wrong minimal automata:

$$L(a^n)(x) = x\cos(n\alpha)$$



« implementation »

For α not a rational multiple of π, the minimal automaton contains countable many copies of $\mathbb{R}$, …one for each n…

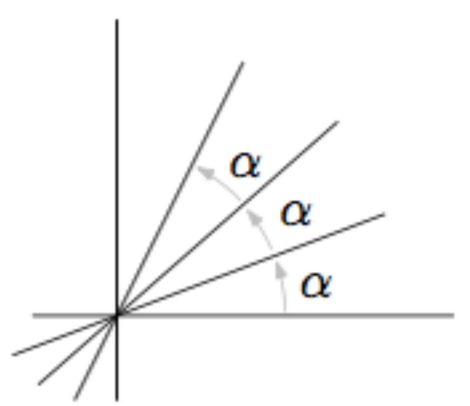
Properties of automata one glueings of vector spaces

There exists an initial and a final automaton for a Glue(Vec)-language.

There is a natural factorization system « (surjection like,injection like) ».

However, this yields wrong minimal automata:

$$L(a^n)(x) = x\cos(n\alpha)$$



« implementation »

For α not a rational multiple of π, the minimal automaton contains countable many copies of $\mathbb{R}$, …one for each n…

This is not what we wanted: we implicitly wanted to minimize among finite glueings of finite dimension vector spaces.

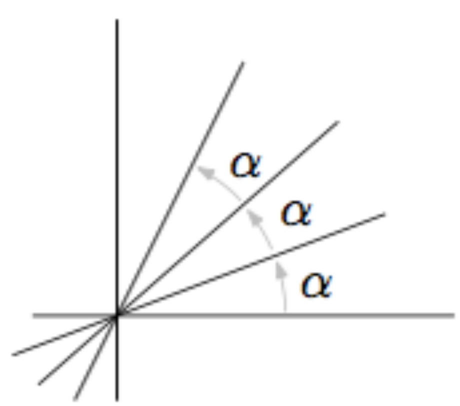
Properties of automata one glueings of vector spaces

There exists an initial and a final automaton for a Glue(Vec)-language.

There is a natural factorization system « (surjection like,injection like) ».

However, this yields wrong minimal automata:

$$L(a^n)(x) = x\cos(n\alpha)$$



« implementation »

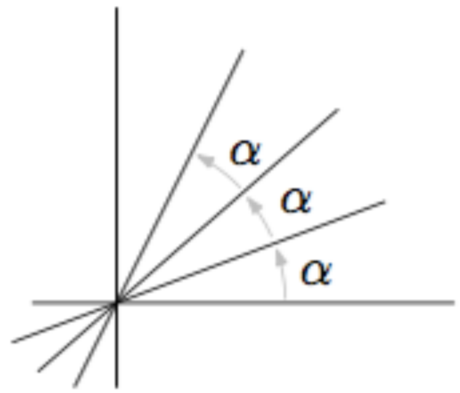
For α not a rational multiple of π, the minimal automaton contains countable many copies of $\mathbb{R}$, …one for each n…

This is not what we wanted: we implicitly wanted to minimize among finite glueings of finite dimension vector spaces.

Theorem: For Glue(Vec)-languages recognized by GlueFin(VecFin)-automata, there exists a minimal automaton for the language among GlueFin(VecFin)-automata.

The problem

$$L(a^n)(x) = x\cos(n\alpha)$$



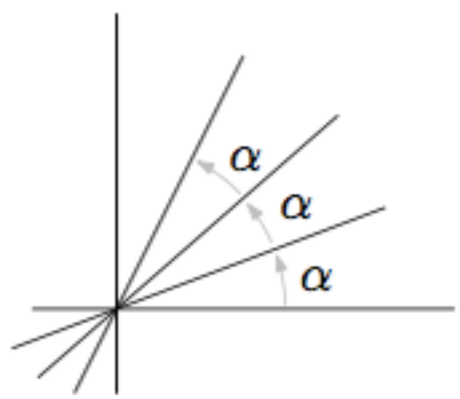
« implementation »

For α not a rational multiple of π, the minimal automaton contains countable many copies of $\mathbb{R}$, …one for each n…

This is not what we wanted: we implicitly wanted to minimize among finite glueings of finite dimension vector spaces.

The problem

$$L(a^n)(x) = x\cos(n\alpha)$$



« implementation »

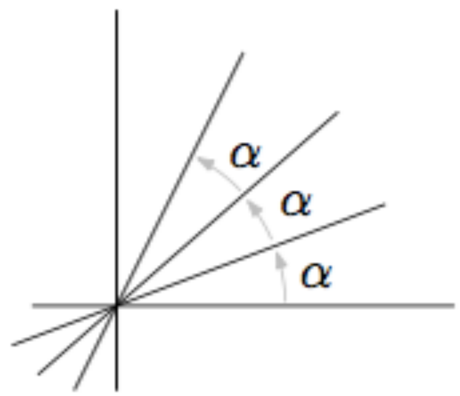
For α not a rational multiple of π, the minimal automaton contains countable many copies of $\mathbb{R}$, …one for each n…

This is not what we wanted: we implicitly wanted to minimize among finite glueings of finite dimension vector spaces.

Glue(Vec)
-automata
for L

The problem

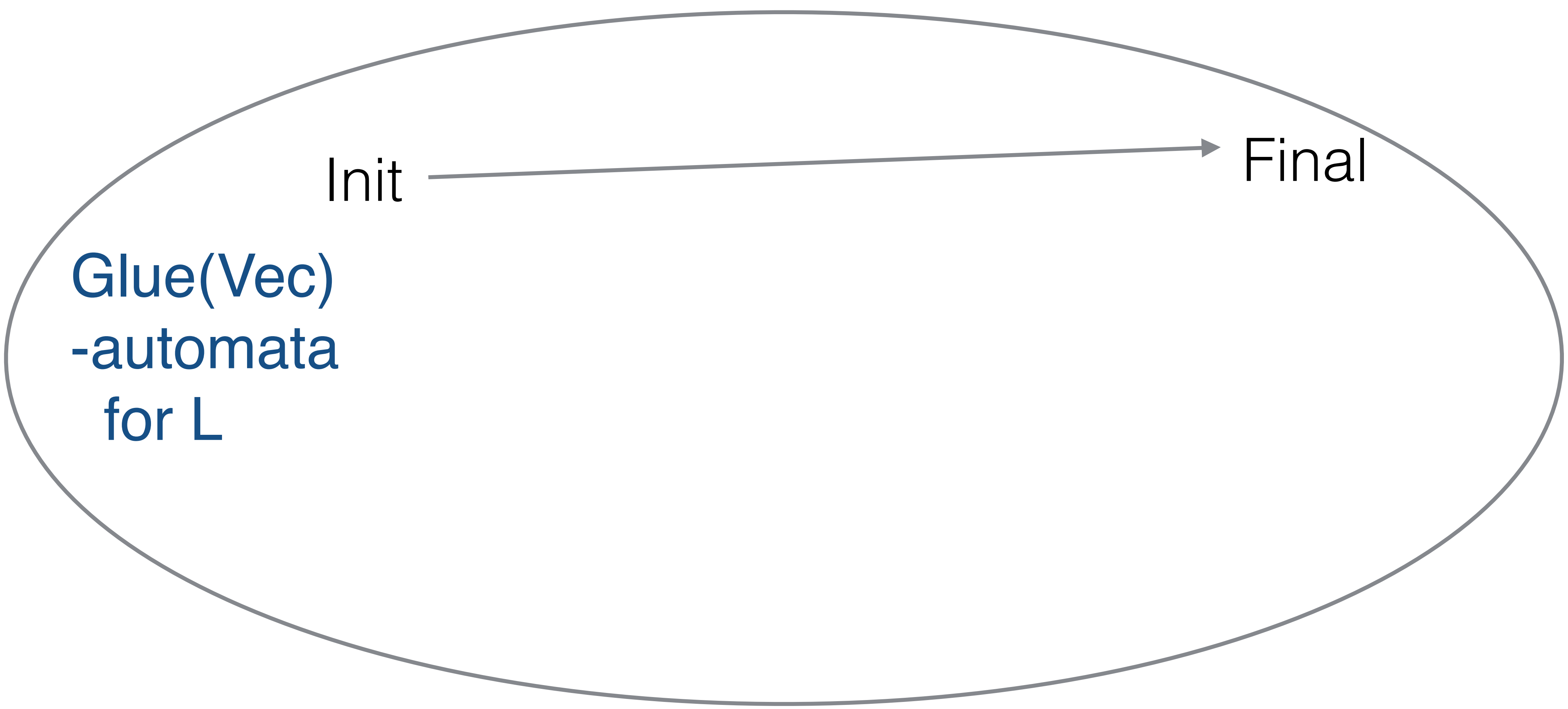
$$L(a^n)(x) = x\cos(n\alpha)$$



« implementation »

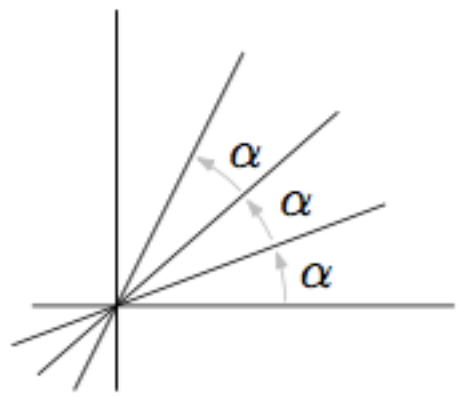
For α not a rational multiple of π, the minimal automaton contains countable many copies of $\mathbb{R}$, …one for each n…

This is not what we wanted: we implicitly wanted to minimize among finite glueings of finite dimension vector spaces.



The problem

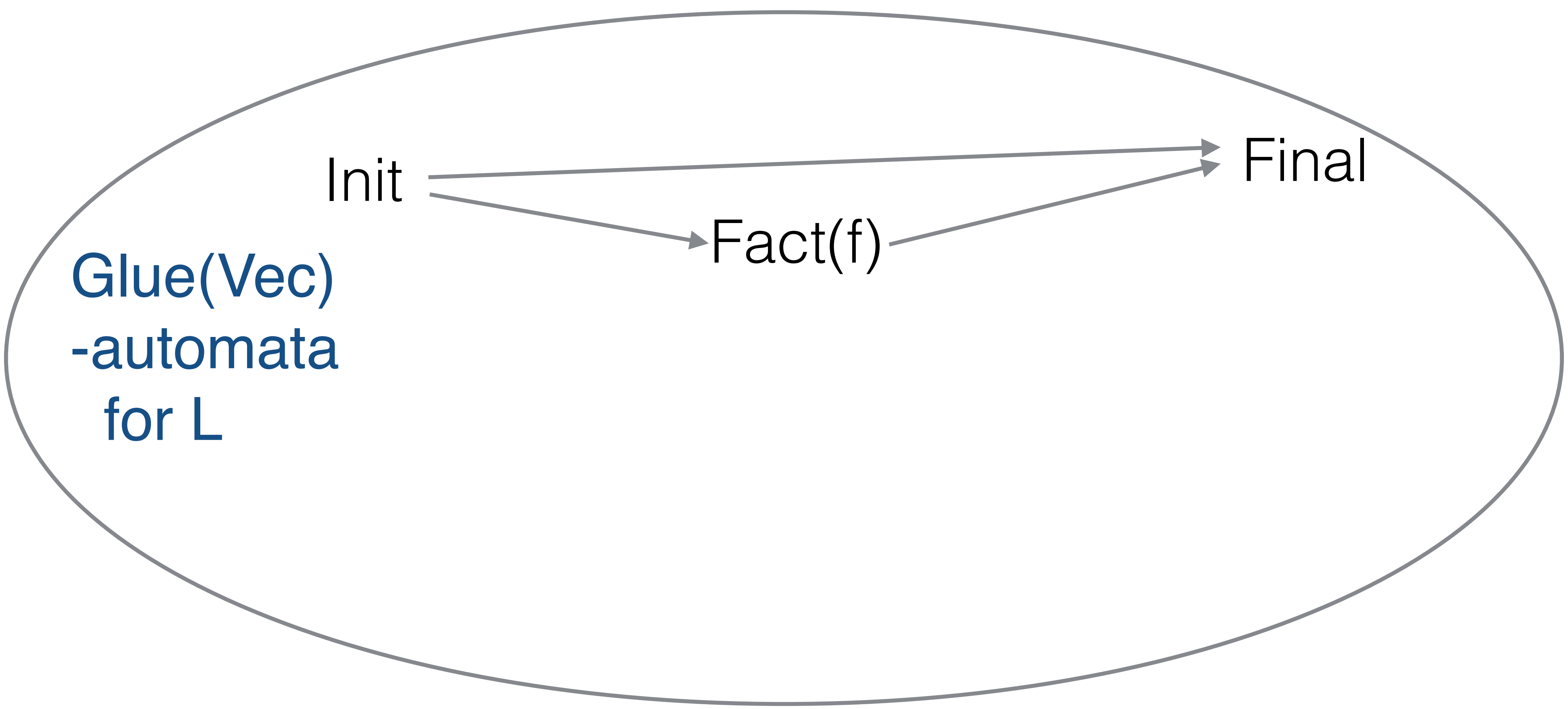
$$L(a^n)(x) = x\cos(n\alpha)$$



« implementation »

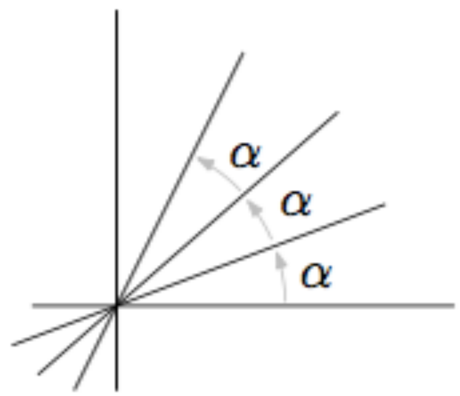
For α not a rational multiple of π, the minimal automaton contains countable many copies of $\mathbb{R}$, …one for each n…

This is not what we wanted: we implicitly wanted to minimize among **finite glueings of finite dimension vector spaces**.



The problem

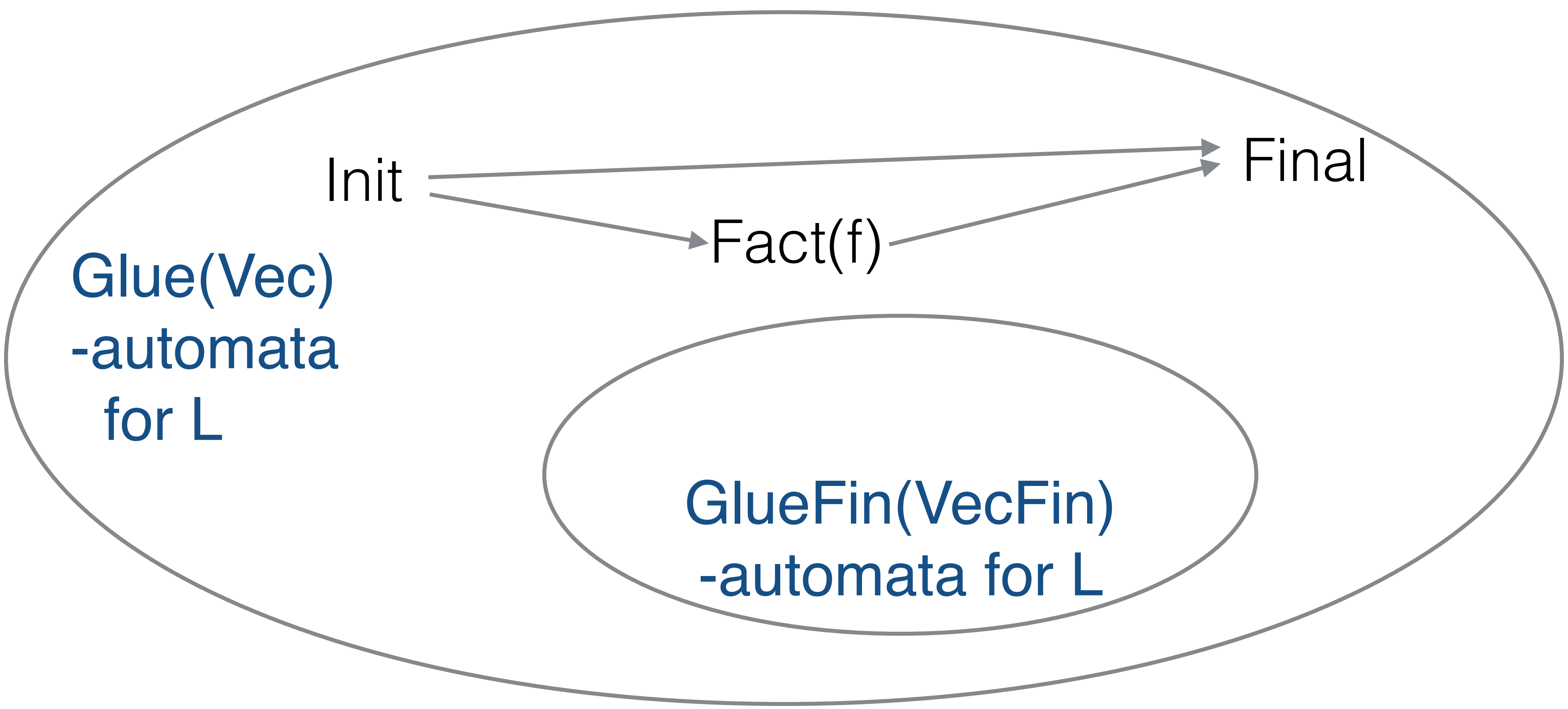
$$L(a^n)(x) = x\cos(n\alpha)$$



« implementation »

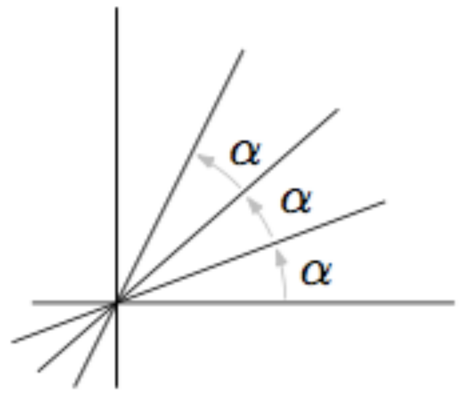
For α not a rational multiple of π, the minimal automaton contains countable many copies of $\mathbb{R}$, …one for each n…

This is not what we wanted: we implicitly wanted to minimize among finite glueings of finite dimension vector spaces.



The problem

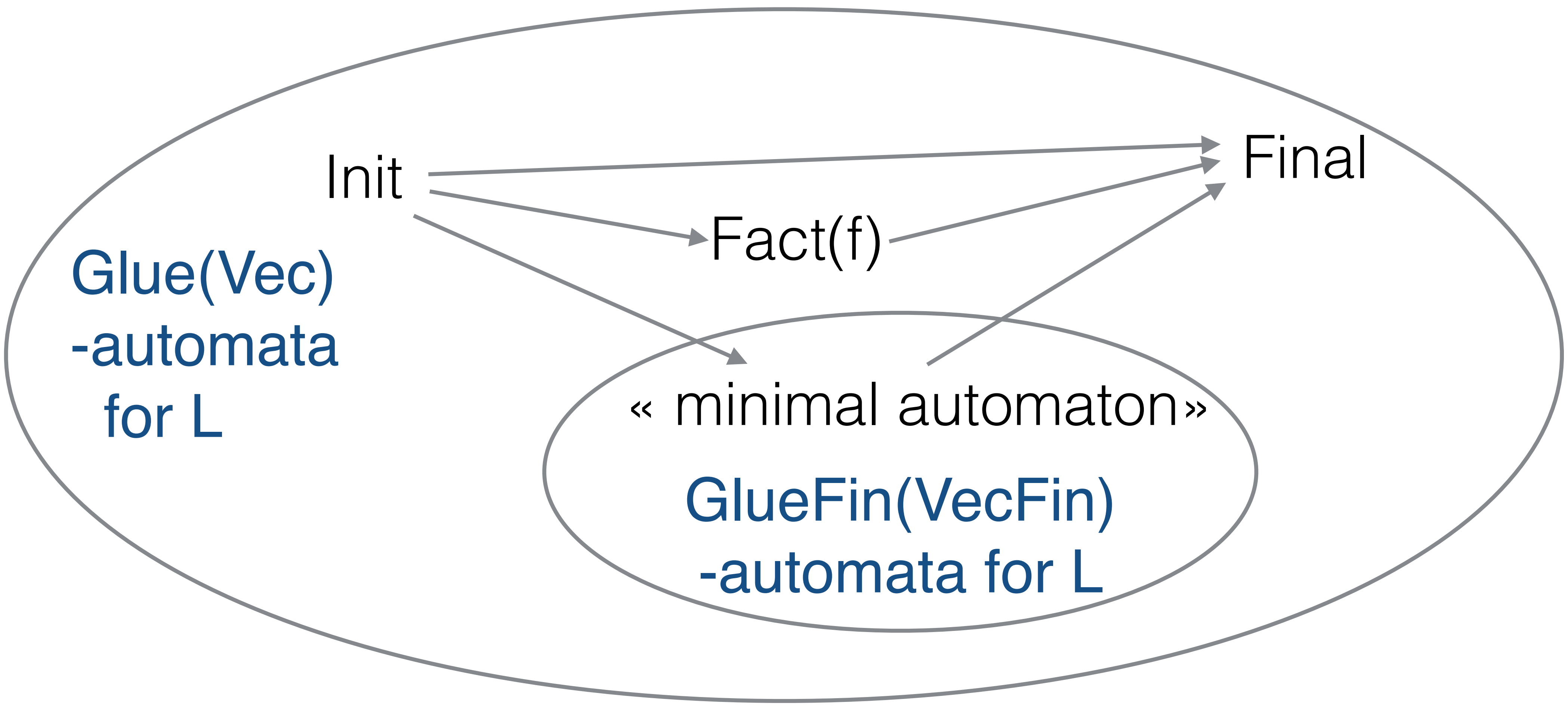
$$L(a^n)(x) = x\cos(n\alpha)$$



« implementation »

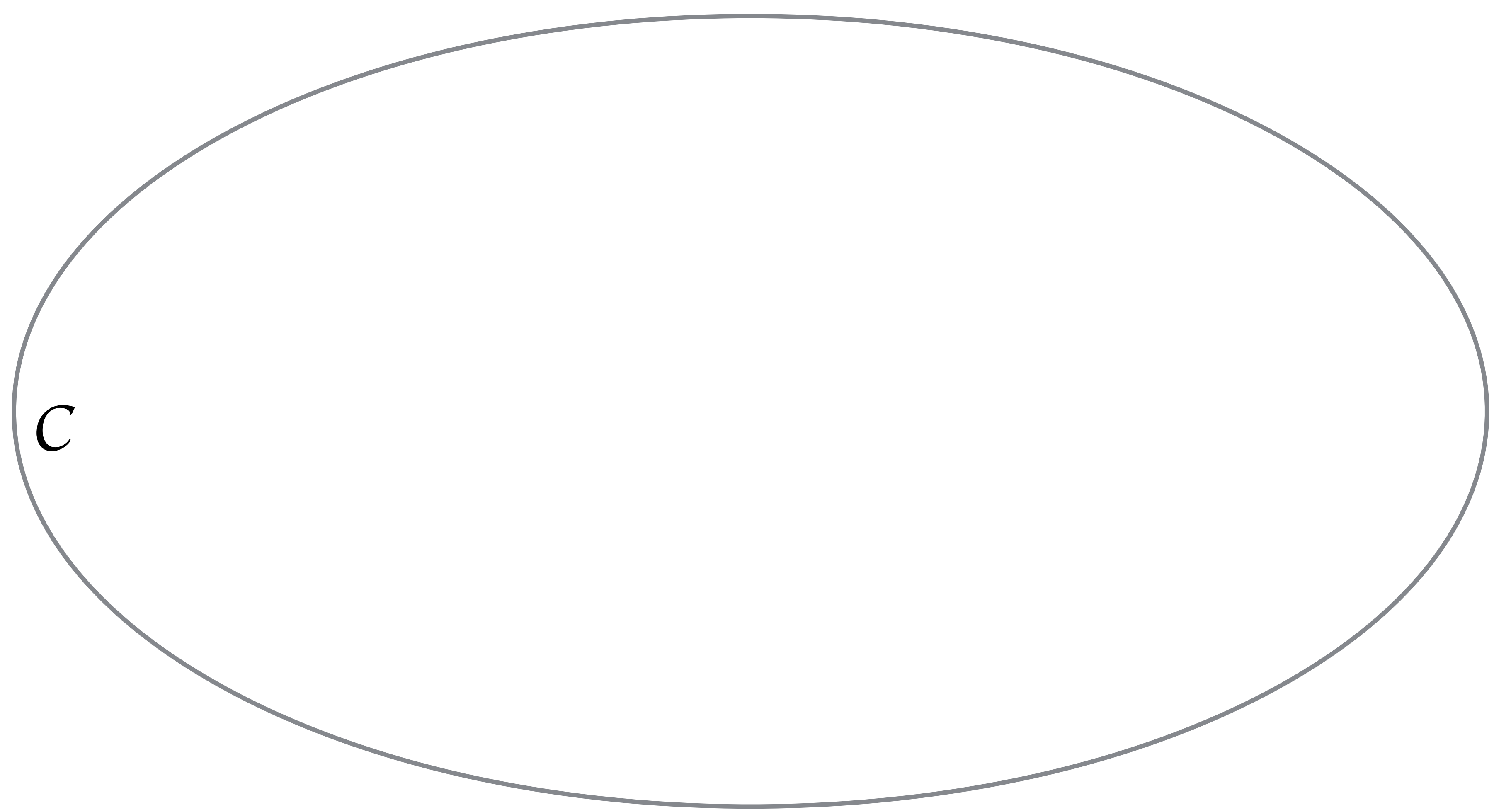
For α not a rational multiple of π, the minimal automaton contains countable many copies of $\mathbb{R}$, …one for each n…

This is not what we wanted: we implicitly wanted to minimize among finite glueings of finite dimension vector spaces.

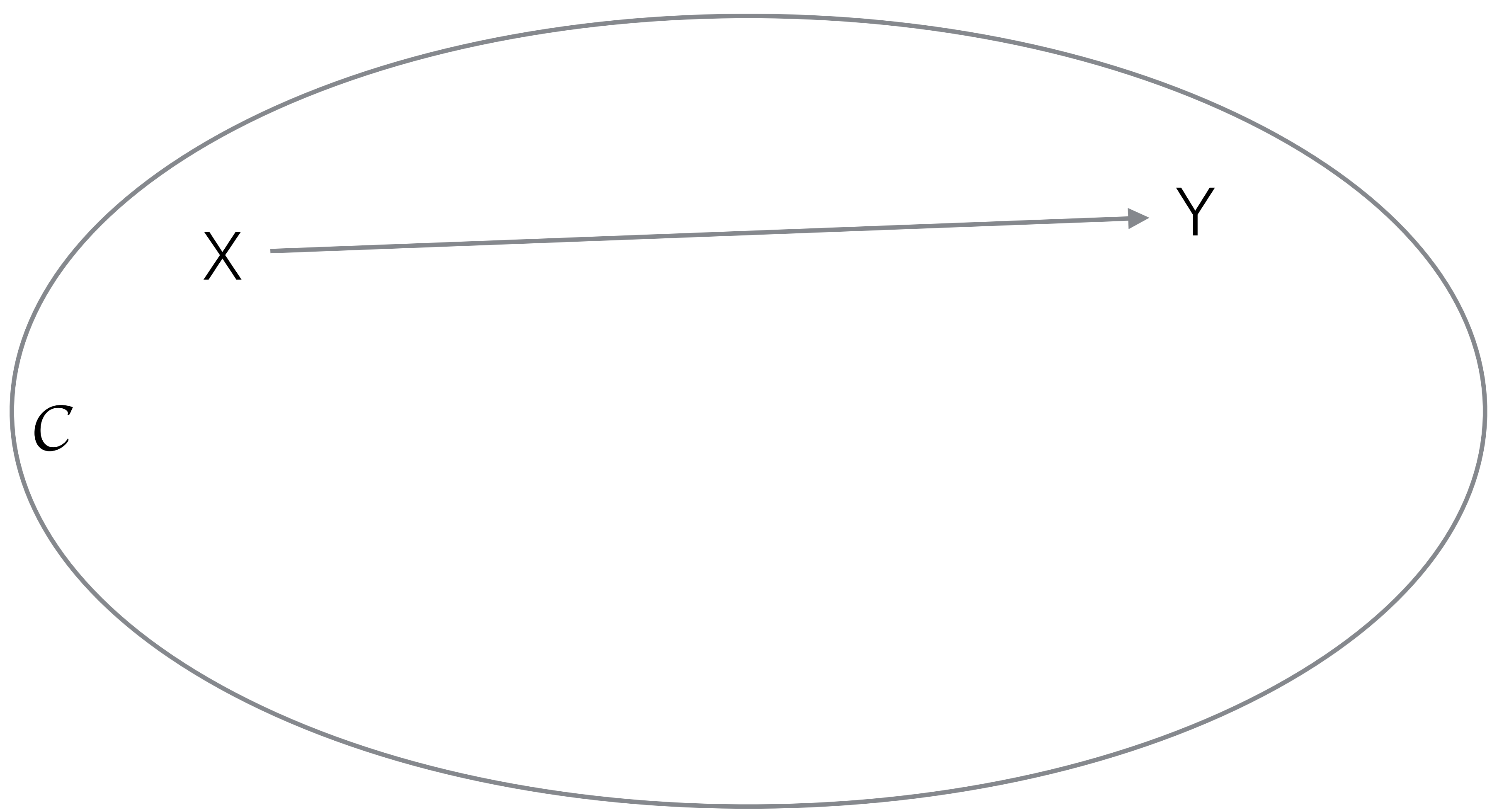


Idea 1: factorization through

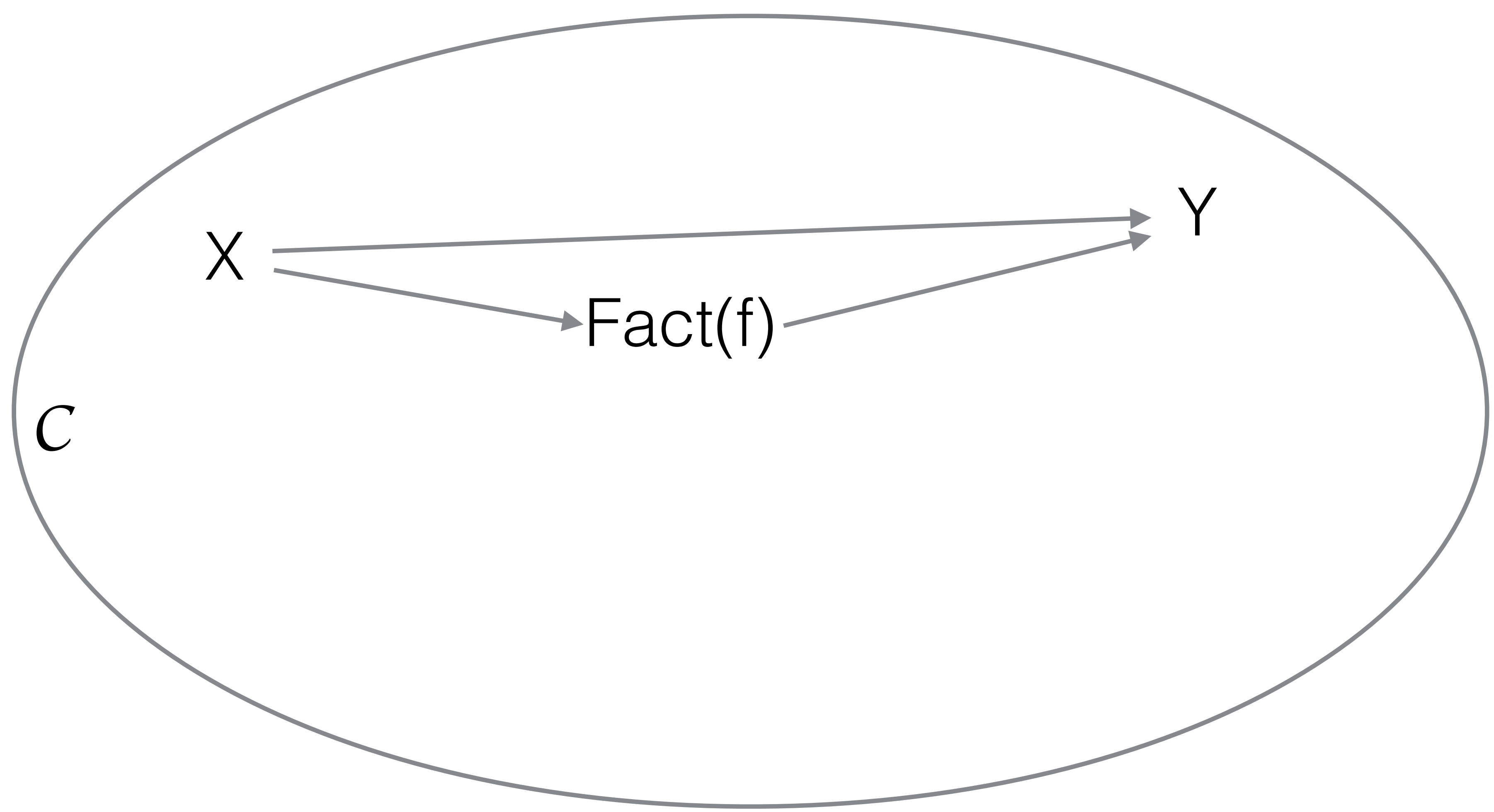
Idea 1: factorization through



Idea 1: factorization through



Idea 1: factorization through

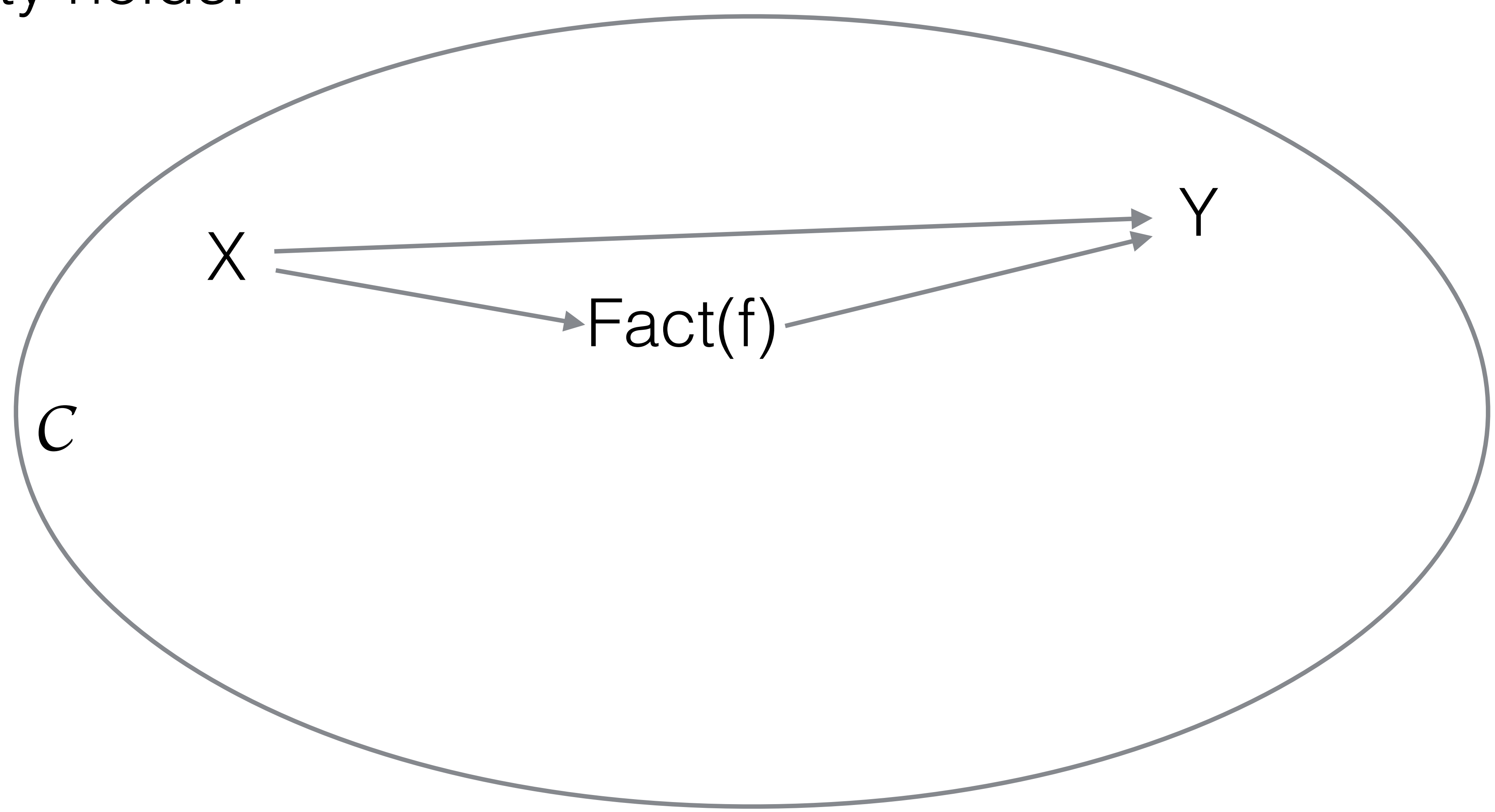


Idea 1: factorization through

A f**actorization system through a subcategory** $\mathcal{S}$

consists of classes $(\mathcal{E},\mathcal{M})$ such that:

- $\mathcal{E}$-arrows end in $\mathcal{S}$ and are closer under composition
- $\mathcal{M}$-arrows start in $\mathcal{S}$ and are closer under composition
- all arrows that factorize through **S has** $(\mathcal{E},\mathcal{M})$ factorization.
- the diagnoal property holds.

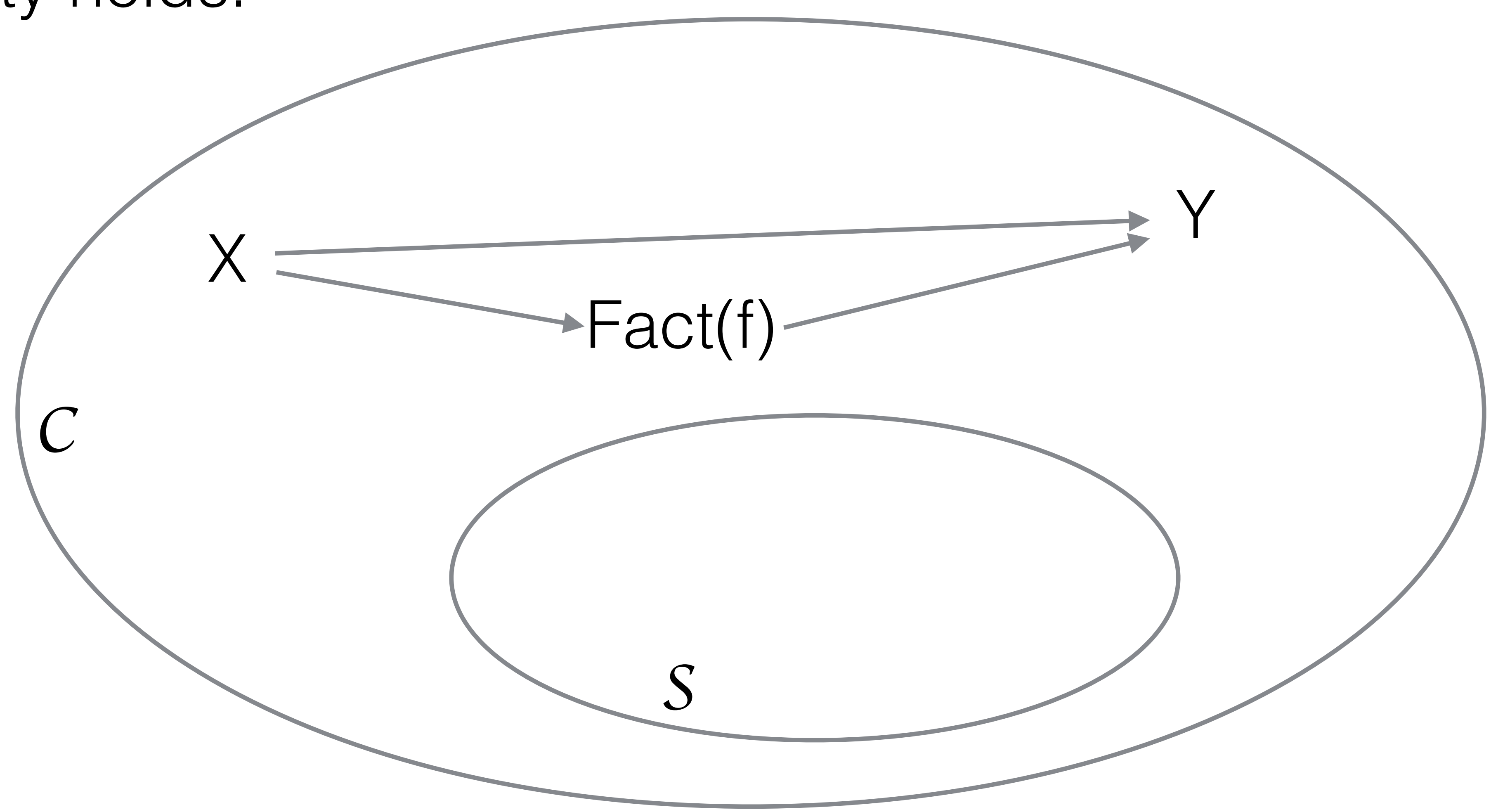


Idea 1: factorization through

A factorization system through a subcategory $\mathcal{S}$

consists of classes $(\mathcal{E},\mathcal{M})$ such that:

- $\mathcal{E}$-arrows end in $\mathcal{S}$ and are closer under composition
- $\mathcal{M}$-arrows start in $\mathcal{S}$ and are closer under composition
- all arrows that factorize through **S has** $(\mathcal{E},\mathcal{M})$ factorization.
- the diagnoal property holds.

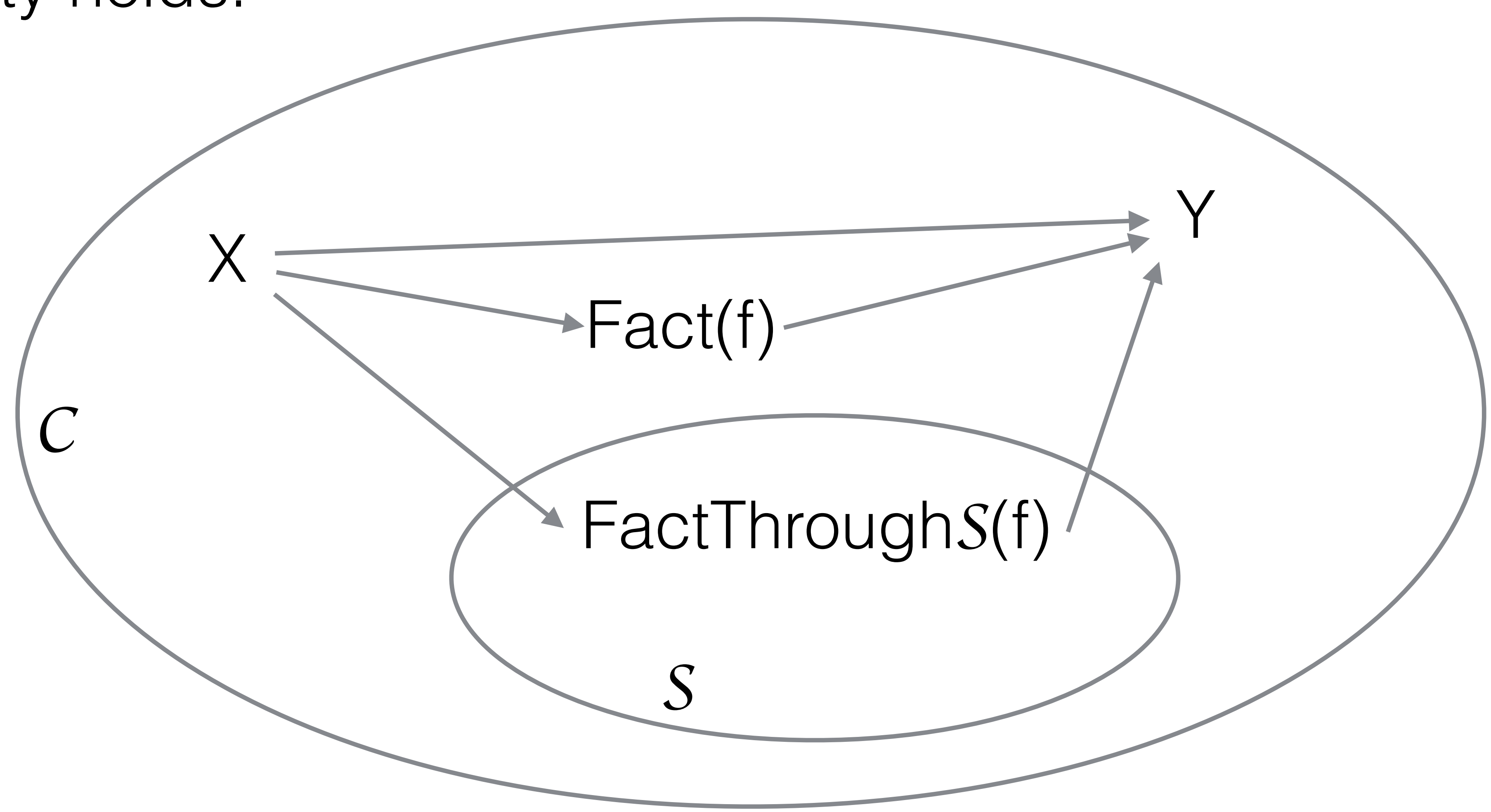


Idea 1: factorization through

A f**actorization system through a subcategory** $\mathcal{S}$

consists of classes ($\mathcal{E}$,$\mathcal{M}$) such that:

- $\mathcal{E}$-arrows end in $\mathcal{S}$ and are closer under composition
- $\mathcal{M}$-arrows start in $\mathcal{S}$ and are closer under composition
- all arrows that factorize through **S has** ($\mathcal{E}$,$\mathcal{M}$) factorization.
- the diagnoal property holds.



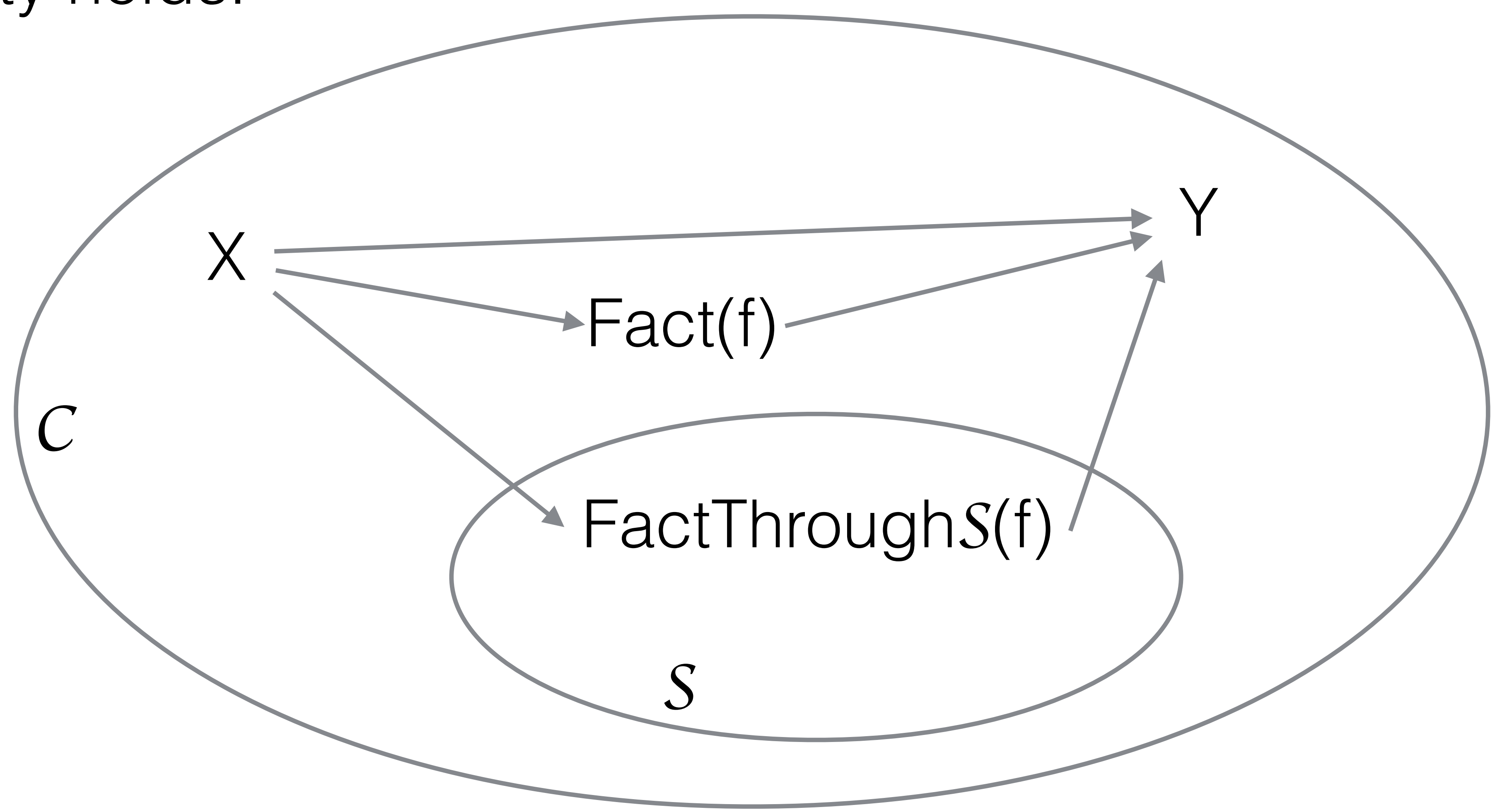
Idea 1: factorization through

A f**actorization system through a subcategory** $\mathcal{S}$

consists of classes $(\mathcal{E},\mathcal{M})$ such that:

- $\mathcal{E}$-arrows end in $\mathcal{S}$ and are closer under composition
- $\mathcal{M}$-arrows start in $\mathcal{S}$ and are closer under composition
- all arrows that factorize through **S has** $(\mathcal{E},\mathcal{M})$ factorization.
- the diagnoal property holds.

Theorem:
Glue(Vec) has a facrotization system through GlueFin(VecFin)



Idea 1: factorization through

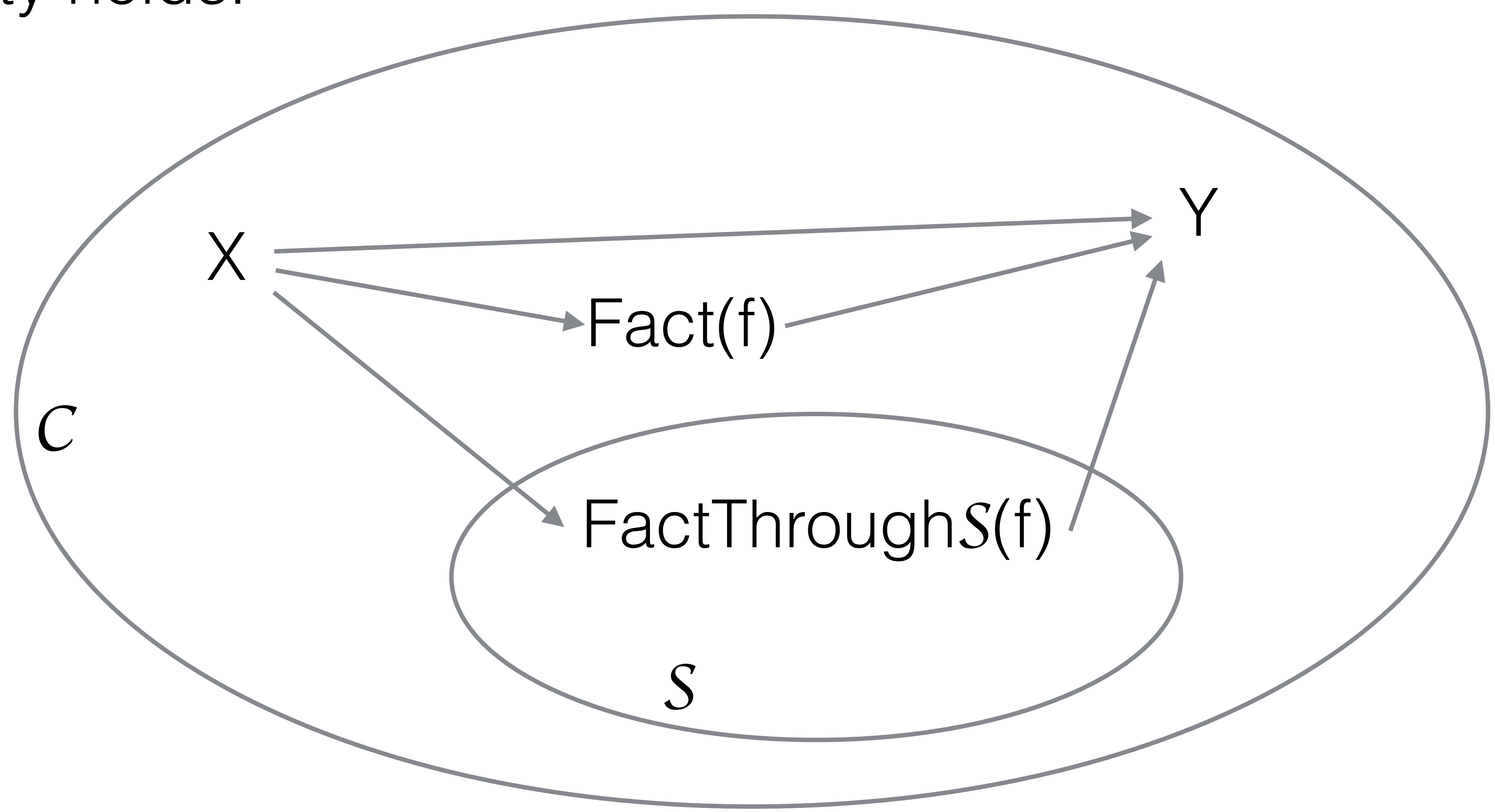
A factorization system through a subcategory $\mathcal{S}$

consists of classes ($\mathcal{E}$,$\mathcal{M}$) such that:

- $\mathcal{E}$-arrows end in $\mathcal{S}$ and are closer under composition
- $\mathcal{M}$-arrows start in $\mathcal{S}$ and are closer under composition
- all arrows that factorize through **S has** ($\mathcal{E}$,$\mathcal{M}$) factorization.
- the diagnoal property holds.

Theorem:
Glue(Vec) has a facrotization system through GlueFin(VecFin)

(the same goes for automata)



Factorization of glueings

Factorization of glueings

Theorem:

Glue(Vec) has a factorization system through GlueFin(VecFin)

Factorization of glueings

Theorem:
Glue(Vec) has a factorization system through GlueFin(VecFin)

Lemma: finite unions of subspaces of a finite dimension vector space are closed under arbitrary intersection.

Factorization of glueings

Theorem:
Glue(Vec) has a factorization system through GlueFin(VecFin)

Lemma: finite unions of subspaces of a finite dimension vector space are closed under arbitrary intersection.

Corollary: for all subset of a (finite dimension) vector space, there is a least finite union of vector spaces that covers it.

Factorization of glueings

Theorem:
Glue(Vec) has a factorization system through GlueFin(VecFin)

Lemma: finite unions of subspaces of a finite dimension vector space are closed under arbitrary intersection.

Corollary: for all subset of a (finite dimension) vector space, there is a least finite union of vector spaces that covers it.

This is the closure in the topology where closed sets are finite unions of subspaces.

Factorization of glueings

Theorem:
Glue(Vec) has a factorization system through GlueFin(VecFin)

Lemma: finite unions of subspaces of a finite dimension vector space are closed under arbitrary intersection.

Corollary: for all subset of a (finite dimension) vector space, there is a least finite union of vector spaces that covers it.

This is the closure in the topology where closed sets are finite unions of subspaces.

This is a coarsening of Zariski topology.

Factorization of glueings

Theorem:
Glue(Vec) has a factorization system through GlueFin(VecFin)

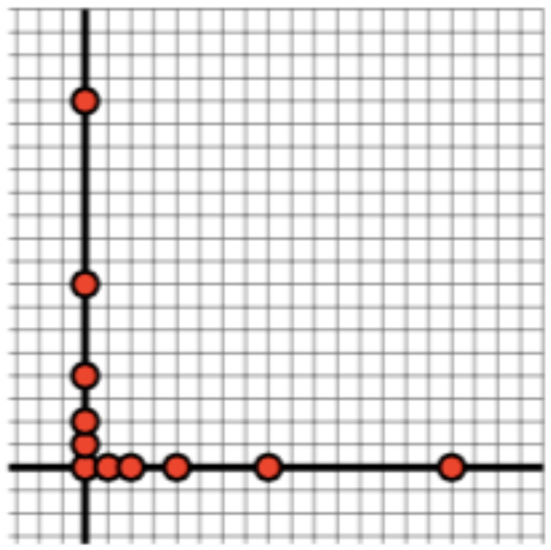
Lemma: finite unions of subspaces of a finite dimension vector space are closed under arbitrary intersection.

Corollary: for all subset of a (finite dimension) vector space, there is a least finite union of vector spaces that covers it.

This is the closure in the topology where closed sets are finite unions of subspaces.

This is a coarsening of Zariski topology.

Proof by example in affine spaces



Factorization of glueings

Theorem:
Glue(Vec) has a factorization system through GlueFin(VecFin)

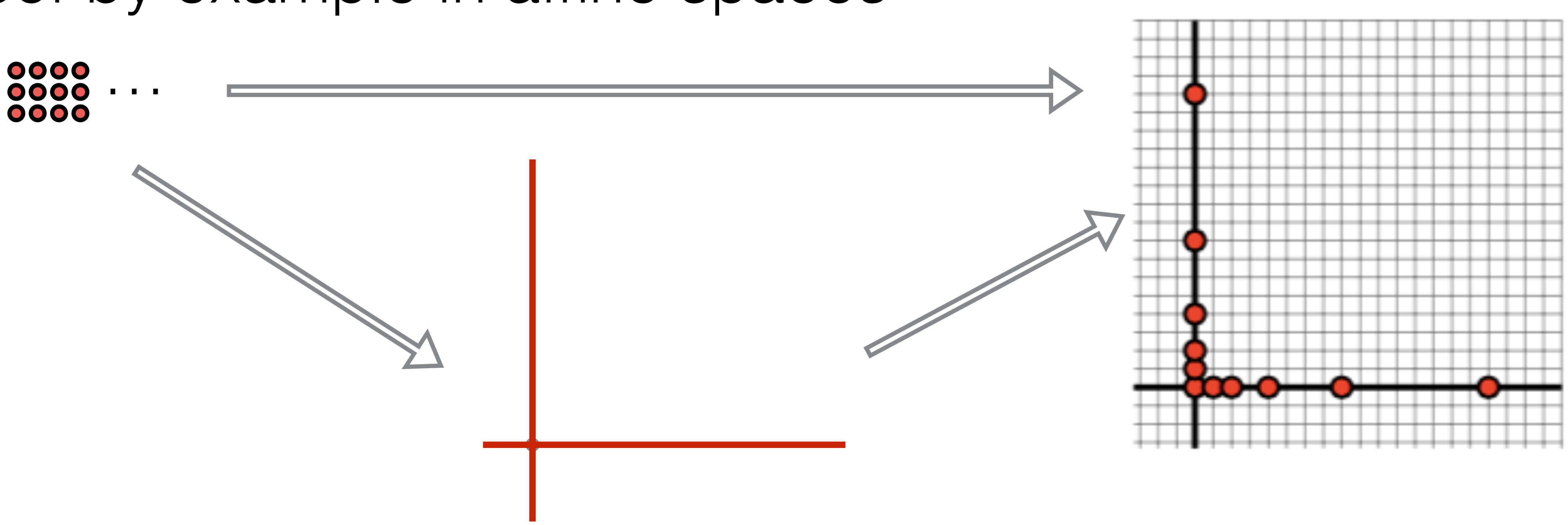
Lemma: finite unions of subspaces of a finite dimension vector space are closed under arbitrary intersection.

Corollary: for all subset of a (finite dimension) vector space, there is a least finite union of vector spaces that covers it.

This is the closure in the topology where closed sets are finite unions of subspaces.

This is a coarsening of Zariski topology.

Proof by example in affine spaces



We obtain

Theorem: For Glue(Vec)-languages recognized by GlueFin(VecFin)-automata, there exists a minimal automaton for the language among GlueFin(VecFin)-automata.

We obtain

Theorem: For Glue(Vec)-languages recognized by GlueFin(VecFin)-automata, there exists a minimal automaton for the language among GlueFin(VecFin)-automata.

But, we do not know how to compute it…

The core algorithmic problem

Open problem: Given n×n matrices $A_1,\ldots,A_k$, compute the least finite union of subspaces of matrices that covers the generated semigroup.

For instance consider the matrice **Rot(α)** for some rational number α.

<u>If α is a rational multiple</u> of π, it should output the (finite union) of the dimension one spaces **Vec(Rot(nα))** for integer n.

Otherwise, the output is the two dimension vector spaces of matrices of the form

$$\begin{pmatrix} a & b \\ -b & a \end{pmatrix}$$

The core algorithmic problem

Open problem: Given n×n matrices $A_1,\ldots,A_k$, compute the least finite union of subspaces of matrices that covers the generated semigroup.

For instance consider the matrice **Rot(α)** for some rational number α.

If α is a rational multiple of π, it should output the (finite union) of the dimension one spaces **Vec(Rot(nα))** for integer n.

Otherwise, the output is the two dimension vector spaces of matrices of the form

$$\begin{pmatrix} a & b \\ -b & a \end{pmatrix}$$

Open problem: Given n×n matrices $A_1,\ldots,A_k$, compute the Zariski closure of the semigroup generated by these matrices.

Conclusion

Contributions

Contributions

- A categorical description of why minimization is possible,
- new categorical concepts on the way,
- new ways to construct categories that yield natural classes of minimizable automata using « glueings ».

Contributions

- A categorical description of why minimization is possible,
- new categorical concepts on the way,
- new ways to construct categories that yield natural classes of minimizable automata using « glueings ».

Related works

Contributions

- A categorical description of why minimization is possible,
- new categorical concepts on the way,
- new ways to construct categories that yield natural classes of minimizable automata using « glueings ».

Related works

- Schützenberger's weighted automata, and its long continuations [Sakarovitch, Lombardy, Droste, Gastin, Vogler, …]
- There is a long history of categorical view of minimization [Arbib, Manes, Adamek, Milius, Silva, Panangaden, Kupke…]

Contributions

- A categorical description of why minimization is possible,
- new categorical concepts on the way,
- new ways to construct categories that yield natural classes of minimizable automata using « glueings ».

Related works

- Schützenberger's weighted automata, and its long continuations [Sakarovitch, Lombardy, Droste, Gastin, Vogler, …]
- There is a long history of categorical view of minimization [Arbib, Manes, Adamek, Milius, Silva, Panangaden, Kupke…]

And then ?

Contributions

- A categorical description of why minimization is possible,
- new categorical concepts on the way,
- new ways to construct categories that yield natural classes of minimizable automata using « glueings ».

Related works

- Schützenberger's weighted automata, and its long continuations [Sakarovitch, Lombardy, Droste, Gastin, Vogler, …]
- There is a long history of categorical view of minimization [Arbib, Manes, Adamek, Milius, Silva, Panangaden, Kupke…]

And then ?

- Make this construction effective… (generalization of sequencialization)
- tree automata
- algebras (monoids,…)
- infinite objects (ω-semigroup, o-semigroup, monads…).

Questions ?