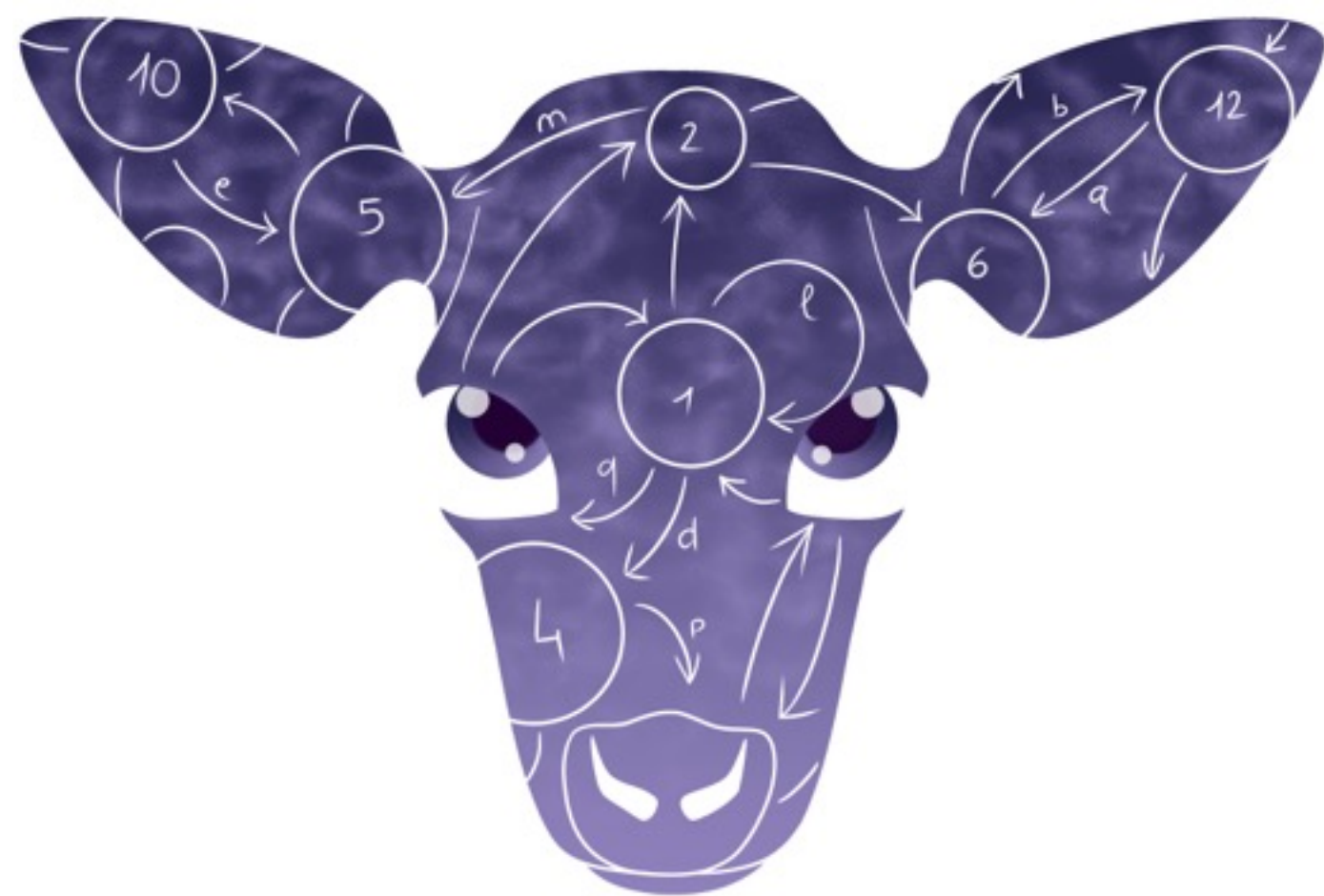


A (co)algebraic theory of succinct acceptors

Alexandra Silva

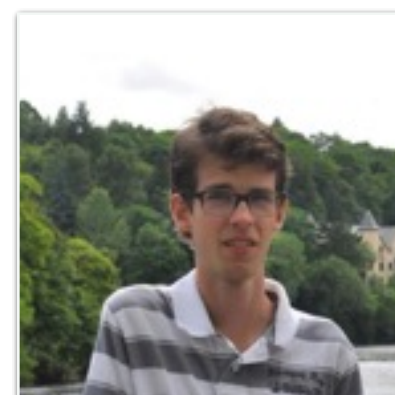


Categorical Automata Learning Framework

calf-project.org



Matteo Sammartino
UCL



Gerco van Heerdt
UCL

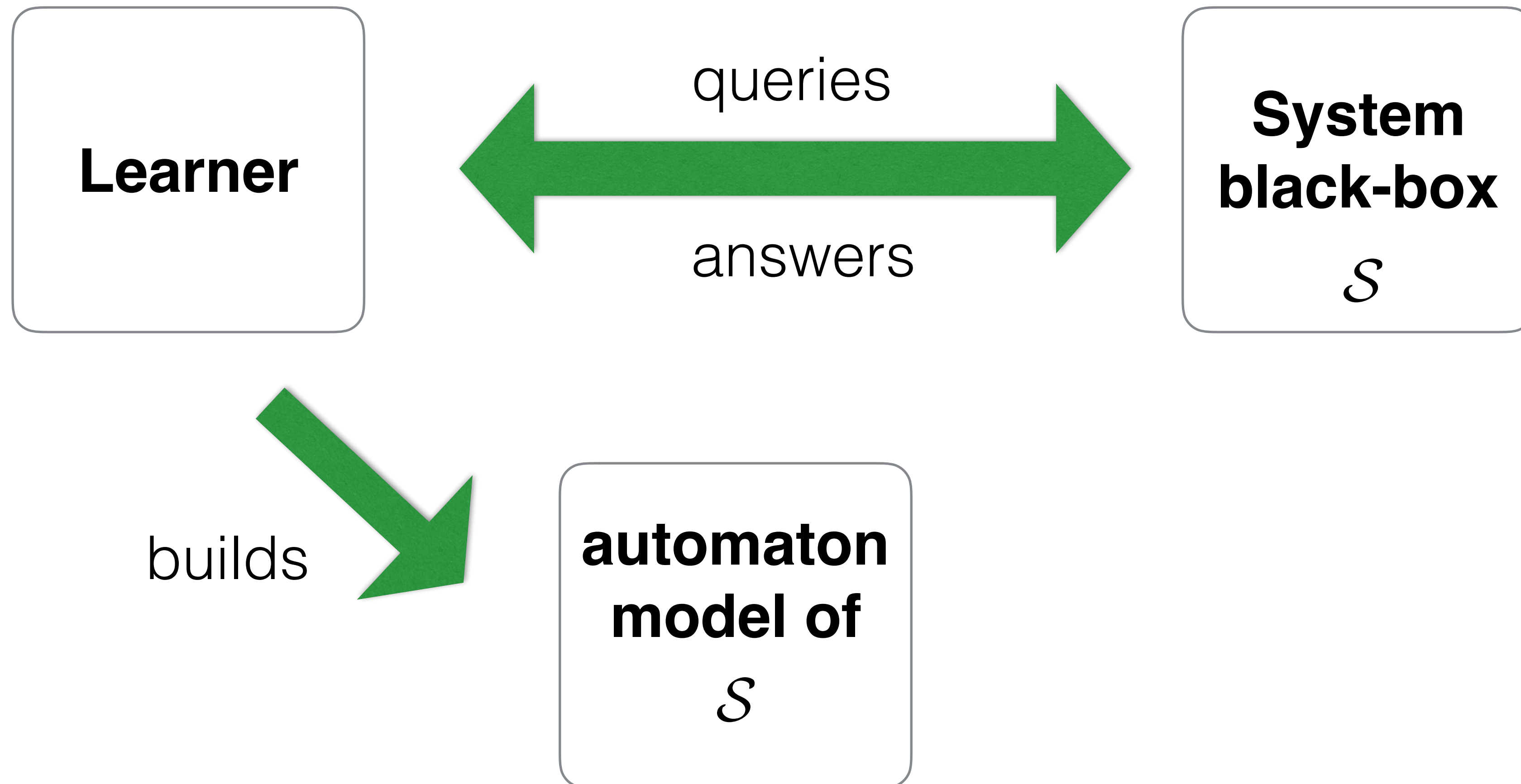


Joshua Moerman
Radboud University

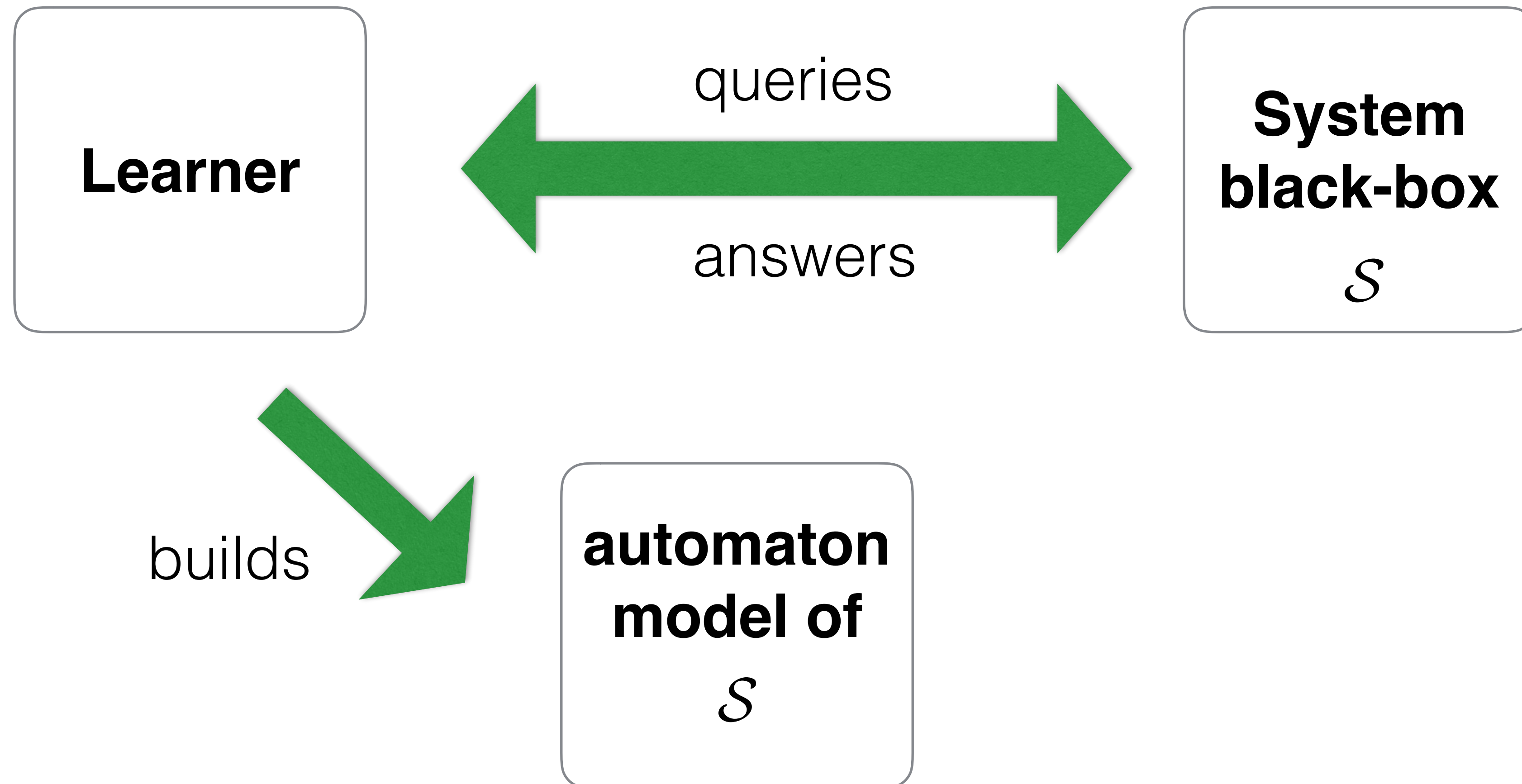


Maverick Chardet
ENS Lyon

Automata learning



Automata learning



No formal specification available? **Learn it!**

L^* algorithm (D.Angluin '87)

Finite alphabet of system's actions A

set of system behaviors is a **regular language** $\mathcal{L} \subseteq A^*$

L^* algorithm (D.Angluin '87)

Finite alphabet of system's actions A

set of system behaviors is a **regular language** $\mathcal{L} \subseteq A^*$

Learner

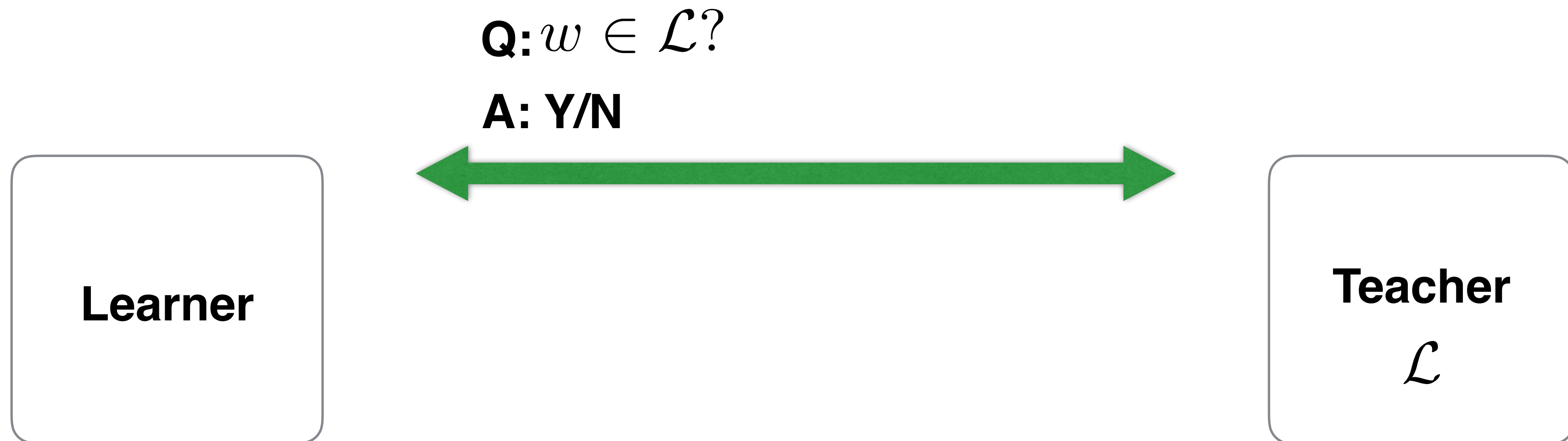
Teacher

\mathcal{L}

L^* algorithm (D.Angluin '87)

Finite alphabet of system's actions A

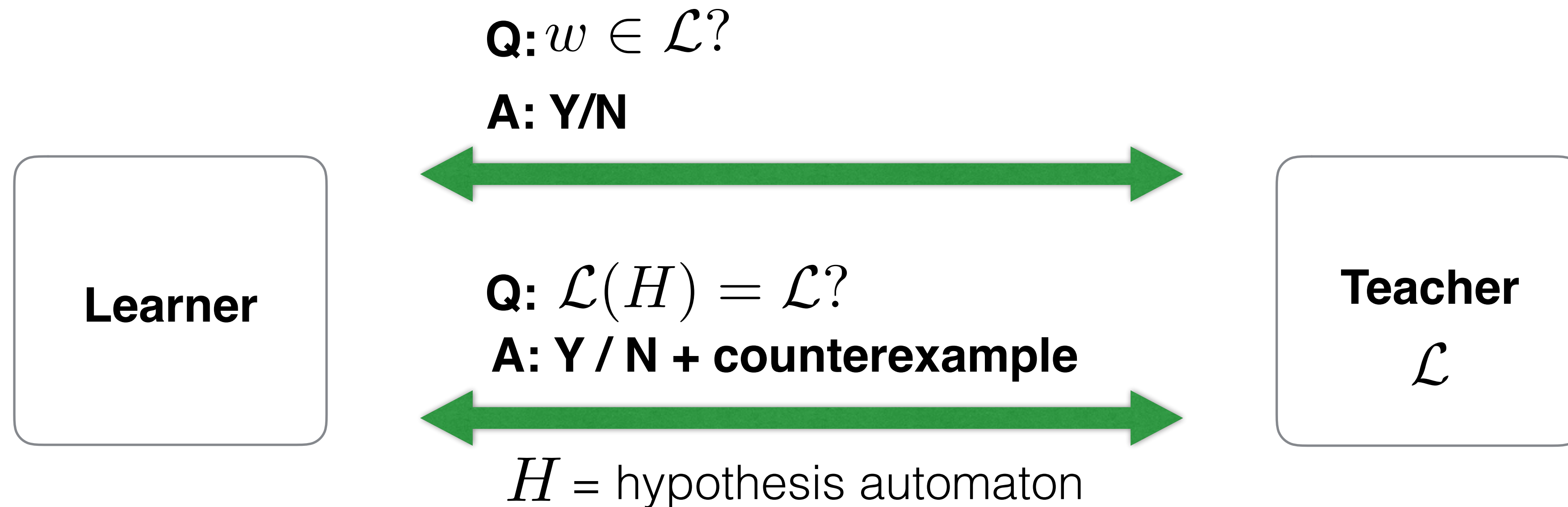
set of system behaviors is a **regular language** $\mathcal{L} \subseteq A^*$



L^* algorithm (D.Angluin '87)

Finite alphabet of system's actions A

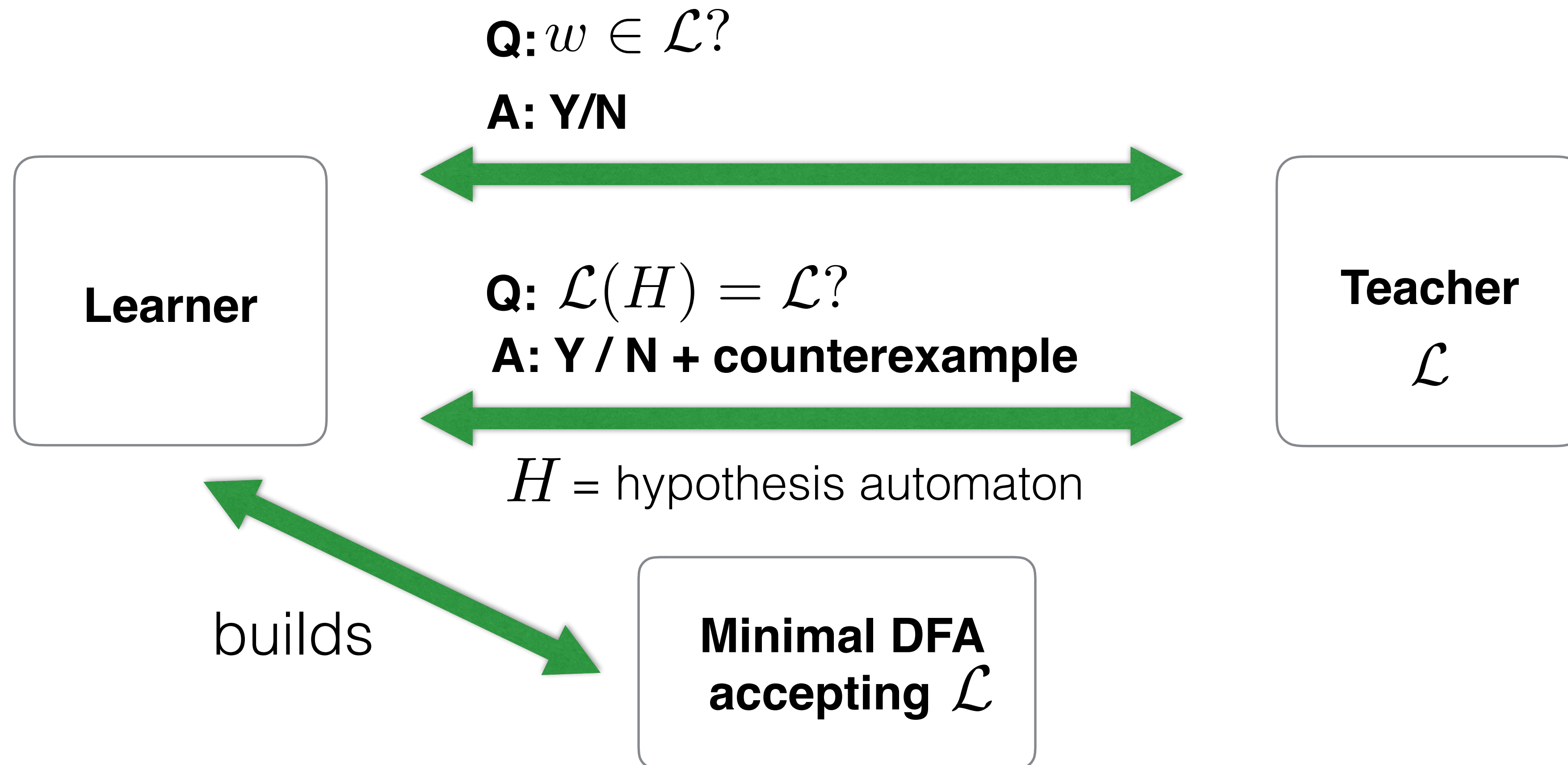
set of system behaviors is a **regular language** $\mathcal{L} \subseteq A^*$



L^* algorithm (D.Angluin '87)

Finite alphabet of system's actions A

set of system behaviors is a **regular language** $\mathcal{L} \subseteq A^*$



A zoo of automata

Probabilistic

Weighted

Alternating

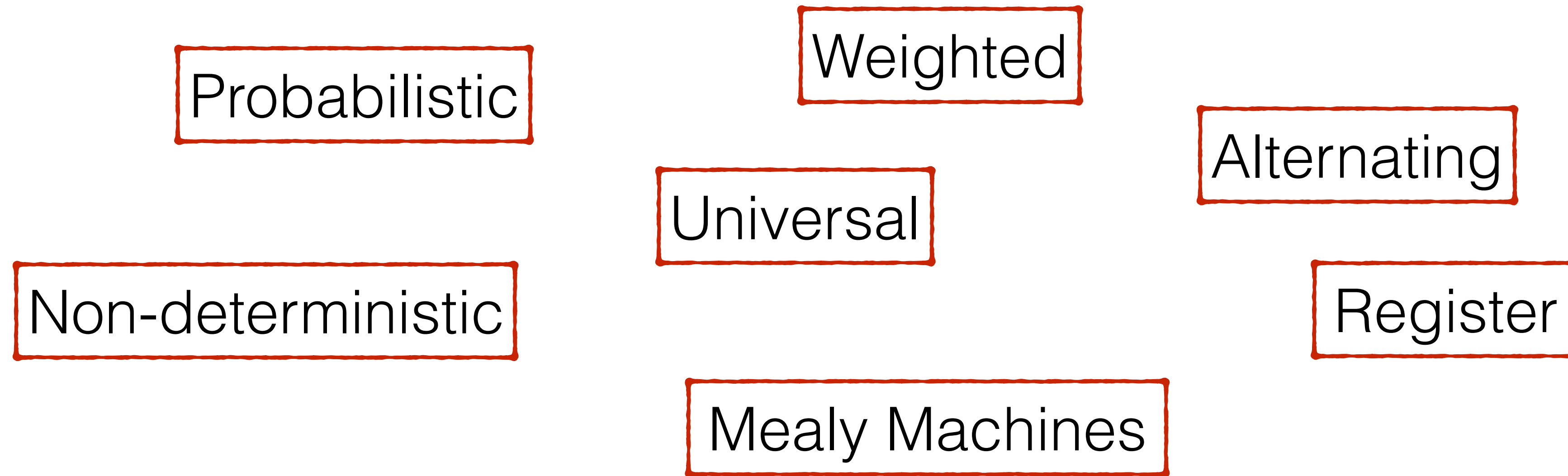
Universal

Non-deterministic

Register

Mealy Machines

A zoo of automata

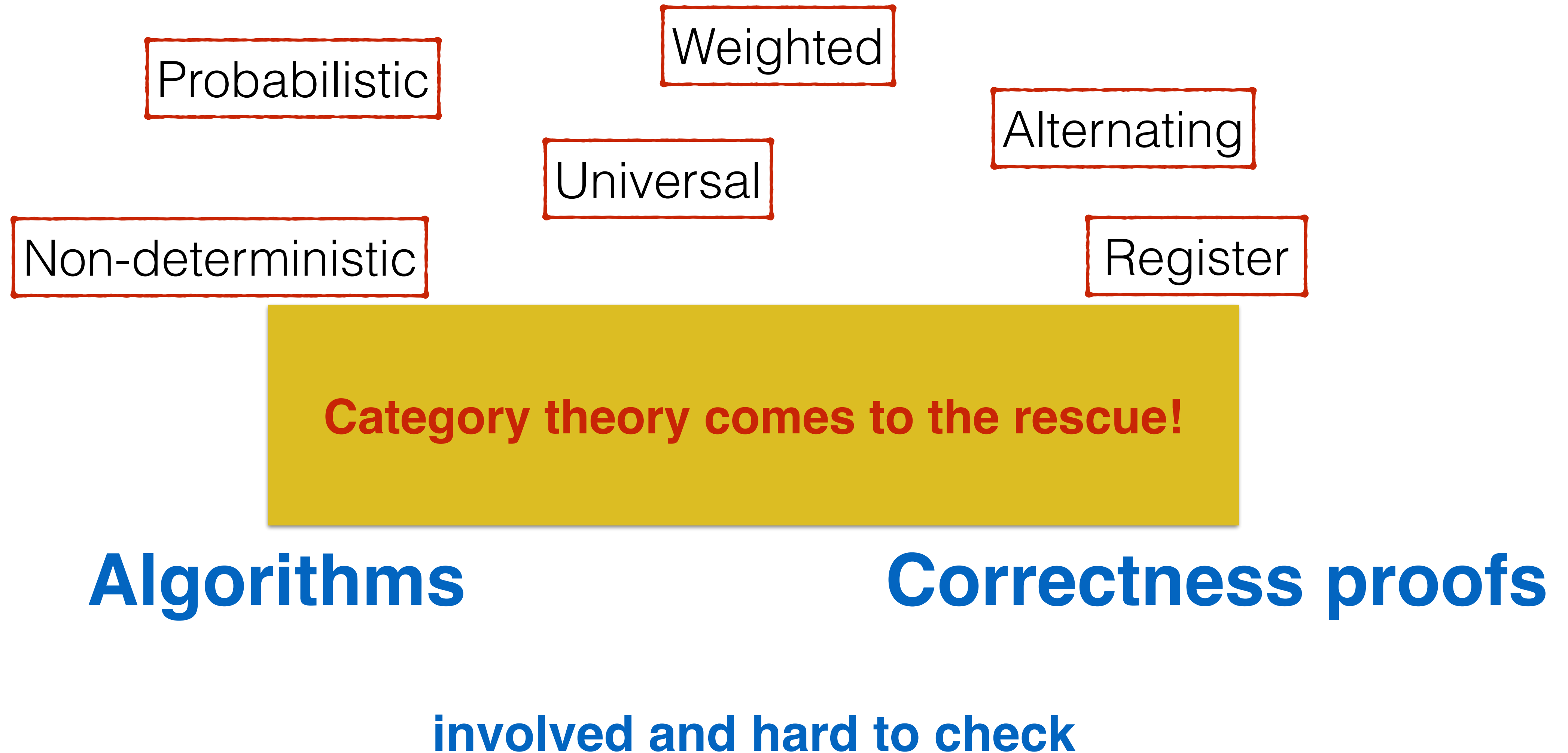


Algorithms

Correctness proofs

involved and hard to check

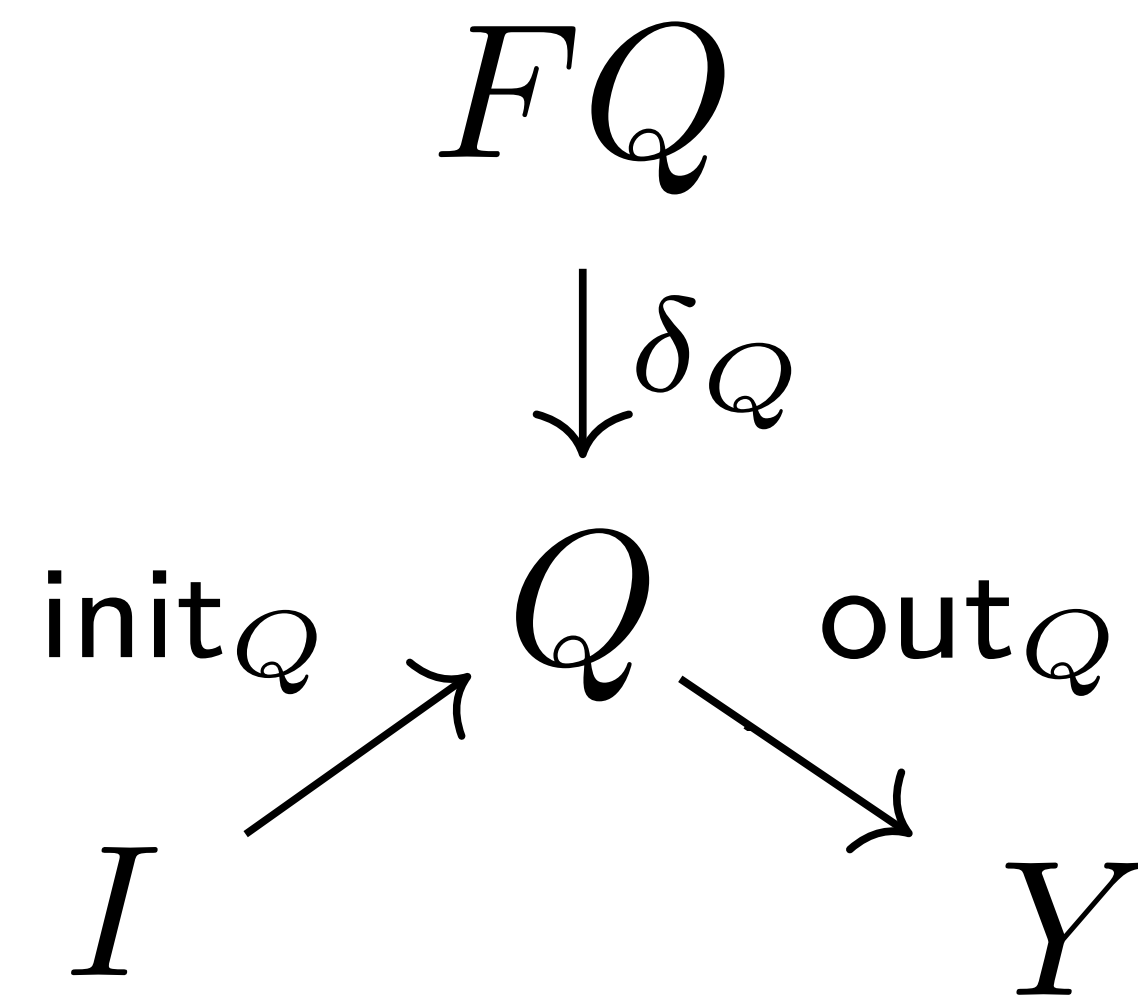
A zoo of automata



Abstract automata

Category \mathbf{C} = universe of state-spaces

Endofunctor $F : \mathbf{C} \rightarrow \mathbf{C}$ = automaton type



Abstract automata

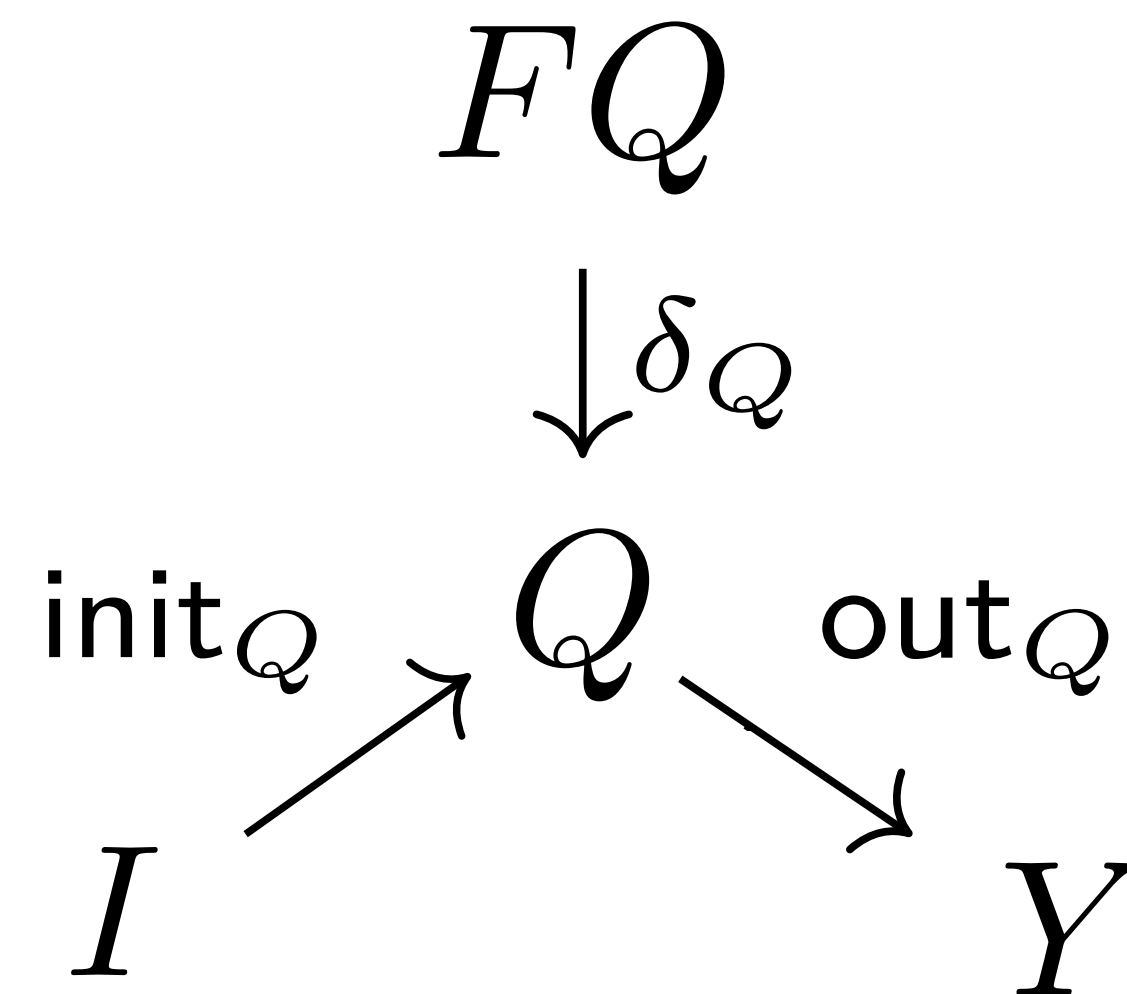
Category \mathbf{C} = universe of state-spaces

Endofunctor $F : \mathbf{C} \rightarrow \mathbf{C}$ = automaton type

DFAs

$\mathbf{C} = \mathbf{Set}$

$F = (-) \times A$



Abstract automata

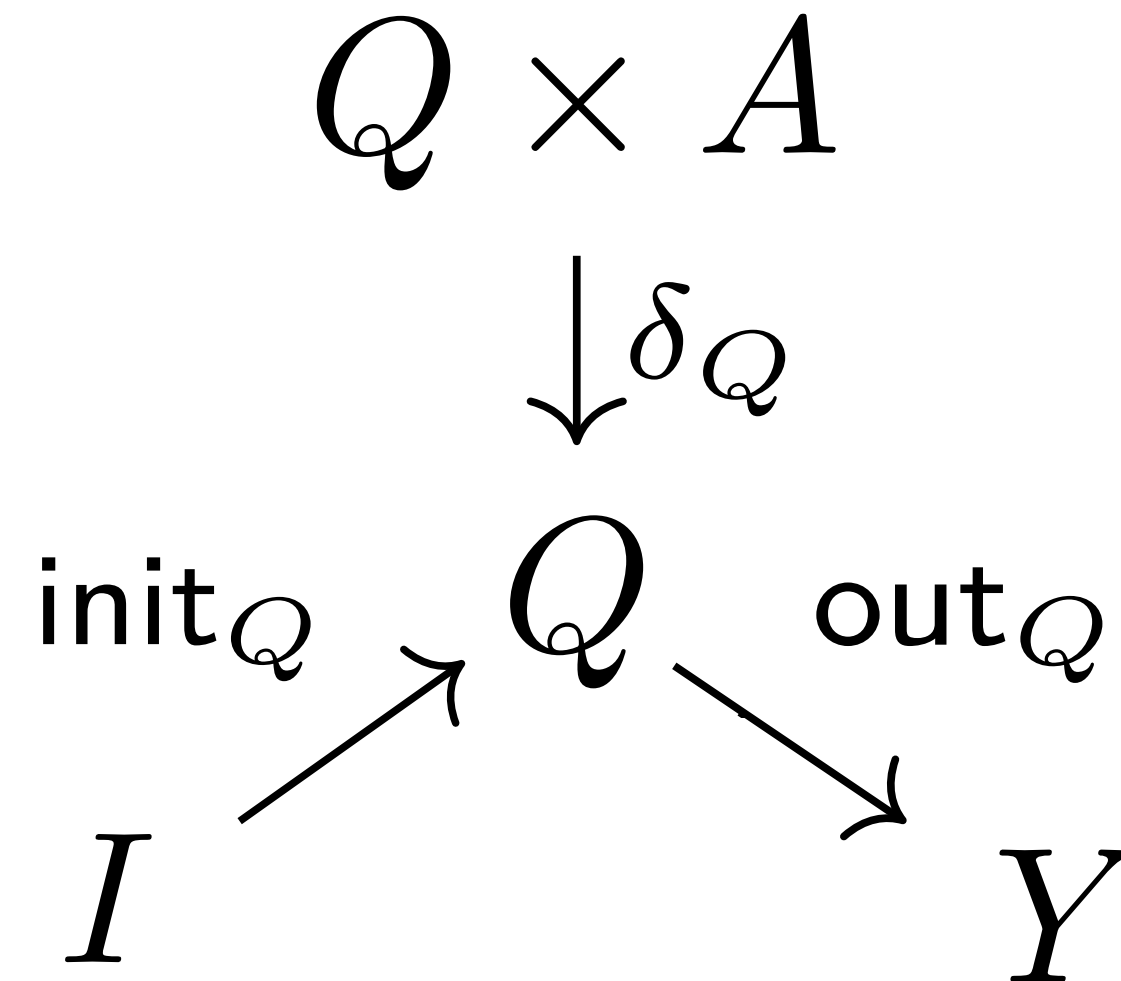
Category \mathbf{C} = universe of state-spaces

Endofunctor $F: \mathbf{C} \rightarrow \mathbf{C}$ = automaton type

DFAs

$\mathbf{C} = \mathbf{Set}$

$F = (-) \times A$



Abstract automata

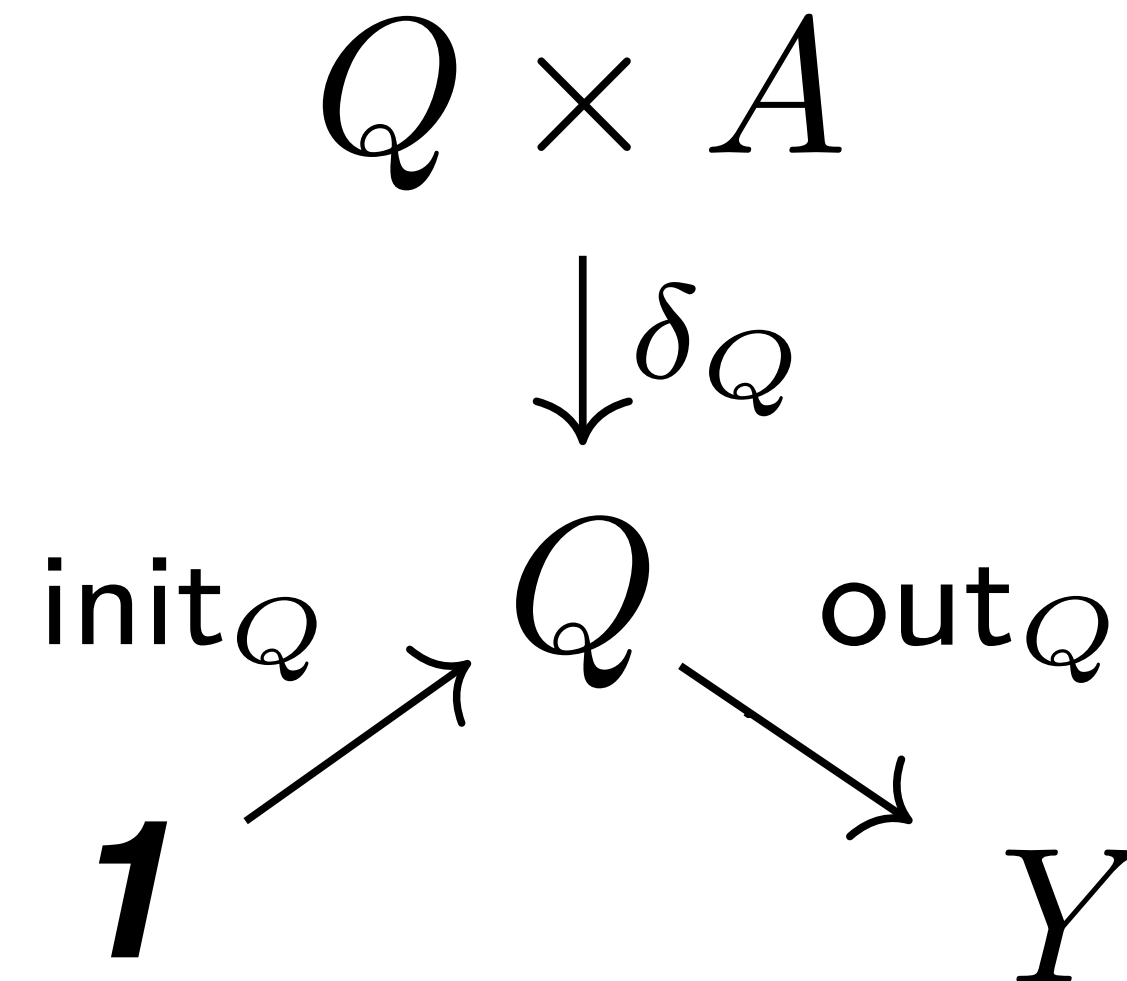
Category \mathbf{C} = universe of state-spaces

Endofunctor $F : \mathbf{C} \rightarrow \mathbf{C}$ = automaton type

DFAs

$\mathbf{C} = \mathbf{Set}$

$F = (-) \times A$



Abstract automata

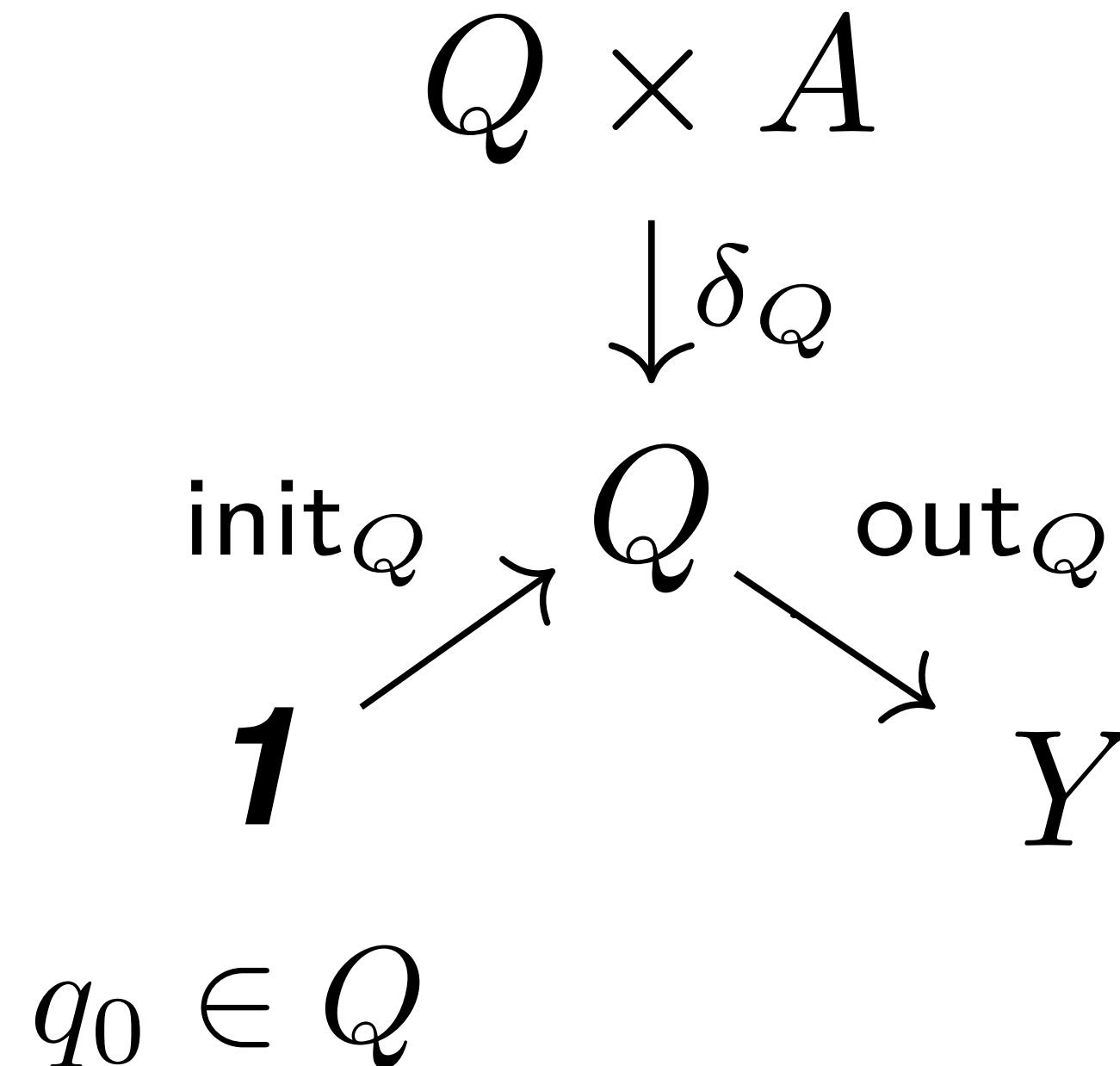
Category \mathbf{C} = universe of state-spaces

Endofunctor $F : \mathbf{C} \rightarrow \mathbf{C}$ = automaton type

DFAs

$\mathbf{C} = \mathbf{Set}$

$F = (-) \times A$



Abstract automata

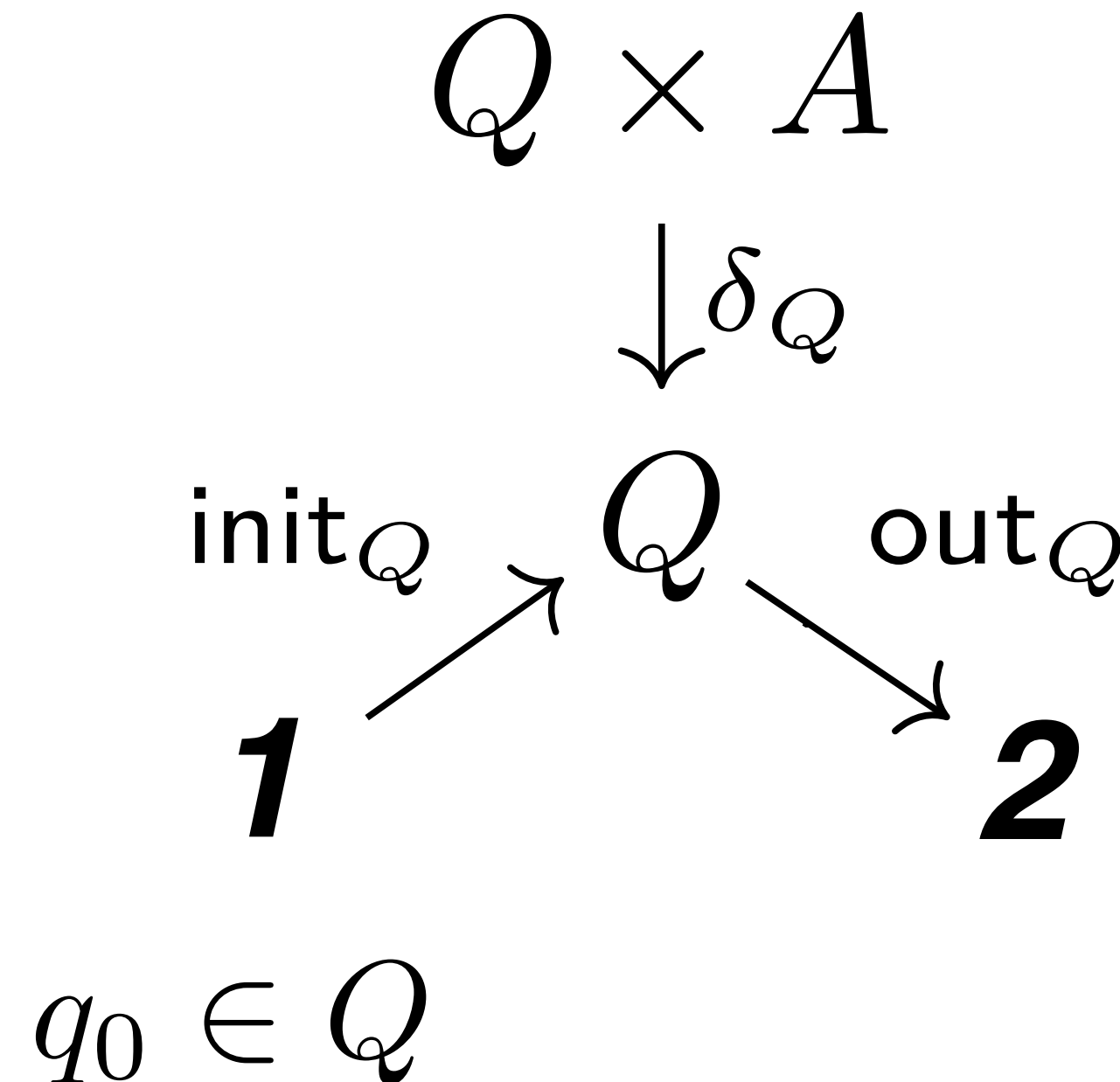
Category \mathbf{C} = universe of state-spaces

Endofunctor $F : \mathbf{C} \rightarrow \mathbf{C}$ = automaton type

DFAs

$\mathbf{C} = \mathbf{Set}$

$F = (-) \times A$



Abstract automata

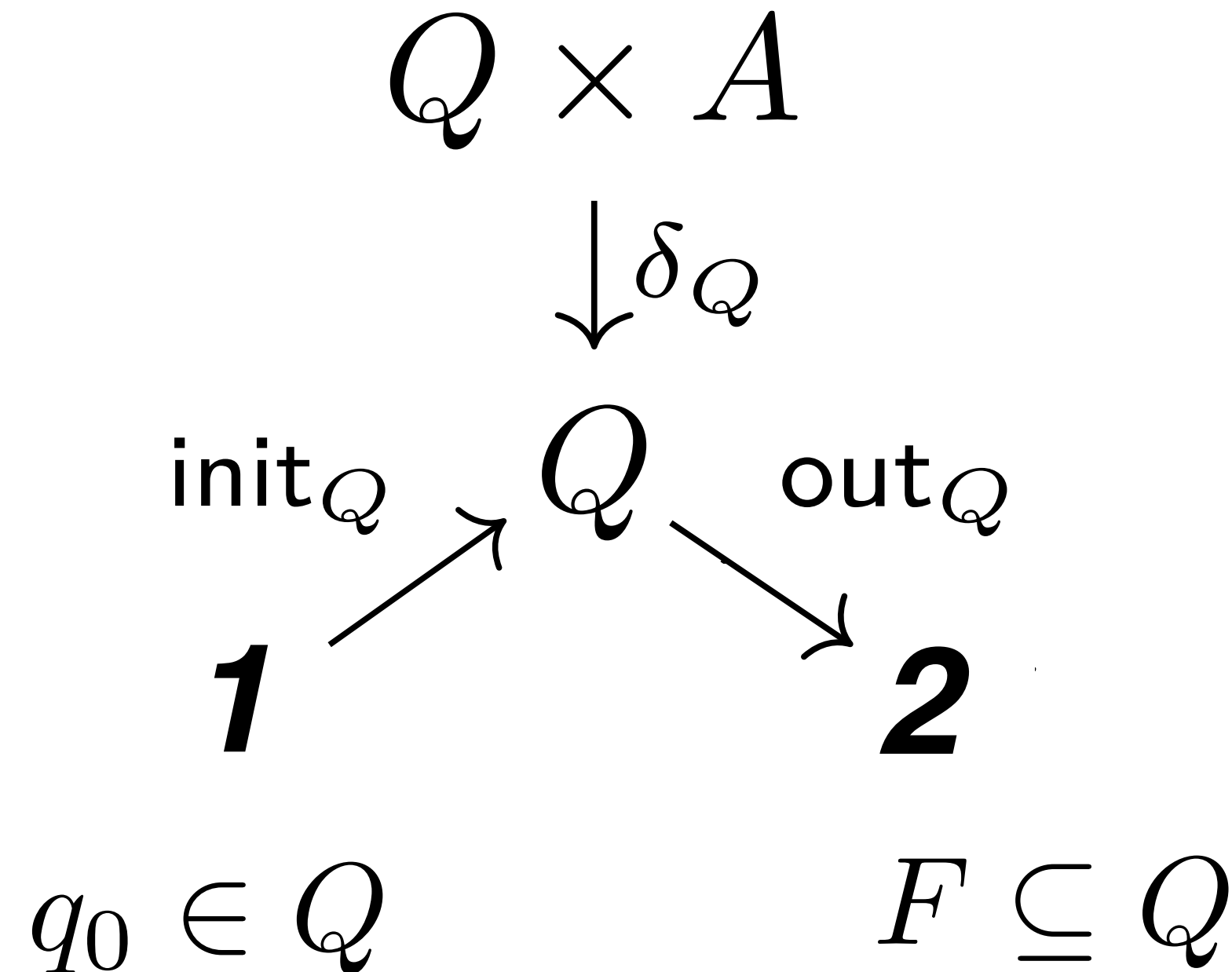
Category \mathbf{C} = universe of state-spaces

Endofunctor $F : \mathbf{C} \rightarrow \mathbf{C}$ = automaton type

DFAs

$\mathbf{C} = \mathbf{Set}$

$F = (-) \times A$



Abstract learning

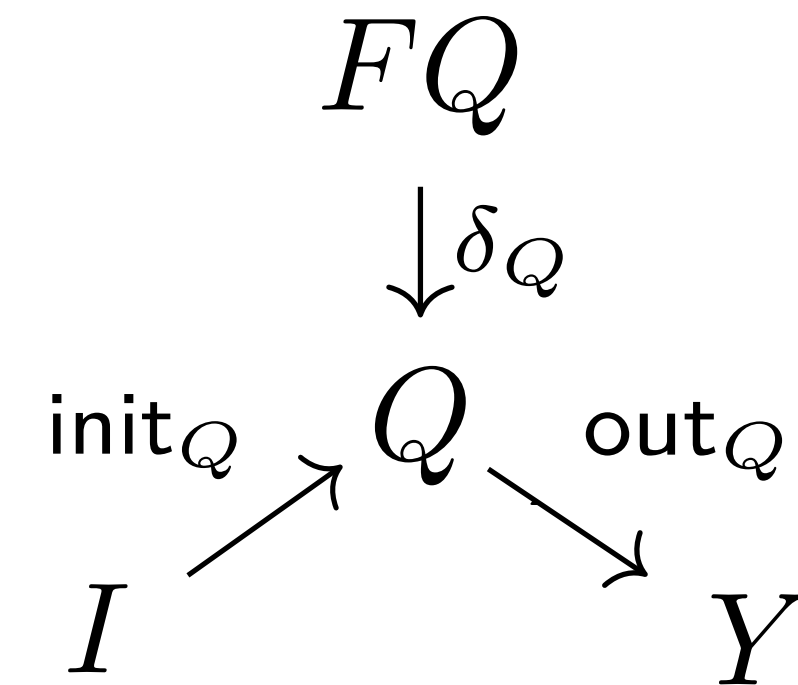
**Abstract observation data
structure**

Abstract learning

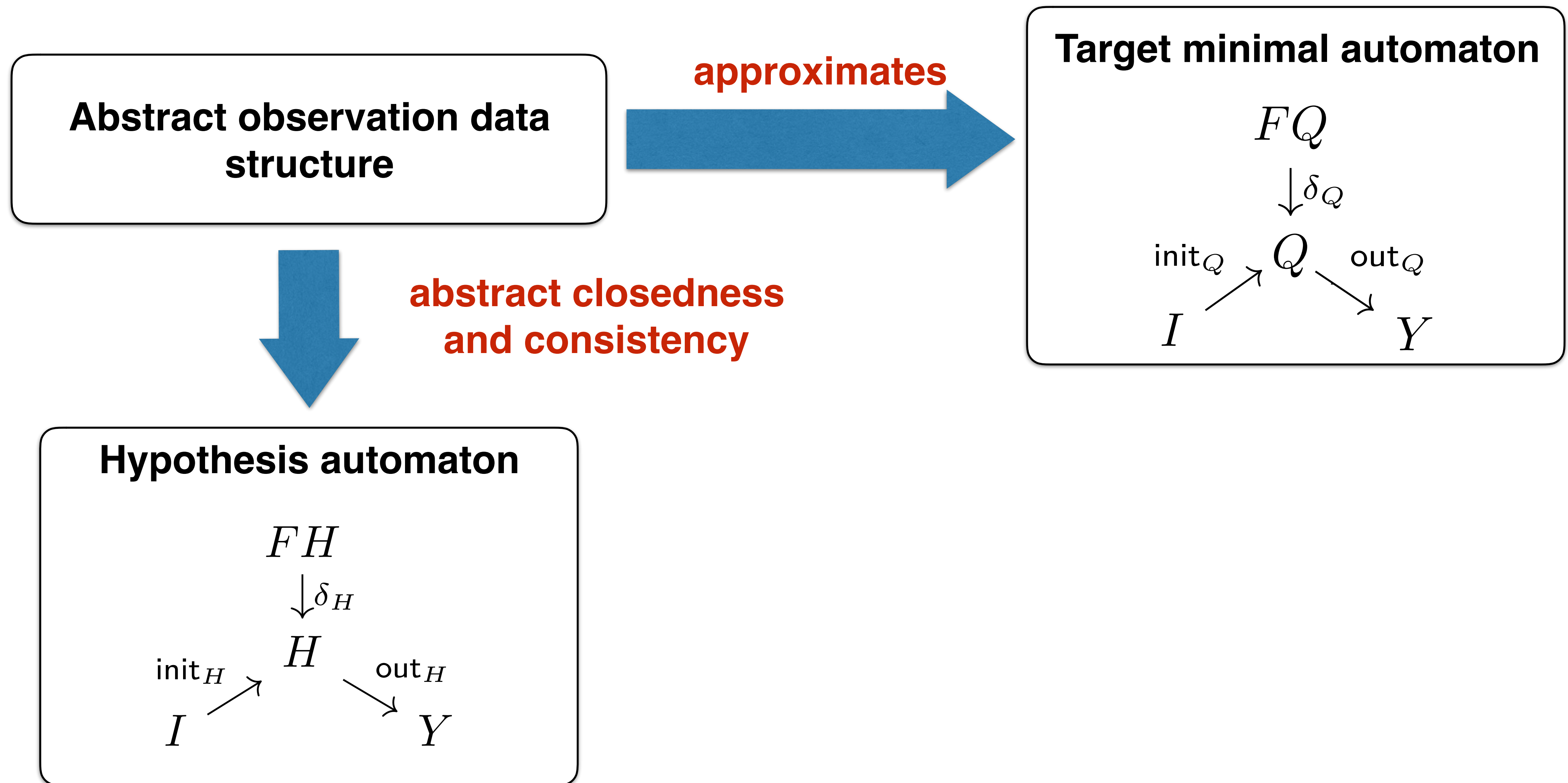
**Abstract observation data
structure**

approximates

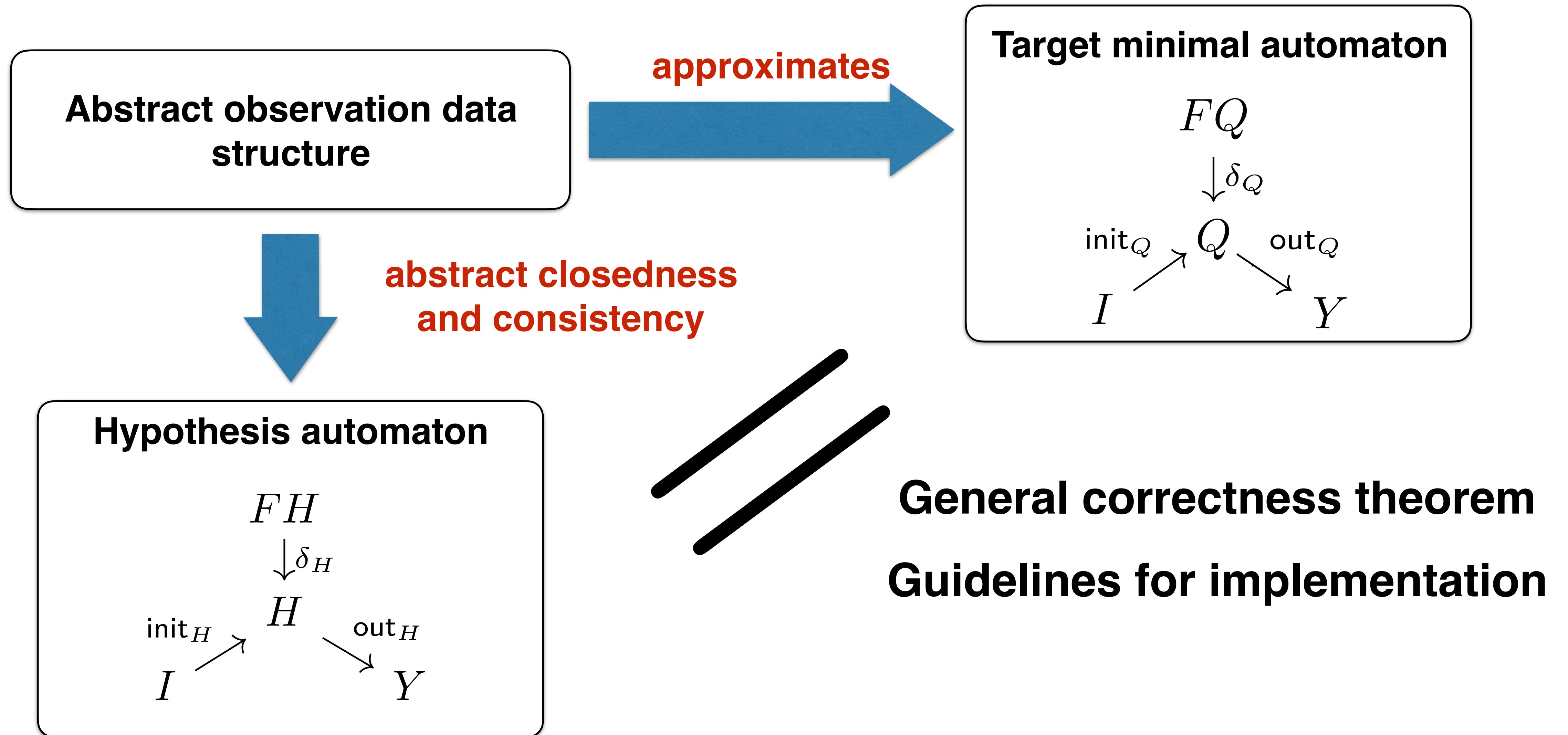
Target minimal automaton



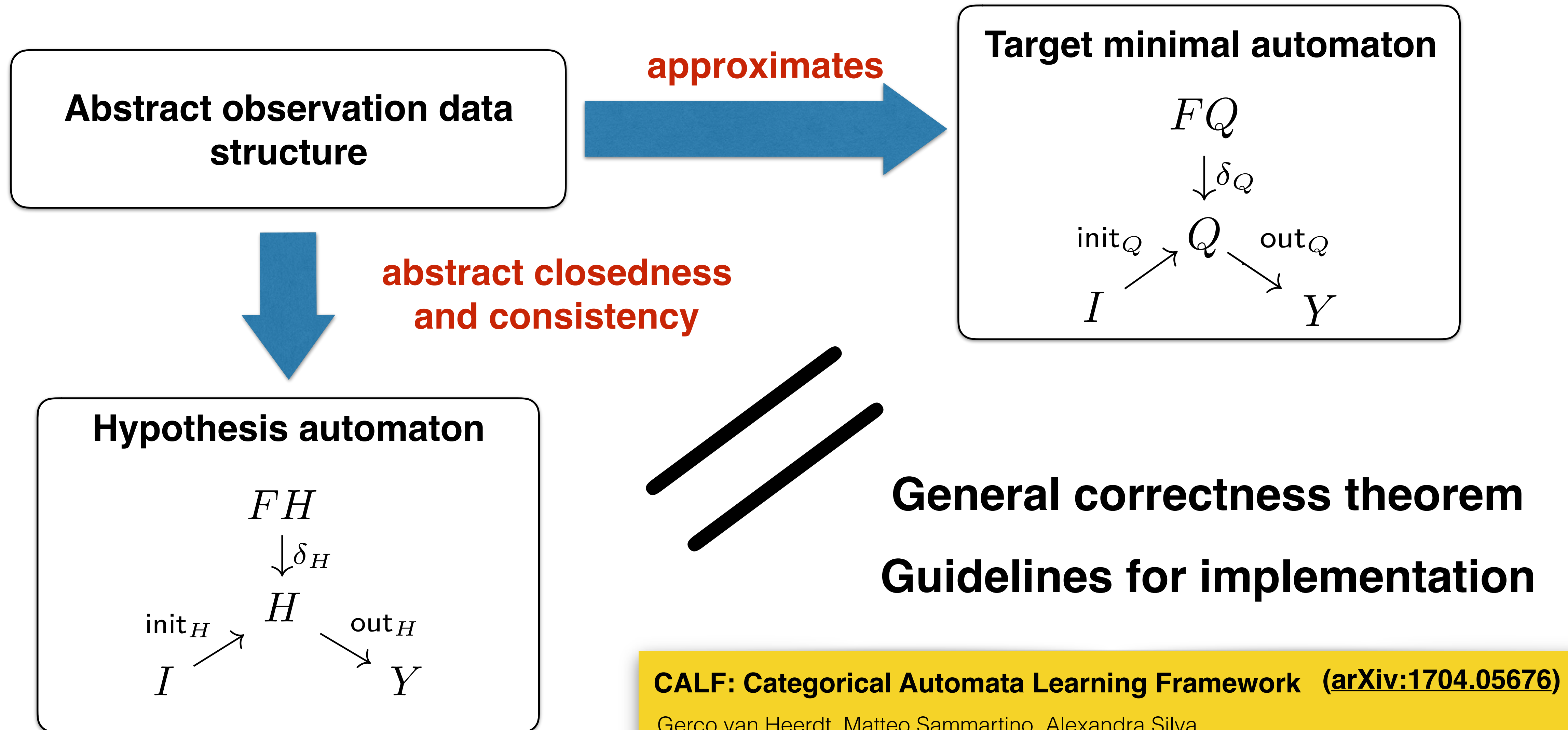
Abstract learning



Abstract learning



Abstract learning



CALF: Categorical Automata Learning Framework ([arXiv:1704.05676](https://arxiv.org/abs/1704.05676))

Gerco van Heerdt, Matteo Sammartino, Alexandra Silva

Other automata & optimizations

Other automata & optimizations

Change base category

Set **DFAs**

Nom **Nominal automata**

Vect **Weighted automata**

Other automata & optimizations

Change base category

Set	DFAs
Nom	Nominal automata
Vect	Weighted automata

Side-effects (via monads)

Powerset	NFAs
Powerset with intersection	Universal automata
Double powerset	Alternating automata
Maybe monad	Partial automata

Other automata & optimizations

Change base category

Set	DFAs
Nom	Nominal automata
Vect	Weighted automata

Change main data structure

Observation tables
Discrimination trees

Side-effects (via monads)

Powerset	NFAs
Powerset with intersection	Universal automata
Double powerset	Alternating automata
Maybe monad	Partial automata

Other automata & optimizations

Change base category

Change main data structure

Set

DFAs

Learning Nominal Automata (POPL '17)

Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, Michal Szynwelski

Nom

Nominal automata

Discrimination trees

Vect

Weighted automata

Side-effects (via monads)

Powerset **NFAs**

Powerset with intersection **Universal automata**

Double powerset **Alternating automata**

Maybe monad **Partial automata**

Other automata & optimizations

Change base category

Change main data structure

Set

DFAs

Learning Nominal Automata (POPL '17)

Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, Michal Szyrwelski

Nom

Nominal automata

Discrimination trees

Vect

Weighted automata

Learning Automata with Side-effects ([arXiv:1704.08055](https://arxiv.org/abs/1704.08055))

Gerco van Heerdt, Matteo Sammartino, Alexandra Silva

Side-effects (via monads)

Powerset **NFAs**

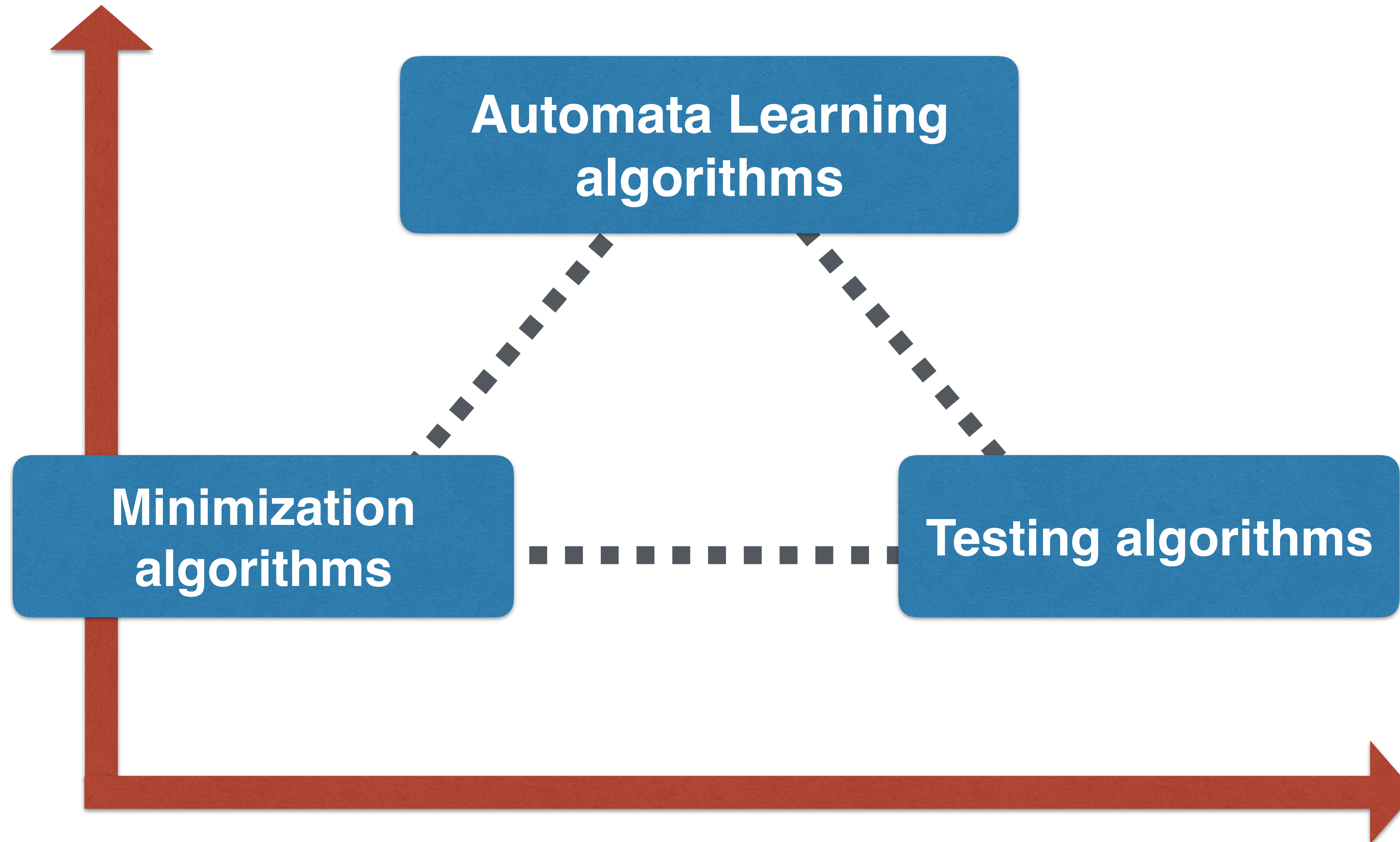
Powerset with intersection **Universal automata**

Double powerset **Alternating automata**

Maybe monad **Partial automata**


Connections with other algorithms

Automaton type

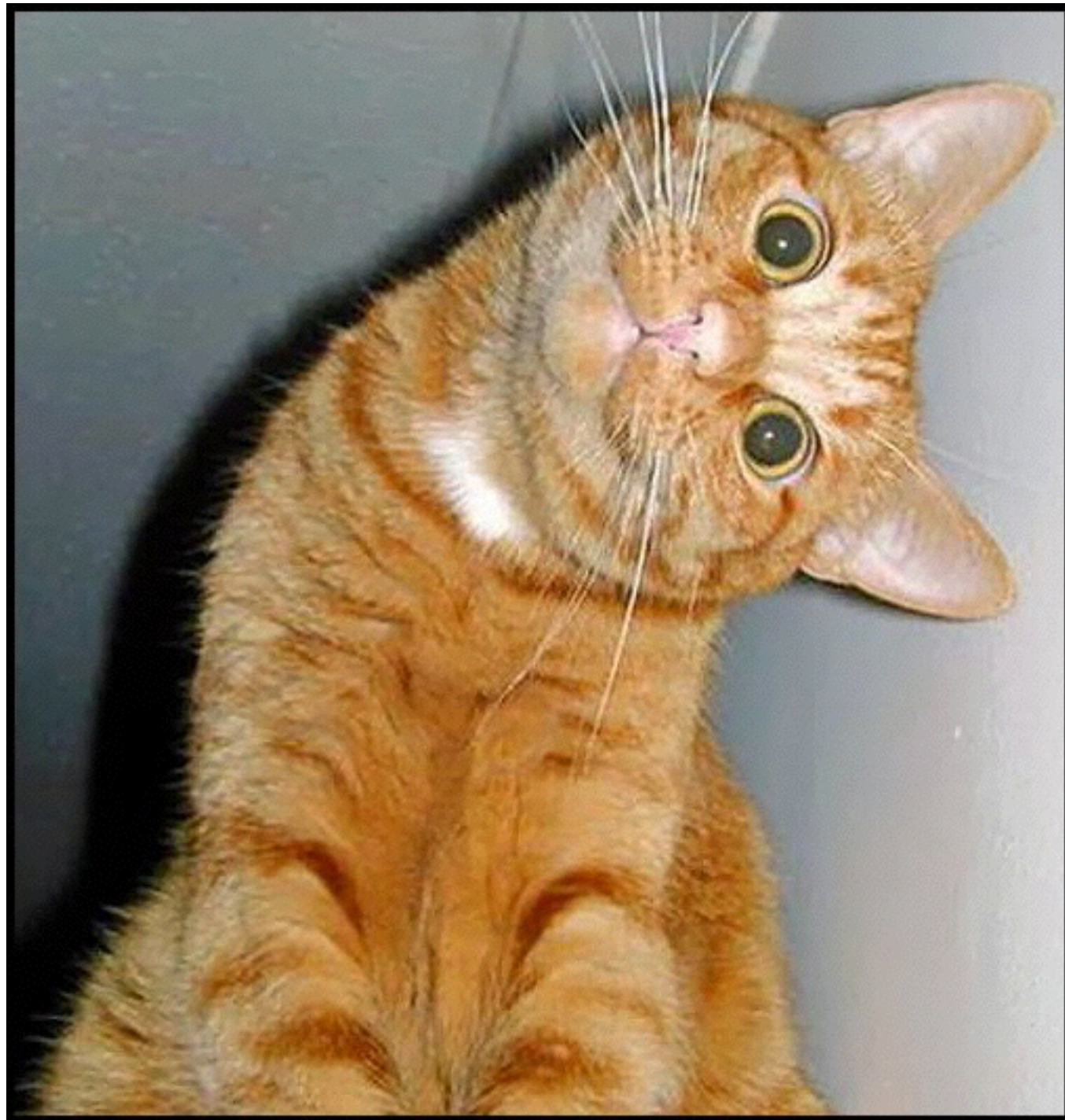


Optimizations

Ongoing and future work

- **Library & tool** to learn control + data-flow models (as **nominal automata**)
- Applications:
 - Specification mining
 - Network verification, with 
 - Verification of cryptographic protocols
 - Ransomware detection

What about succinct
acceptors?

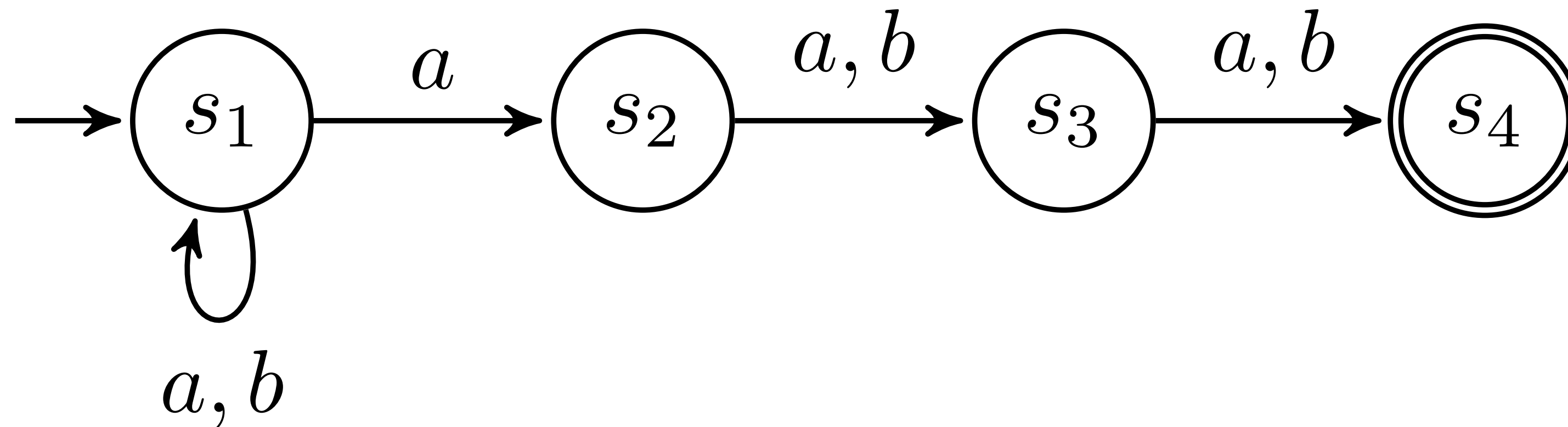


Back to basics

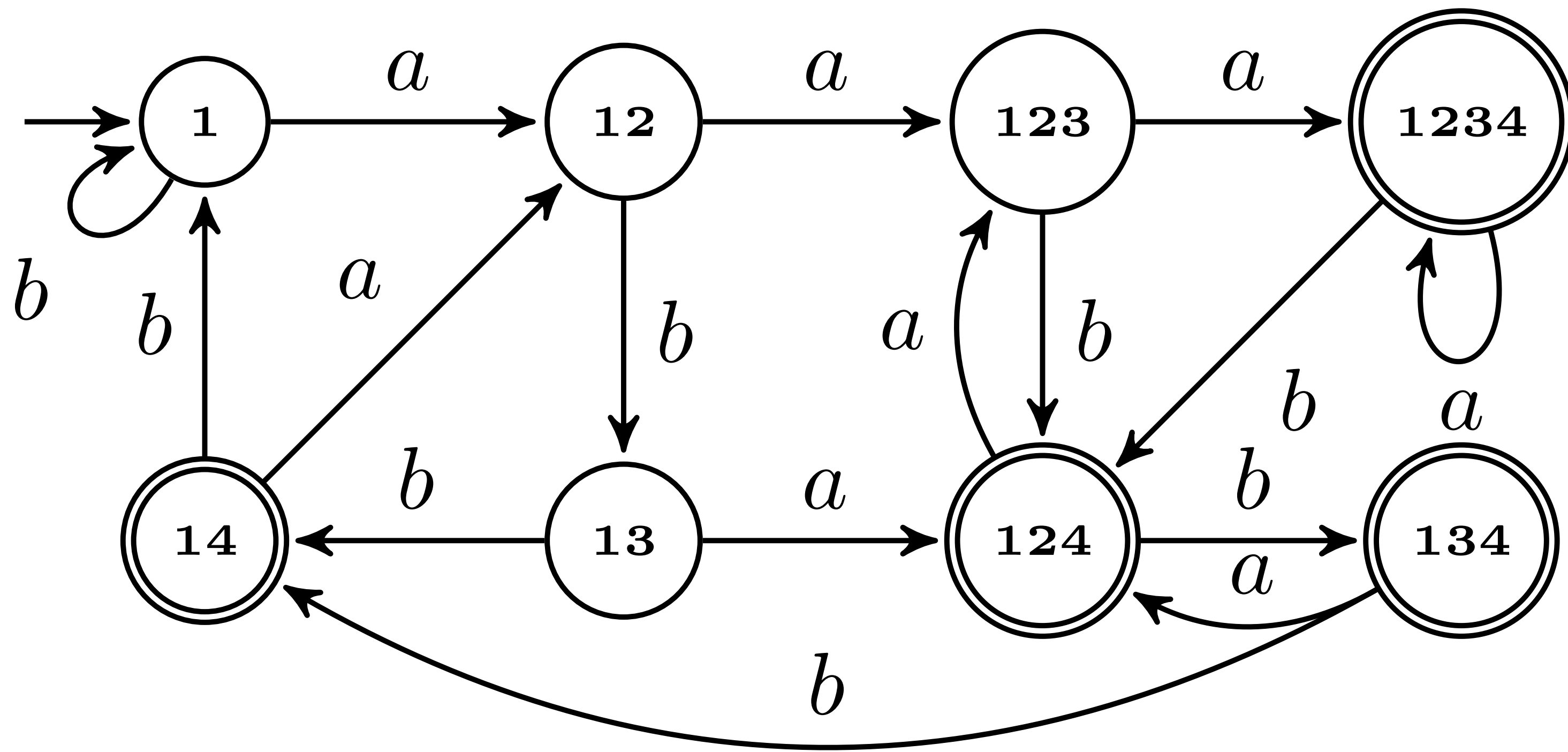
$$\mathcal{L} = \{w \in \{a, b\}^* \mid |w| > 2 \text{ and the } 3^{rd} \text{ letter from the right is an } a\}$$

Back to basics

$\mathcal{L} = \{w \in \{a, b\}^* \mid |w| > 2 \text{ and the } 3^{rd} \text{ letter from the right is an } a\}$



Back to basics



Subset construction

$$\begin{array}{ccccc}
 X & \xrightarrow{\{\cdot\}} & \mathcal{P}(X) & \xrightarrow{l} & 2^{A^*} \\
 \downarrow \delta & & \swarrow \delta^\# & & \downarrow \omega \\
 2 \times \mathcal{P}(X)^A & \xrightarrow{id \times l^A} & & & 2 \times (2^{A^*})^A
 \end{array}$$

Generalised powerset construction

$$\begin{array}{ccccc}
 X & \xrightarrow{\eta} & TX & \xrightarrow{l} & \Omega \\
 \downarrow \delta & & \searrow \delta^\# & & \downarrow \omega \\
 FTX & \xrightarrow{\quad F l \quad} & & & F\Omega
 \end{array}$$

Other examples : partial, probabilistic, and weighted automata

Rich algebraic structure

$$\begin{array}{ccccc}
 X & \xrightarrow{\{\cdot\}} & \mathcal{P}(X) & \xrightarrow{l} & 2^{A^*} \\
 \downarrow \delta & & \searrow \delta^\# & & \downarrow \omega \\
 2 \times \mathcal{P}(X)^A & \xrightarrow{id \times l^A} & & & 2 \times (2^{A^*})^A
 \end{array}$$

JSL map

Rich algebraic structure

$$\begin{array}{ccccc}
 X & \xrightarrow{\{\cdot\}} & \mathcal{P}(X) & \xrightarrow{l} & 2^{A^*} \\
 \downarrow \delta & & \searrow \delta^\# & & \downarrow \omega \\
 2 \times \mathcal{P}(X)^A & \xrightarrow{id \times l^A} & 2 \times (2^{A^*})^A & &
 \end{array}$$

JSL map

$$\begin{array}{ccccc}
 X & \xrightarrow{\eta} & TX & \xrightarrow{l} & \Omega \\
 \downarrow \delta & & \searrow \delta^\# & & \downarrow \omega \\
 FTX & \xrightarrow{Fl} & F\Omega & &
 \end{array}$$

T-algebra map

Rich algebraic structure

$$\begin{array}{ccccc}
 X & \xrightarrow{\{\cdot\}} & \mathcal{P}(X) & \xrightarrow{l} & 2^{A^*} \\
 \downarrow \delta & & \searrow \delta^\# & & \downarrow \omega \\
 2 \times \mathcal{P}(X)^A & \xrightarrow{id \times l^A} & 2 \times (2^{A^*})^A & &
 \end{array}$$

T-algebras

\mathbb{R}^-	Vector Spaces
\mathbb{S}^-	Semimodules
$1 + -$	Pointed sets
$\mathcal{D}(-)$	Convex sets

$$\begin{array}{ccccc}
 X & \xrightarrow{\eta} & TX & \xrightarrow{l} & \Omega \\
 \downarrow \delta & & \searrow \delta^\# & & \downarrow \omega \\
 FTX & \xrightarrow{Fl} & F\Omega & &
 \end{array}$$

T-algebra map

Up-to techniques

Algebraic structure



Better Proof Techniques

Up-to techniques

Algebraic structure

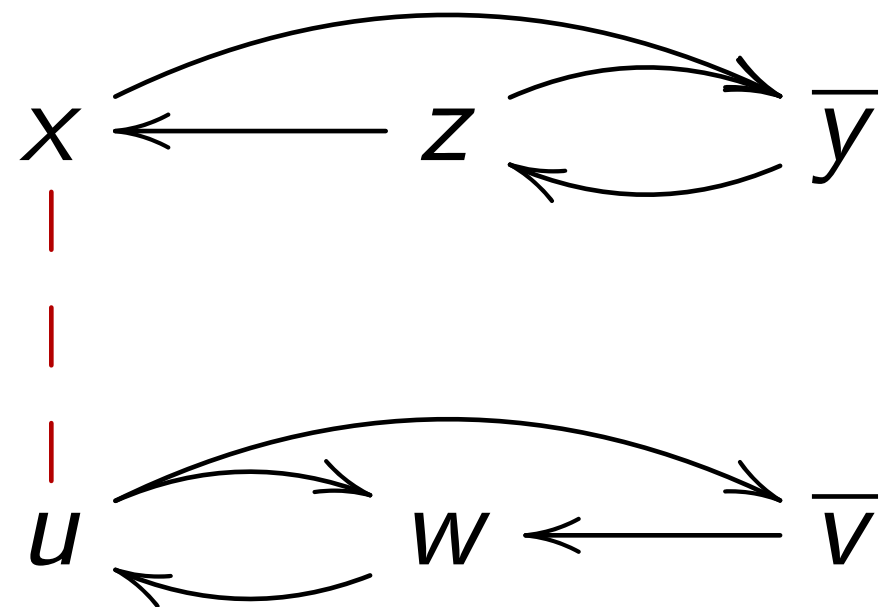


Better Proof Techniques

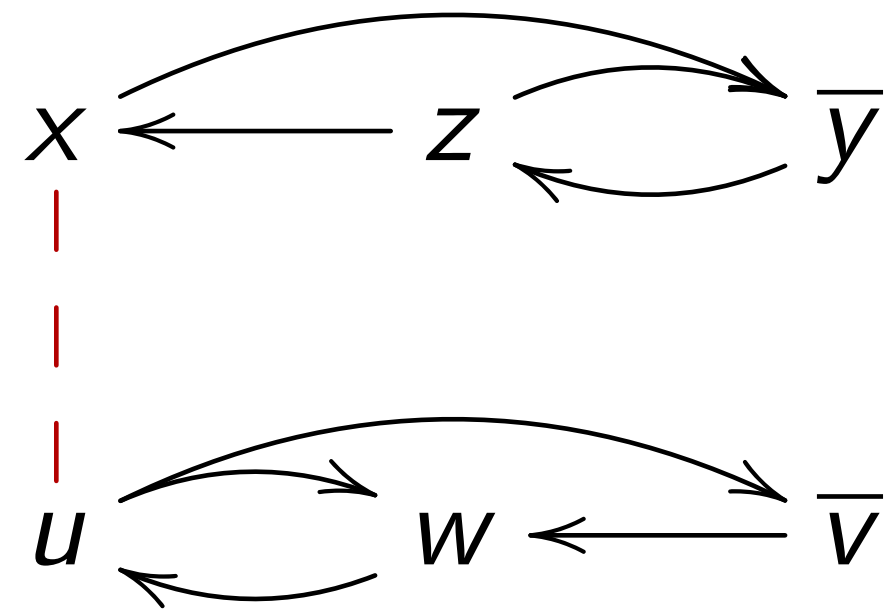


HKC algorithm - Bonchi and Pous 2014

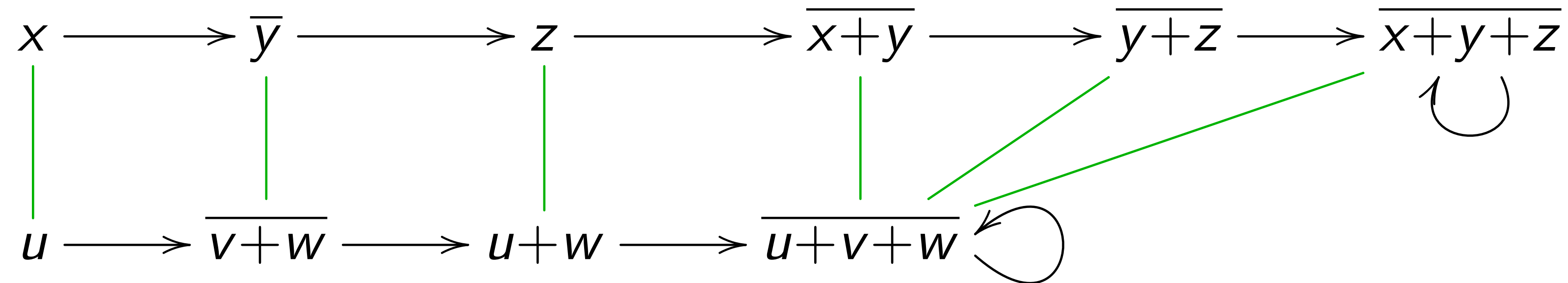
Example



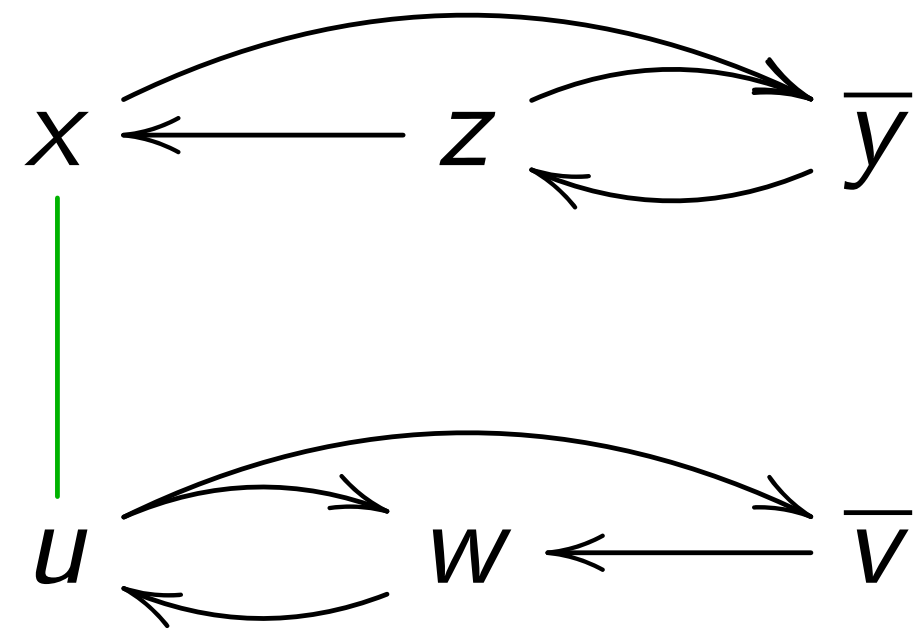
Example



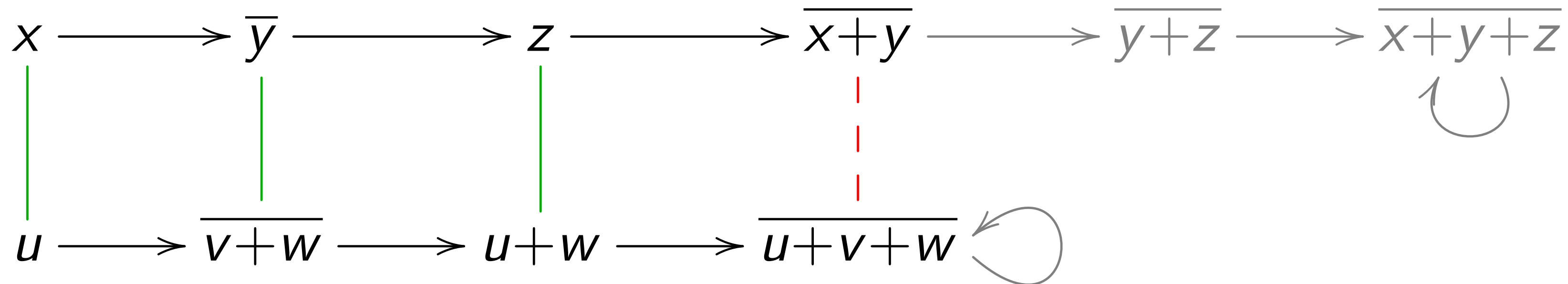
Build a bisimulation using
powerset construction on the fly



Example

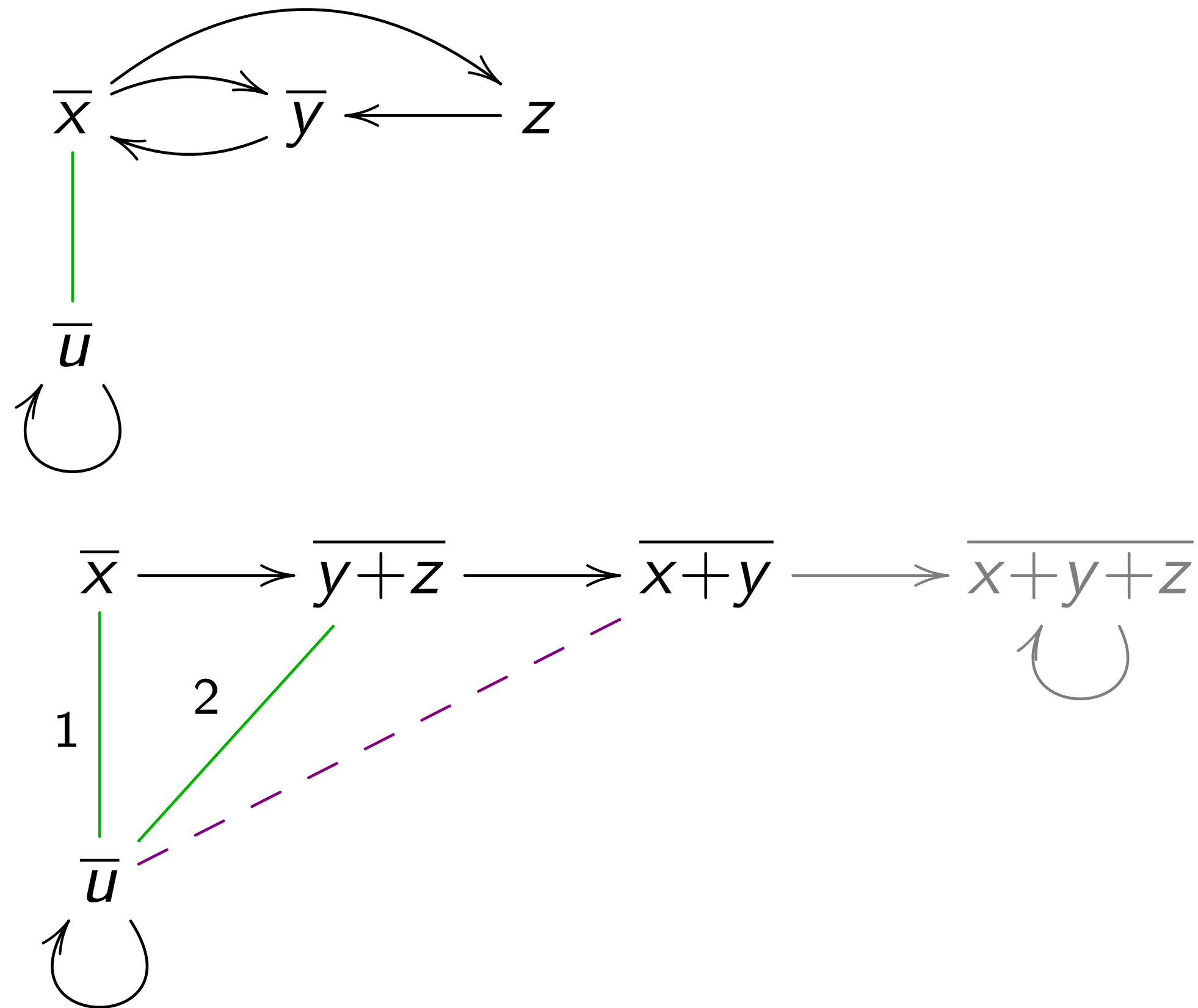


$$\begin{array}{r} (x, u) \\ + \quad (y, v+w) \\ \hline = \quad (x+y, u+v+w) \end{array}$$

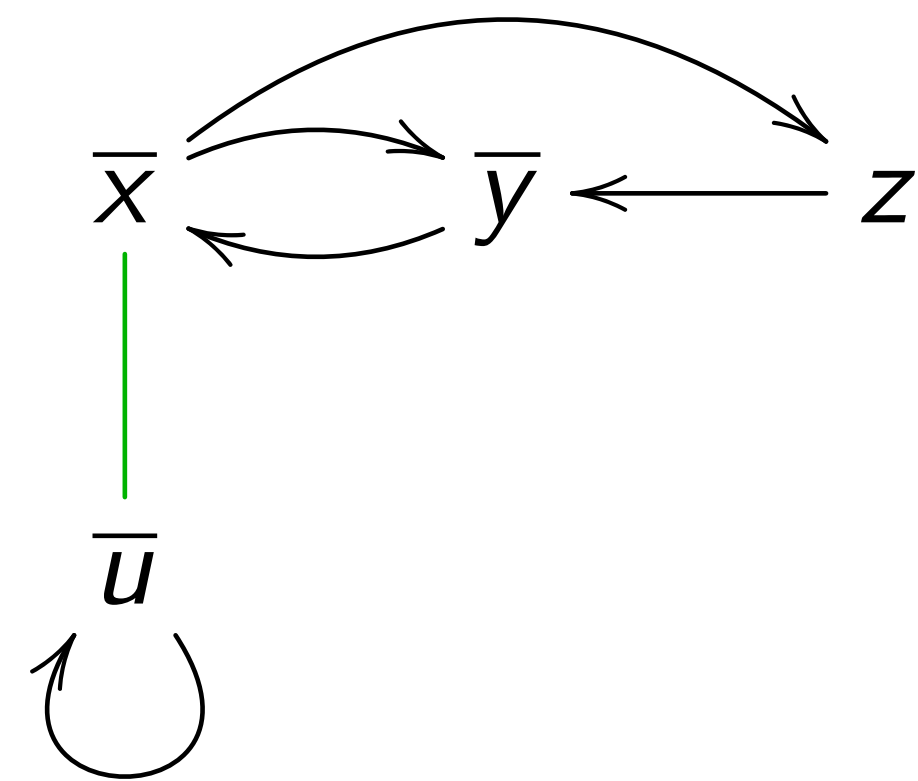


using bisimulations up to union

Another example



Another example

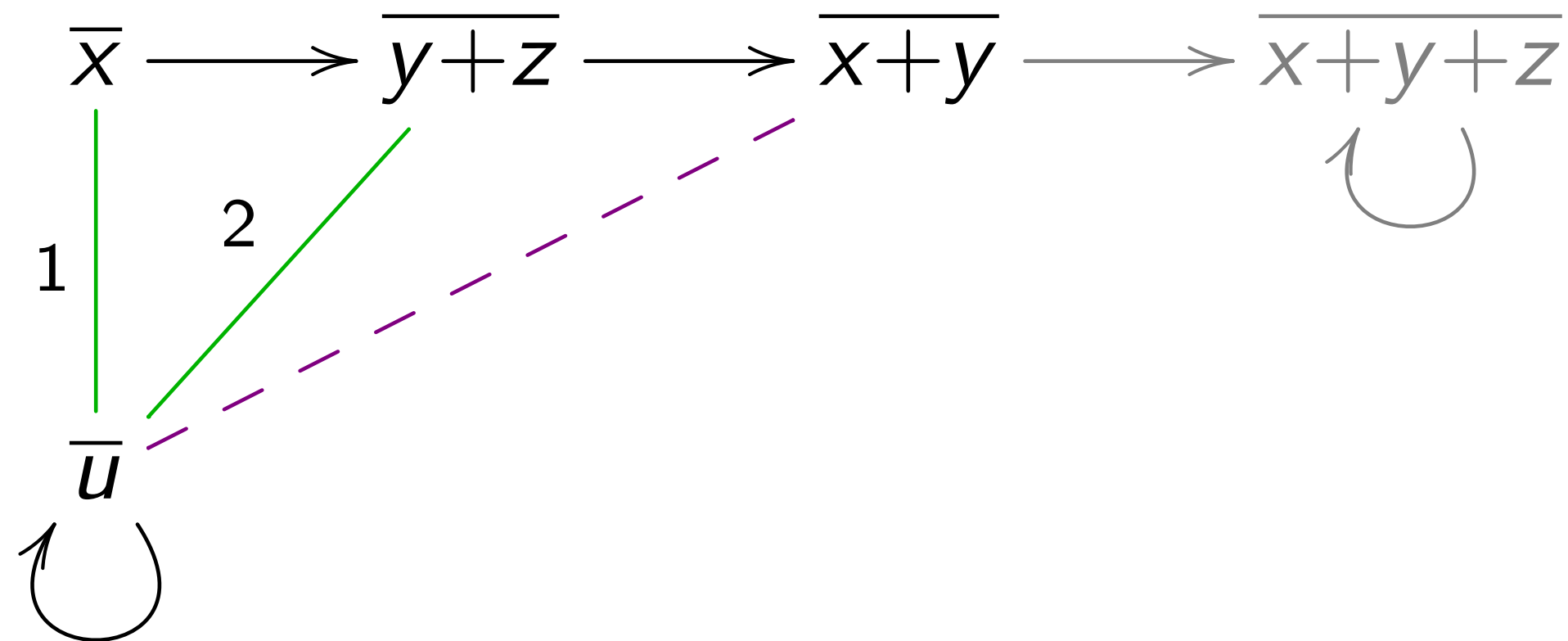


$$x+y = u+y \quad (1)$$

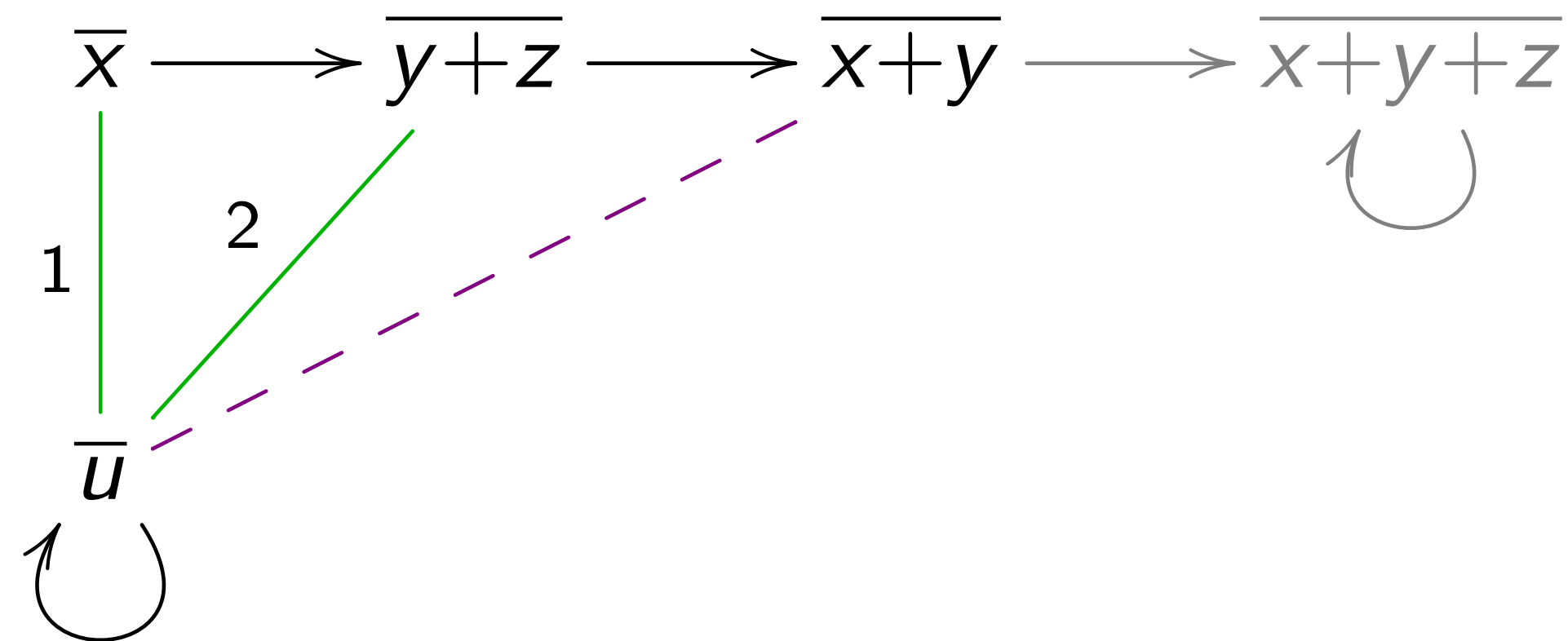
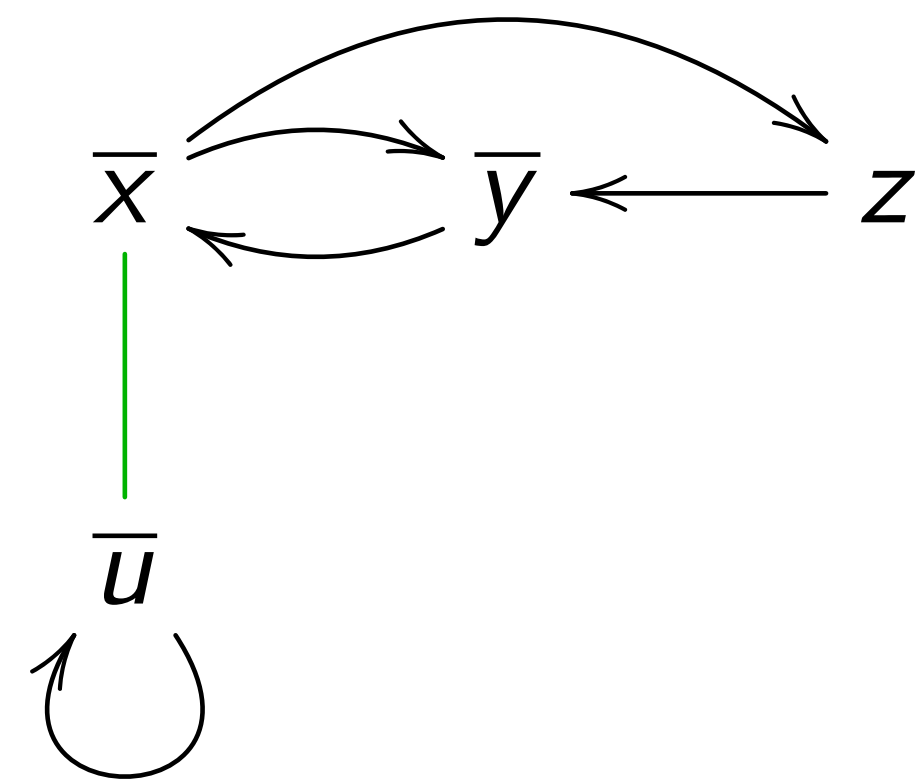
$$= y+z+y \quad (2)$$

$$= y+z$$

$$= u \quad (2)$$



Another example



$$x+y = u+y \quad (1)$$

$$= y+z+y \quad (2)$$

$$= y+z$$

$$= u \quad (2)$$

Bisimulations up-to **congruence**
HKC algorithm of Bonchi&Pous

More examples

Up-To Techniques for Weighted Systems. (TACAS '17)

Filippo Bonchi, Barbara König, Sebastian Küpper

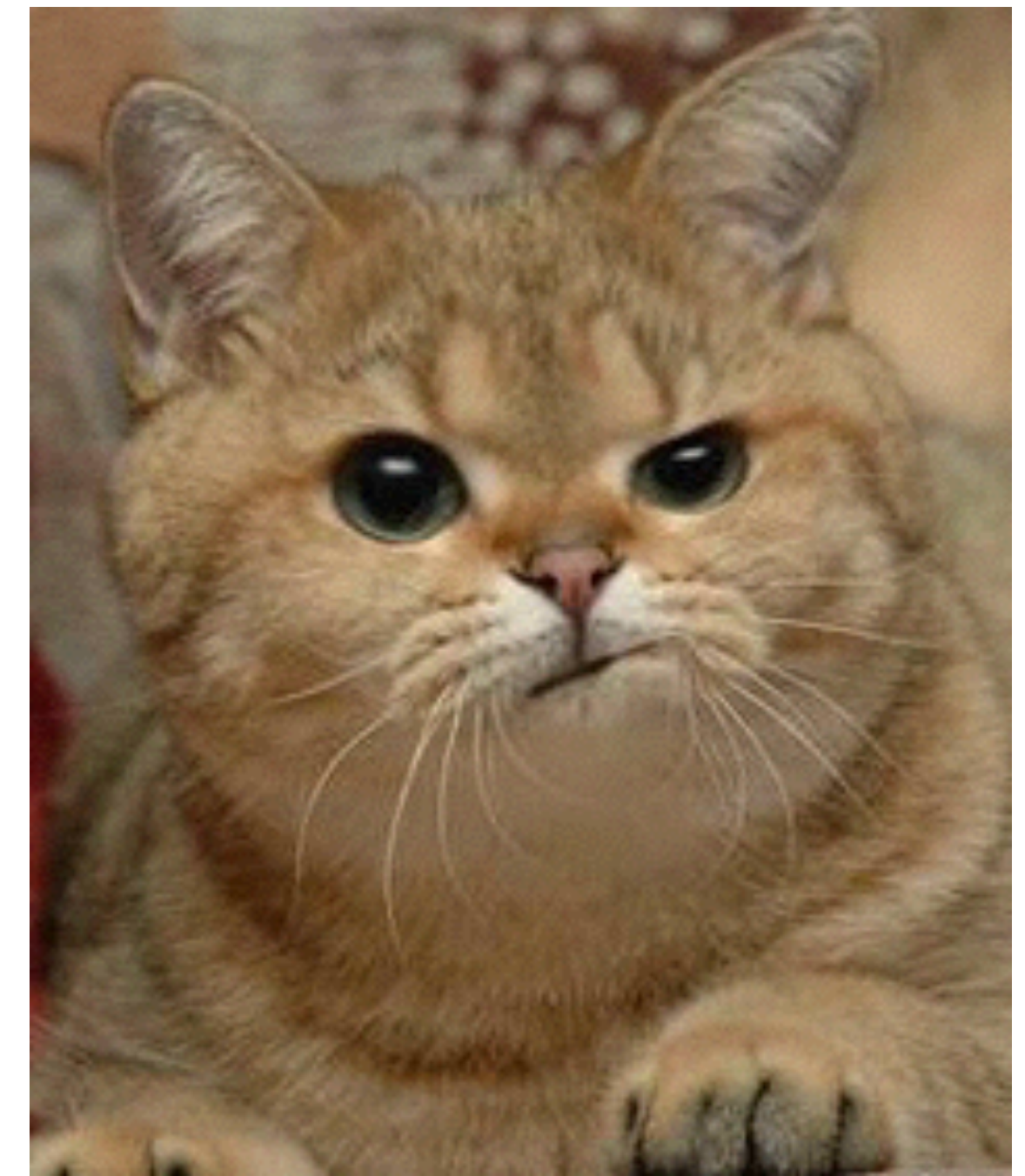
The Power of Convex Algebras (under submission)

Filippo Bonchi, Alexandra Silva, Ana Sokolova

Coinduction up-to in a fibrational setting (CSL-LICS 2014)

Filippo Bonchi, Daniela Petrisan, Damien Pous, Jurriaan Rot

What about succinct
acceptors?



Goal

$$\begin{array}{ccccc}
 X & \xrightarrow{\{\cdot\}} & \mathcal{P}(X) & \xrightarrow{\quad l \quad} & 2^{A^*} \\
 \downarrow \delta & & \searrow \delta^\# & & \downarrow \omega \\
 2 \times \mathcal{P}(X)^A & \xrightarrow{\quad id \times l^A \quad} & & & 2 \times (2^{A^*})^A
 \end{array}$$

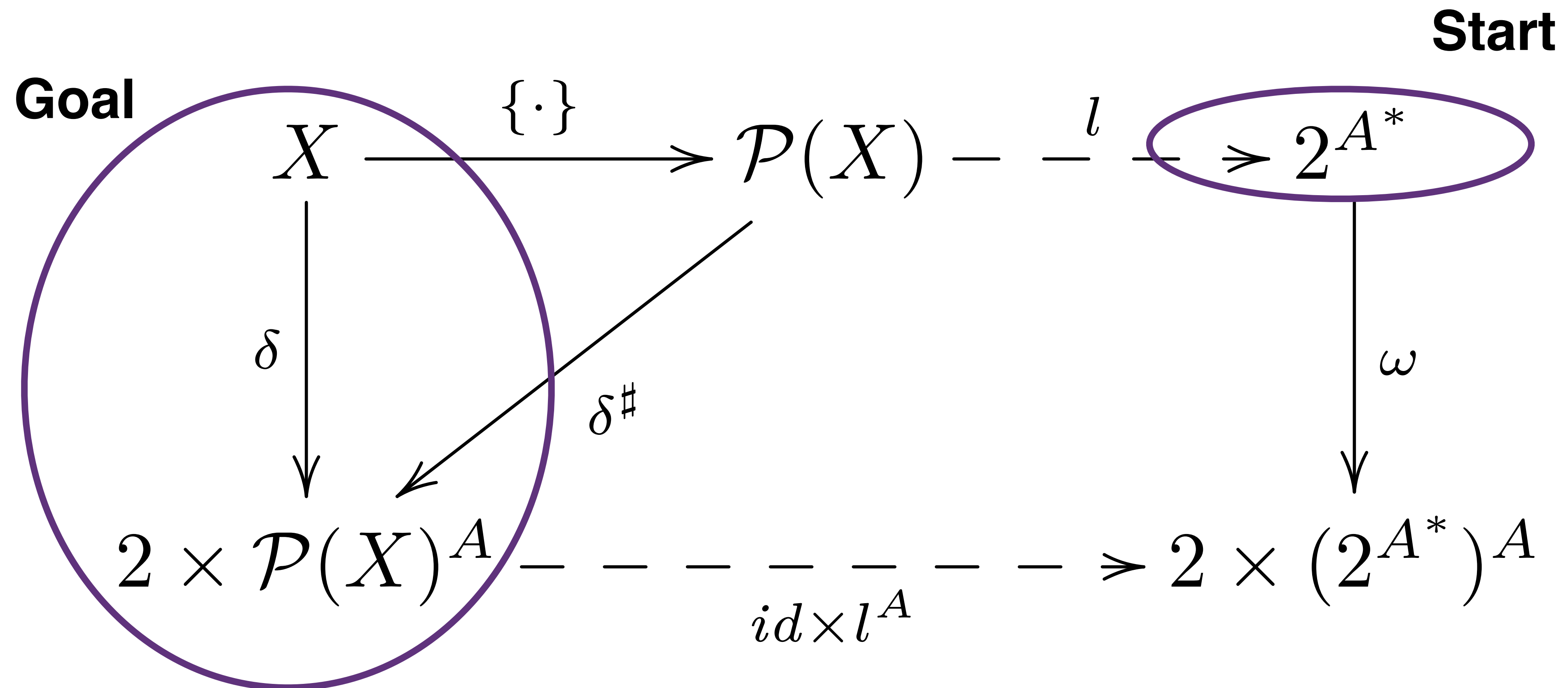
Goal

Start

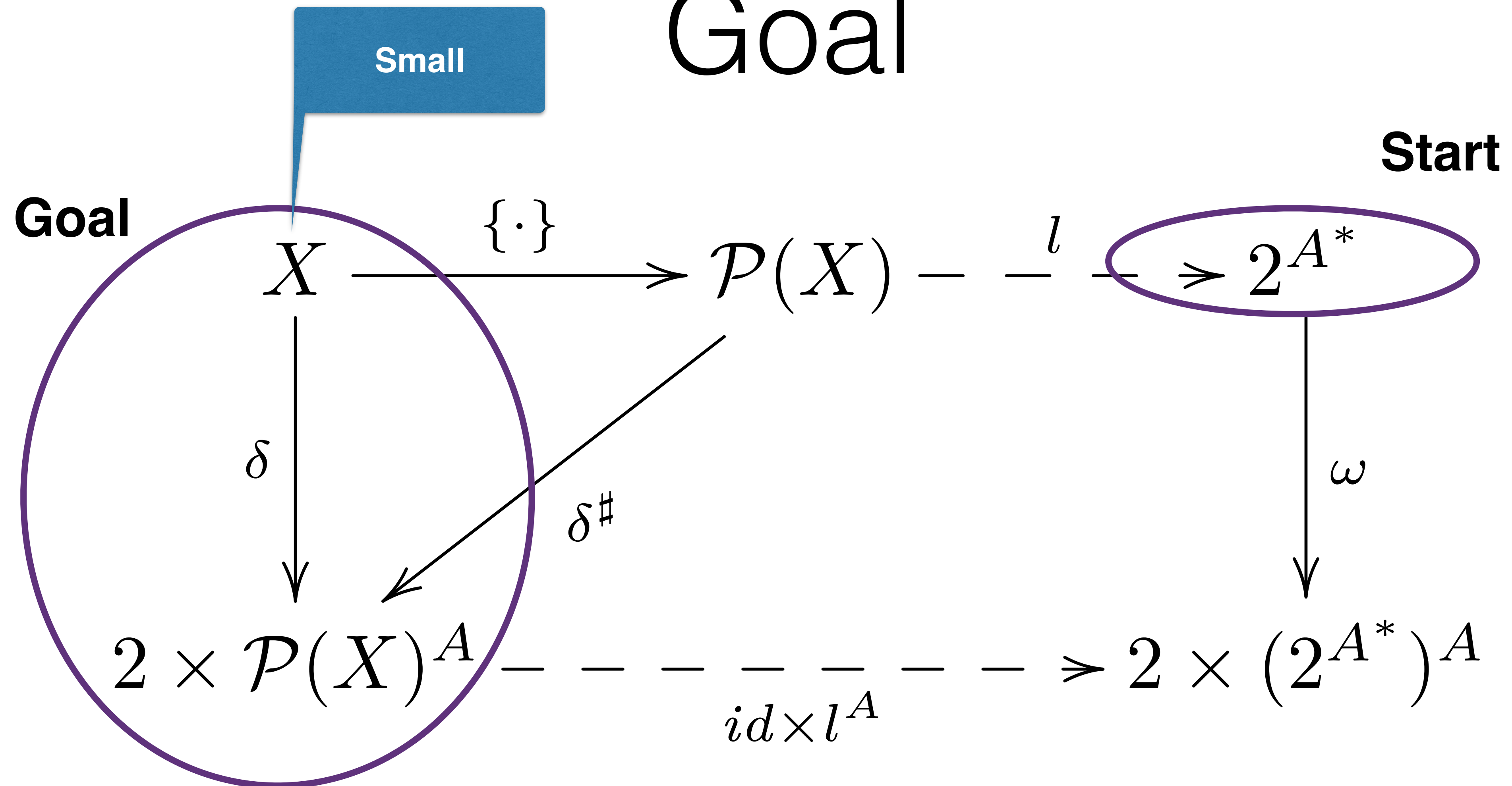
$$\begin{array}{ccccc}
 X & \xrightarrow{\{\cdot\}} & \mathcal{P}(X) & \xrightarrow{\quad l \quad} & 2^{A^*} \\
 \downarrow \delta & & \searrow \delta^\# & & \downarrow \omega \\
 2 \times \mathcal{P}(X)^A & \xrightarrow{\quad id \times l^A \quad} & & & 2 \times (2^{A^*})^A
 \end{array}$$

The diagram illustrates a commutative square. The top row shows a sequence of maps: $X \xrightarrow{\{\cdot\}} \mathcal{P}(X) \xrightarrow{l} 2^{A^*}$. The bottom row shows a sequence of maps: $2 \times \mathcal{P}(X)^A \xrightarrow{id \times l^A} 2 \times (2^{A^*})^A$. A vertical arrow δ points from X to $2 \times \mathcal{P}(X)^A$. A diagonal arrow $\delta^\#$ points from $\mathcal{P}(X)$ to $2 \times \mathcal{P}(X)^A$. A vertical arrow ω points from 2^{A^*} to $2 \times (2^{A^*})^A$. The term 2^{A^*} is circled in purple.

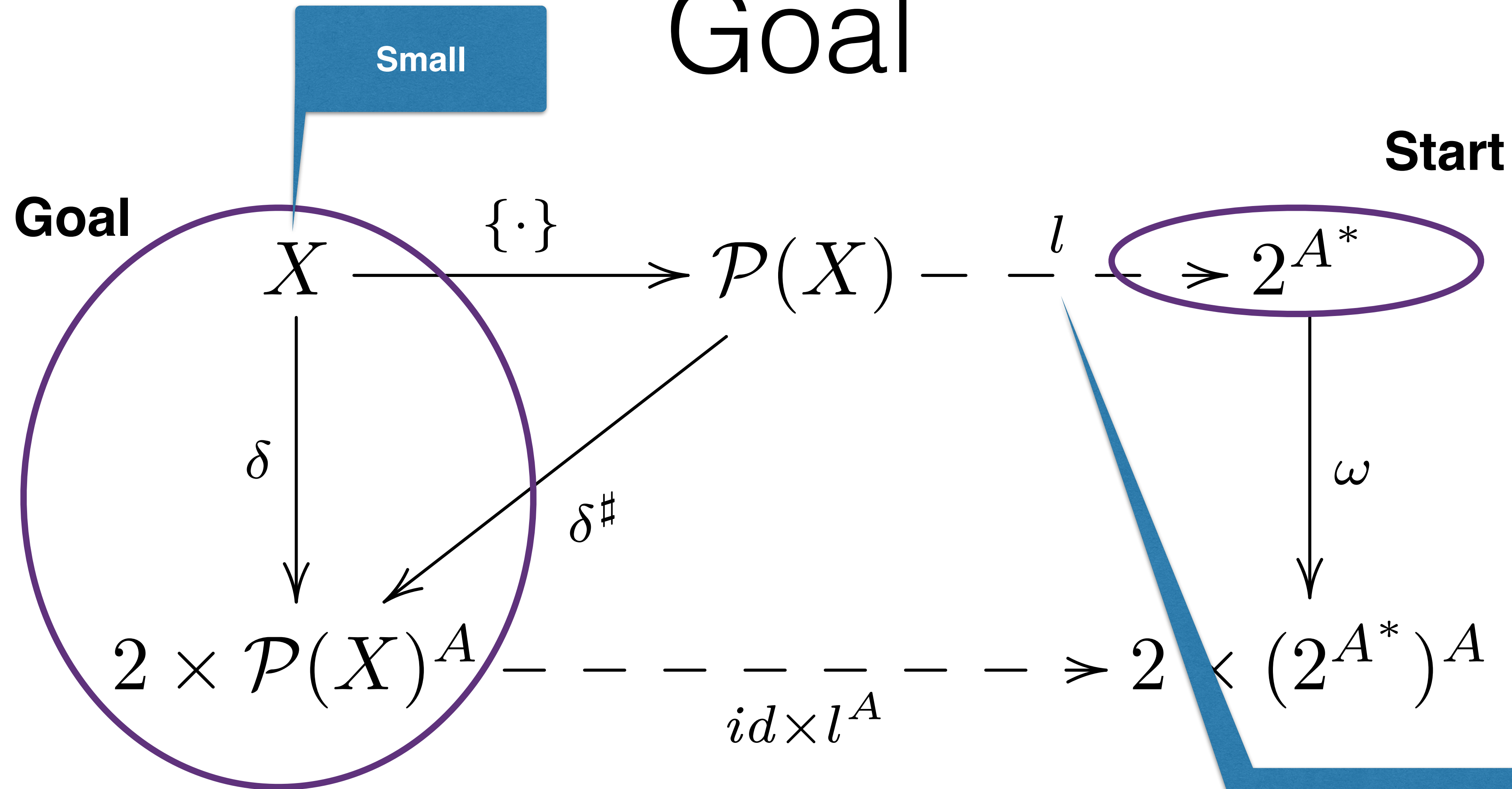
Goal



Goal



Goal



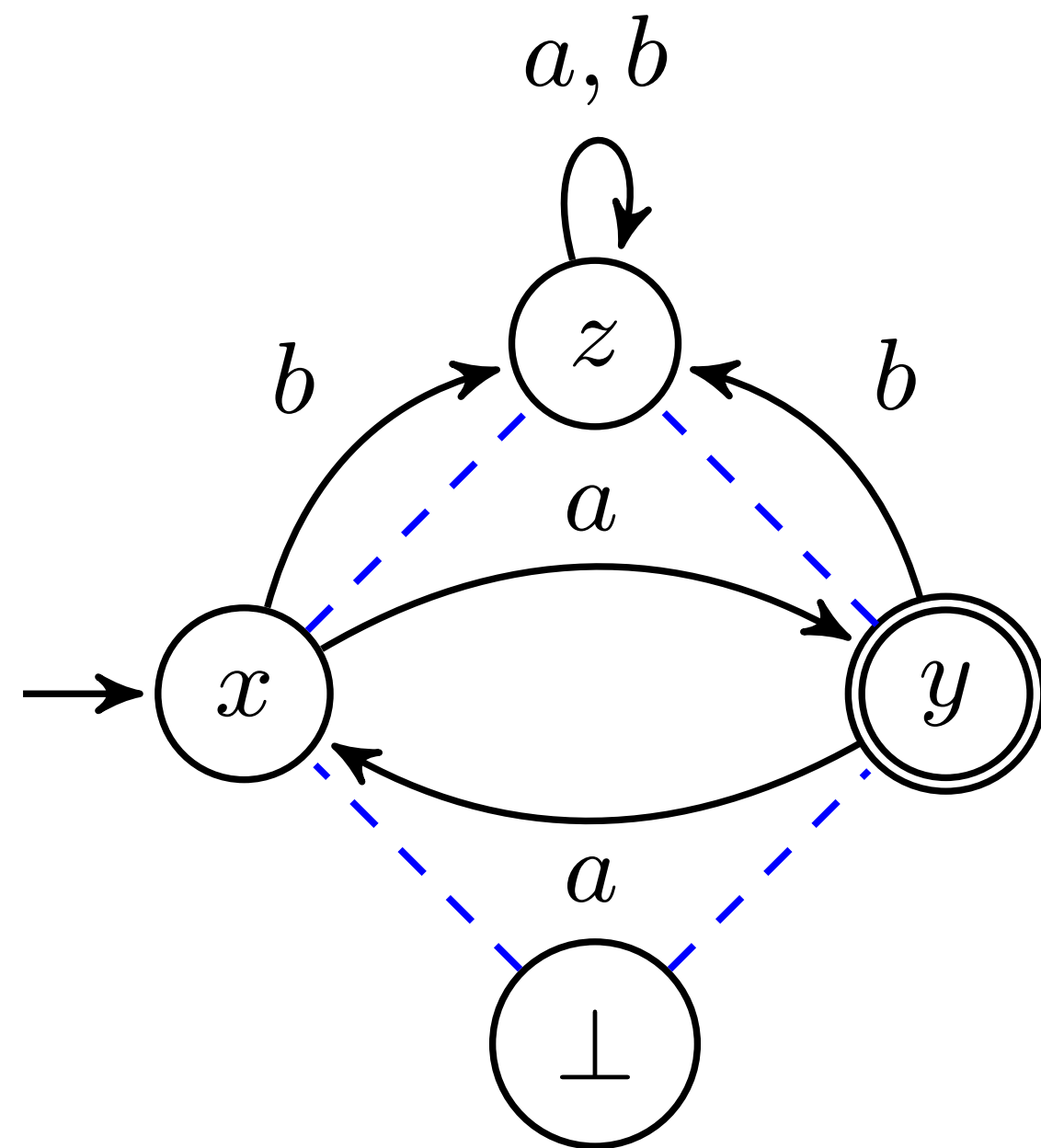
Exploit the algebraic structure!!

Let's try....

$$\mathcal{L} = \{w \in \{a, b\}^* \mid |w|_a \text{ is odd}\}$$

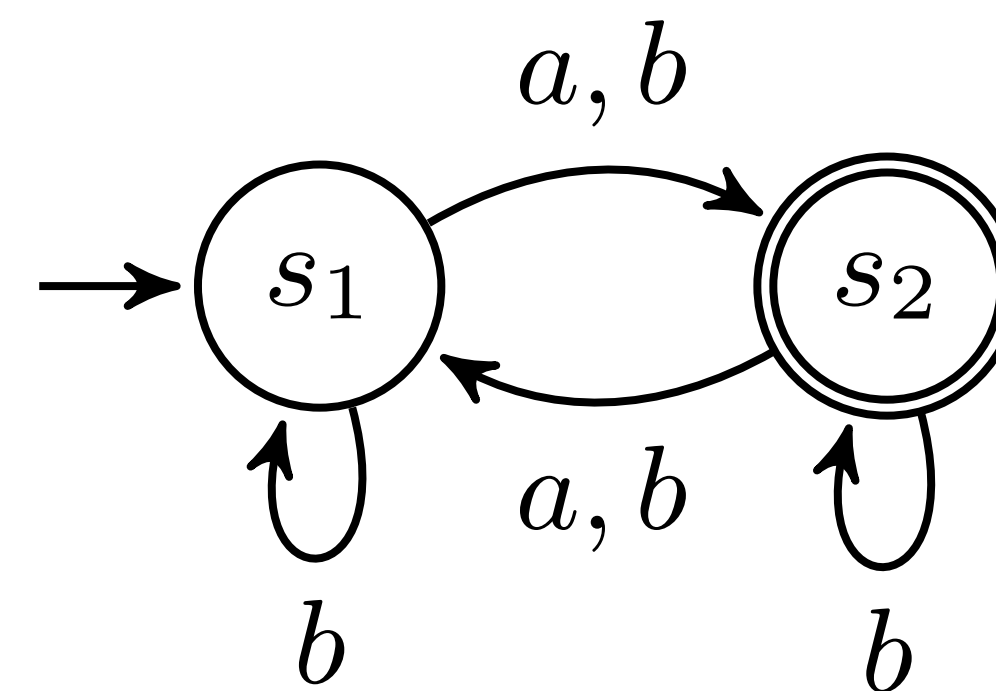
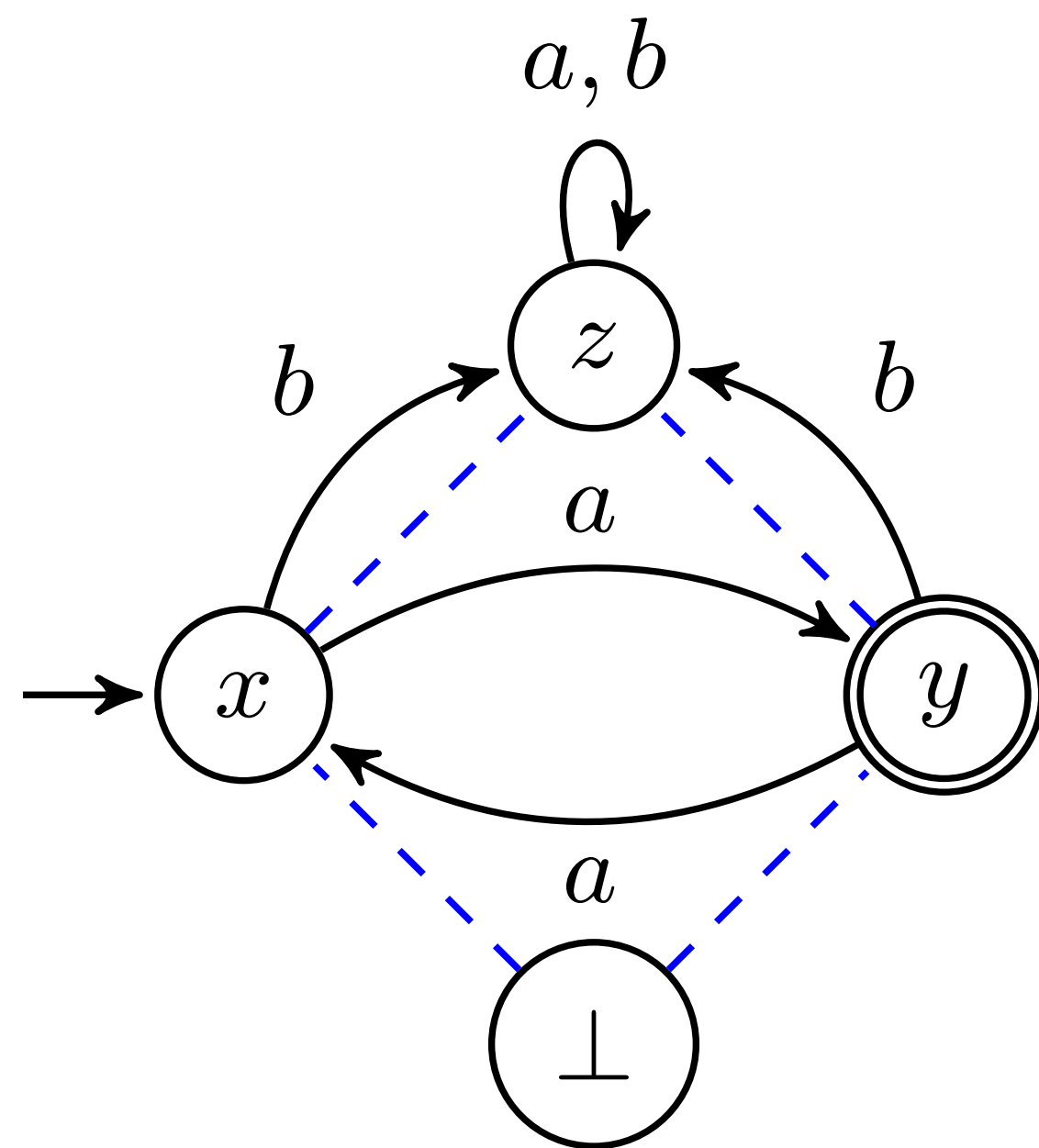
Let's try....

$$\mathcal{L} = \{w \in \{a, b\}^* \mid |w|_a \text{ is odd}\}$$



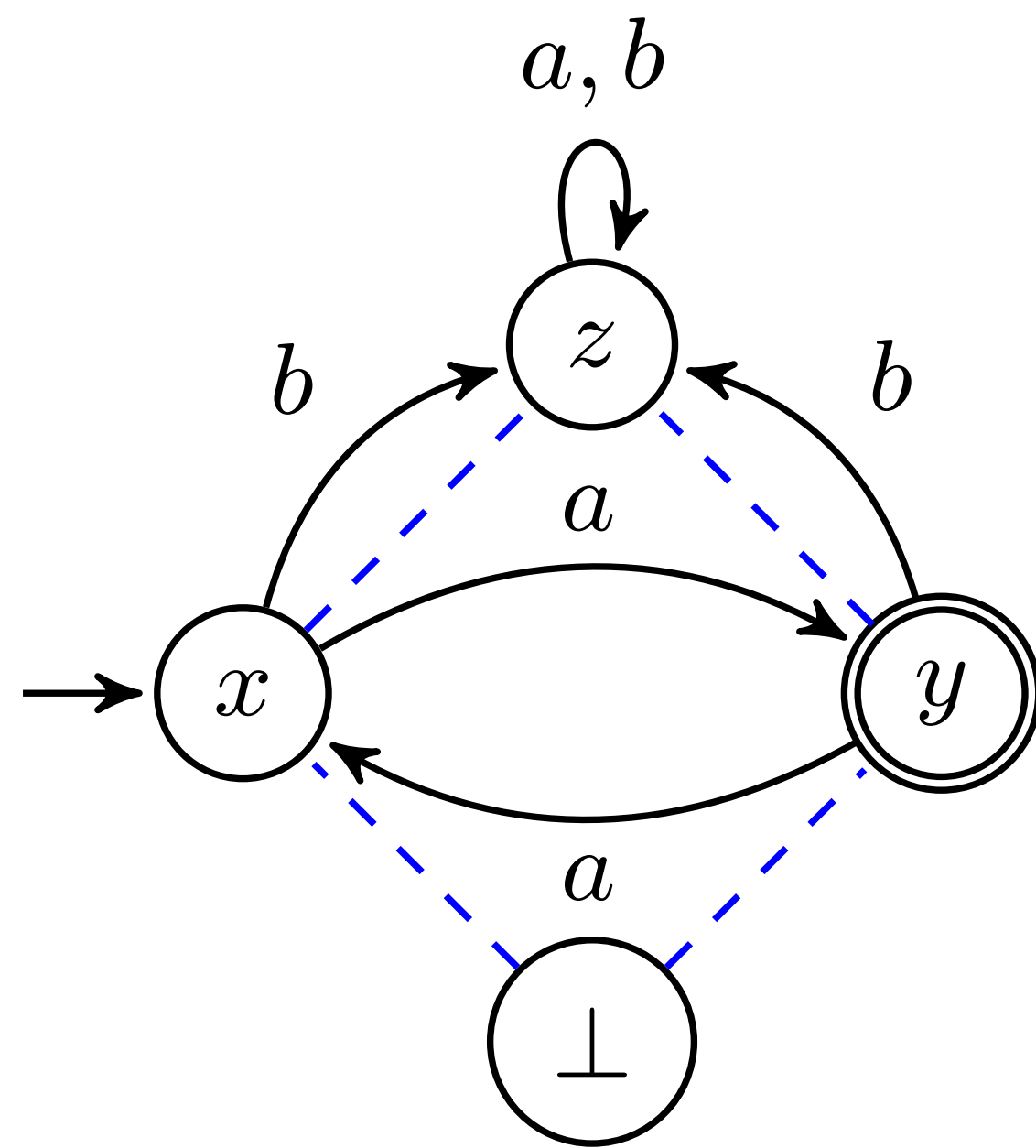
Let's try....

$$\mathcal{L} = \{w \in \{a, b\}^* \mid |w|_a \text{ is odd}\}$$

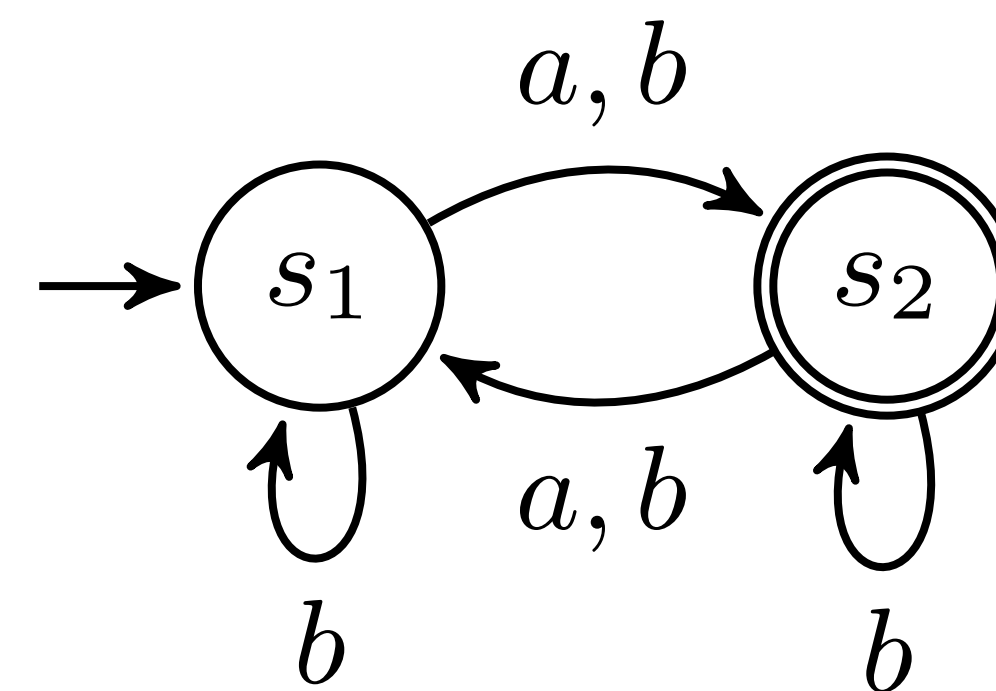


Let's try....

$$\mathcal{L} = \{w \in \{a, b\}^* \mid |w|_a \text{ is odd}\}$$



Join-irreducibles



What else can we do?

Languages are much richer algebraically — CABA & DL

$$\mathcal{L} = \Sigma^* aa \Sigma^* \cap \Sigma^* bb \Sigma^*$$

What else can we do?

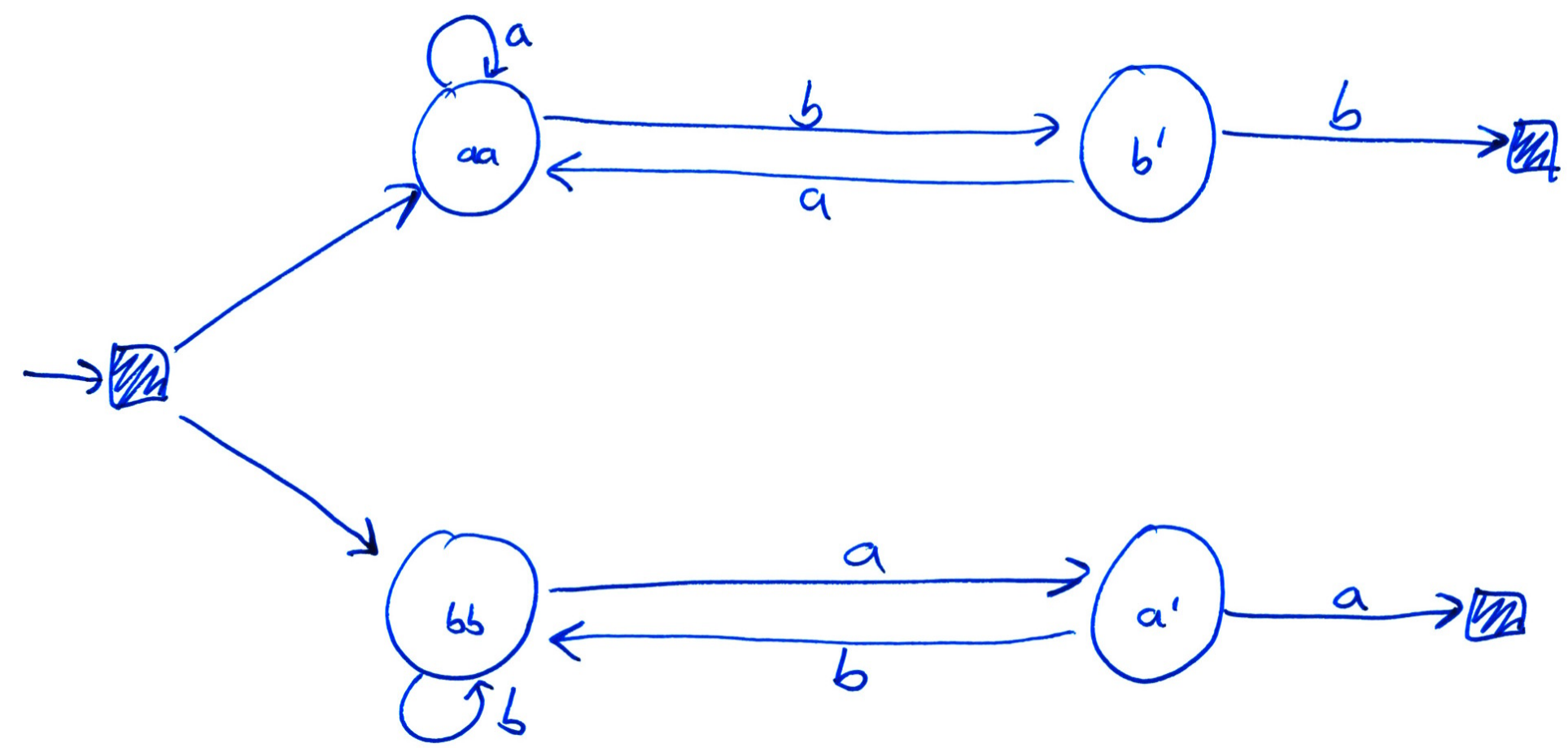
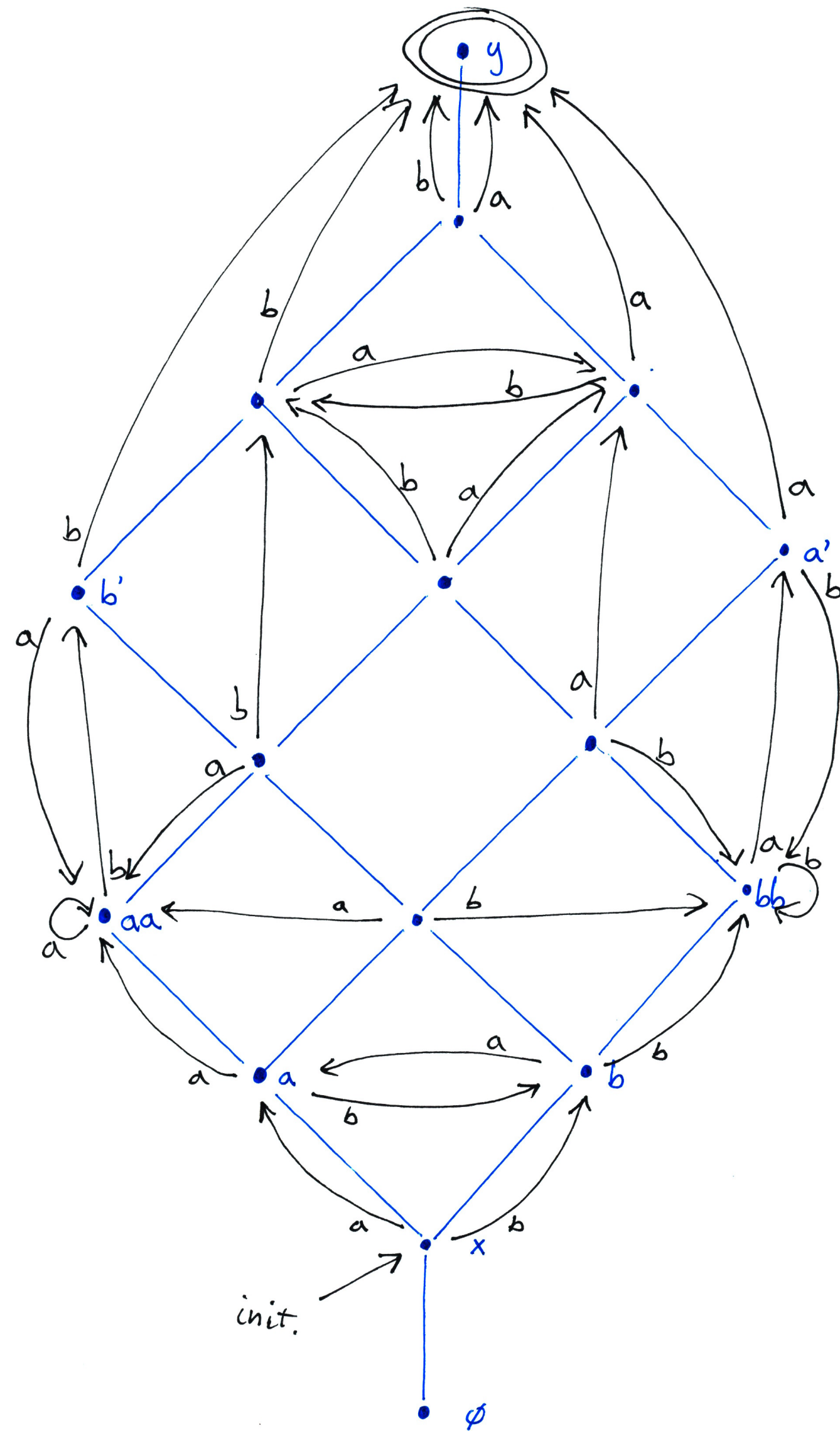
Languages are much richer algebraically — CABA & DL

Brzowski's atomaton

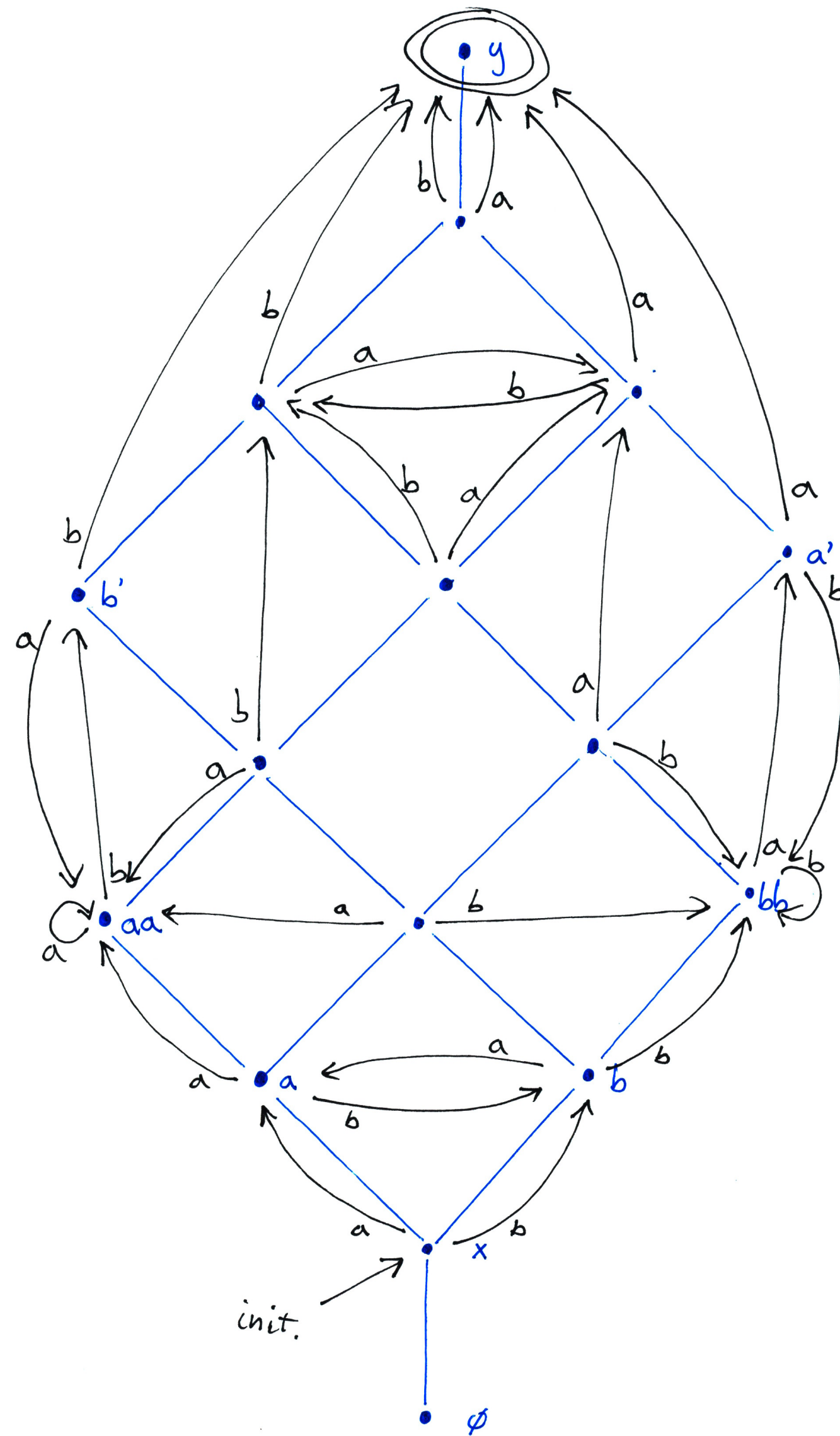
Alternating automaton

$$\mathcal{L} = \Sigma^* aa \Sigma^* \cap \Sigma^* bb \Sigma^*$$

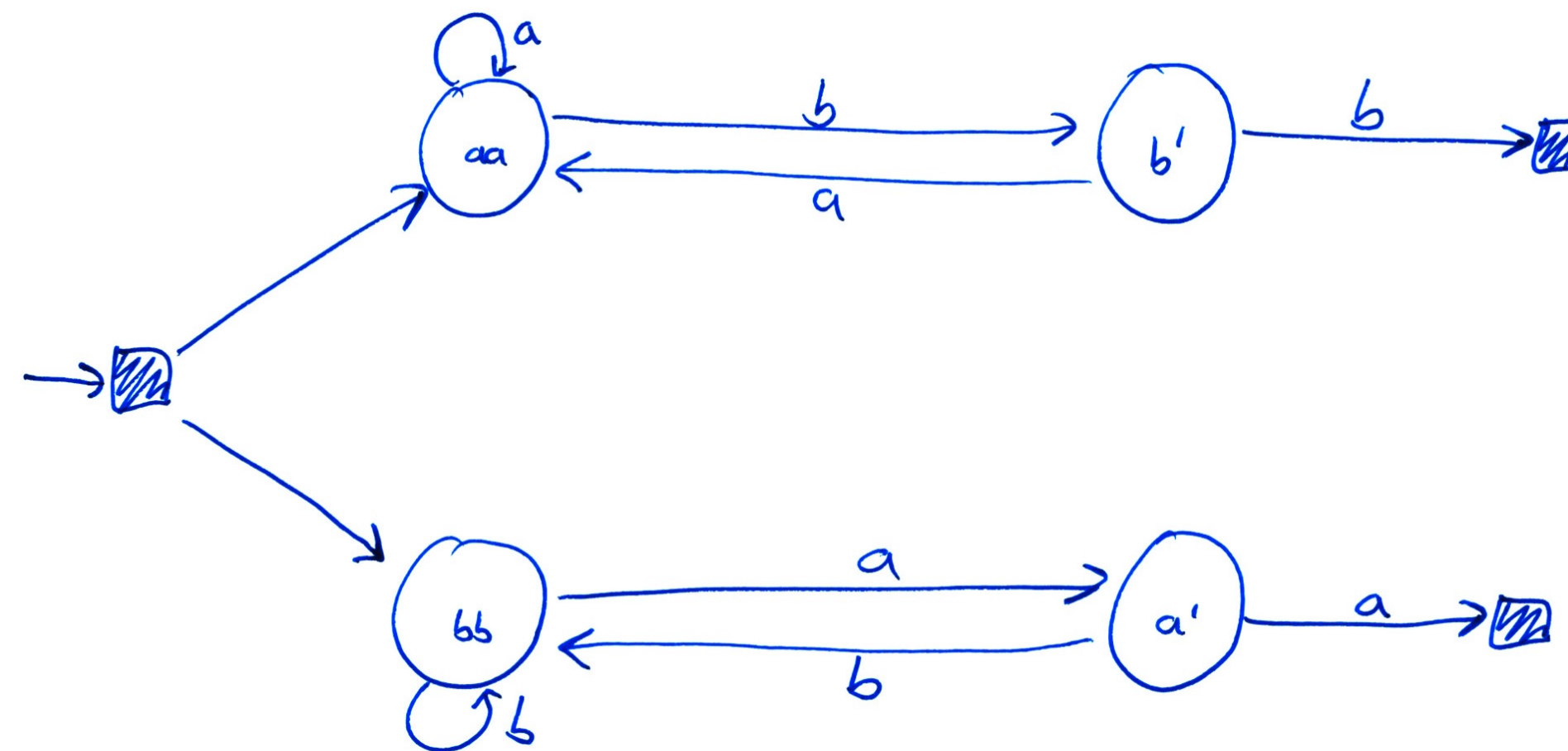
$$\mathcal{L} = \Sigma^* aa \Sigma^* \cap \Sigma^* bb \Sigma^*$$



$$\mathcal{L} = \Sigma^* aa \Sigma^* \cap \Sigma^* bb \Sigma^*$$



Smallest DFA - 8 states



Weighted languages

$$\mathcal{L} = \begin{cases} \varepsilon \mapsto 1 \\ (aa)^n \mapsto 2^n \end{cases}$$

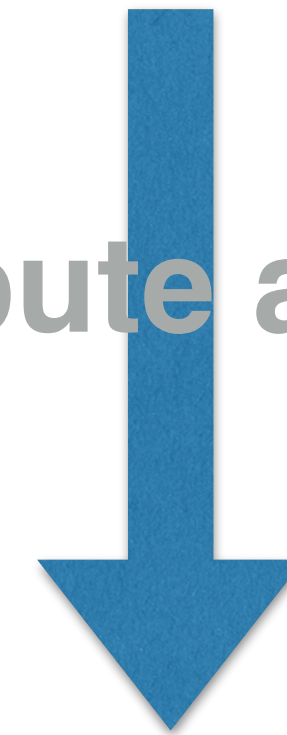
Infinite Moore automaton
in **Vect**

Weighted languages

$$\mathcal{L} = \begin{cases} \varepsilon \mapsto 1 \\ (aa)^n \mapsto 2^n \end{cases}$$

Infinite Moore automaton
in **Vect**

Compute a basis



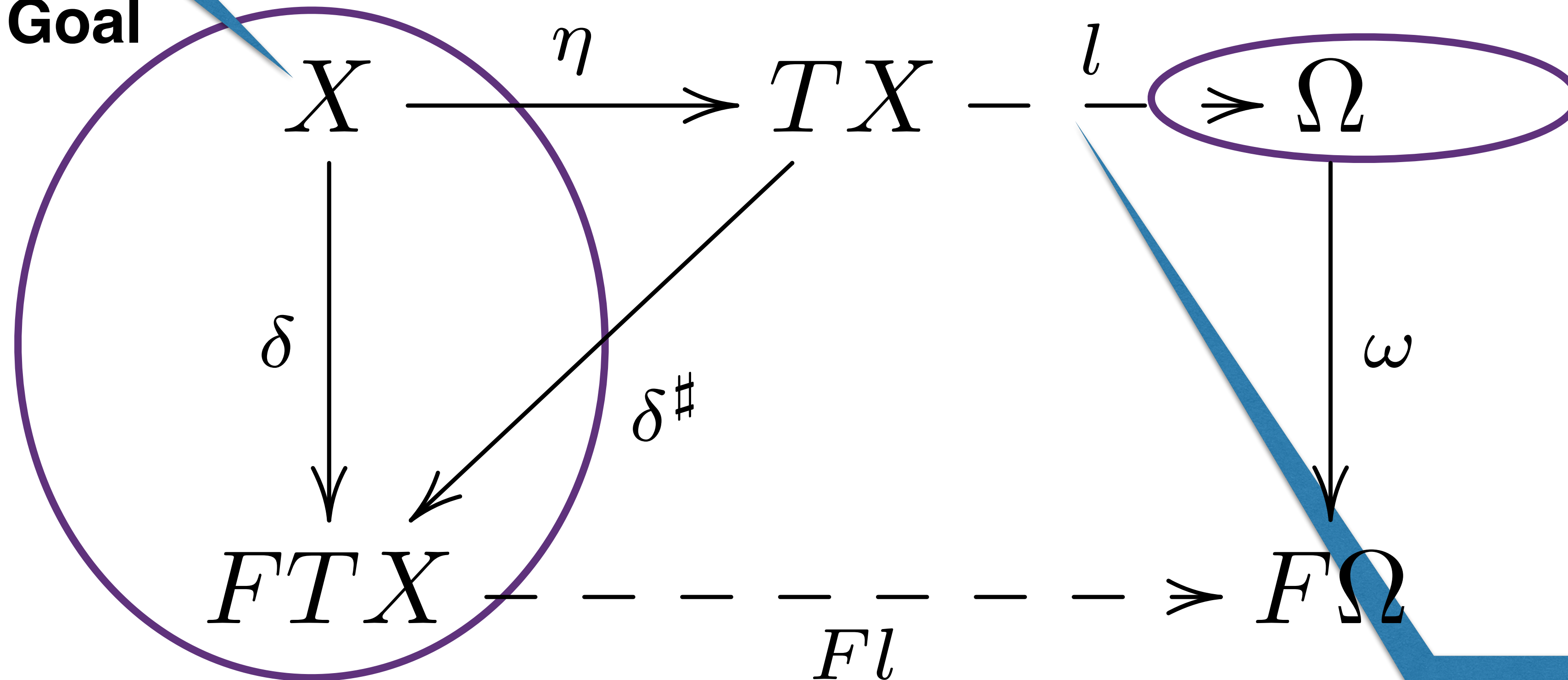
Finite Weighted automaton

General Picture

Small

Goal

Start



Exploit the T-algebraic structure!!

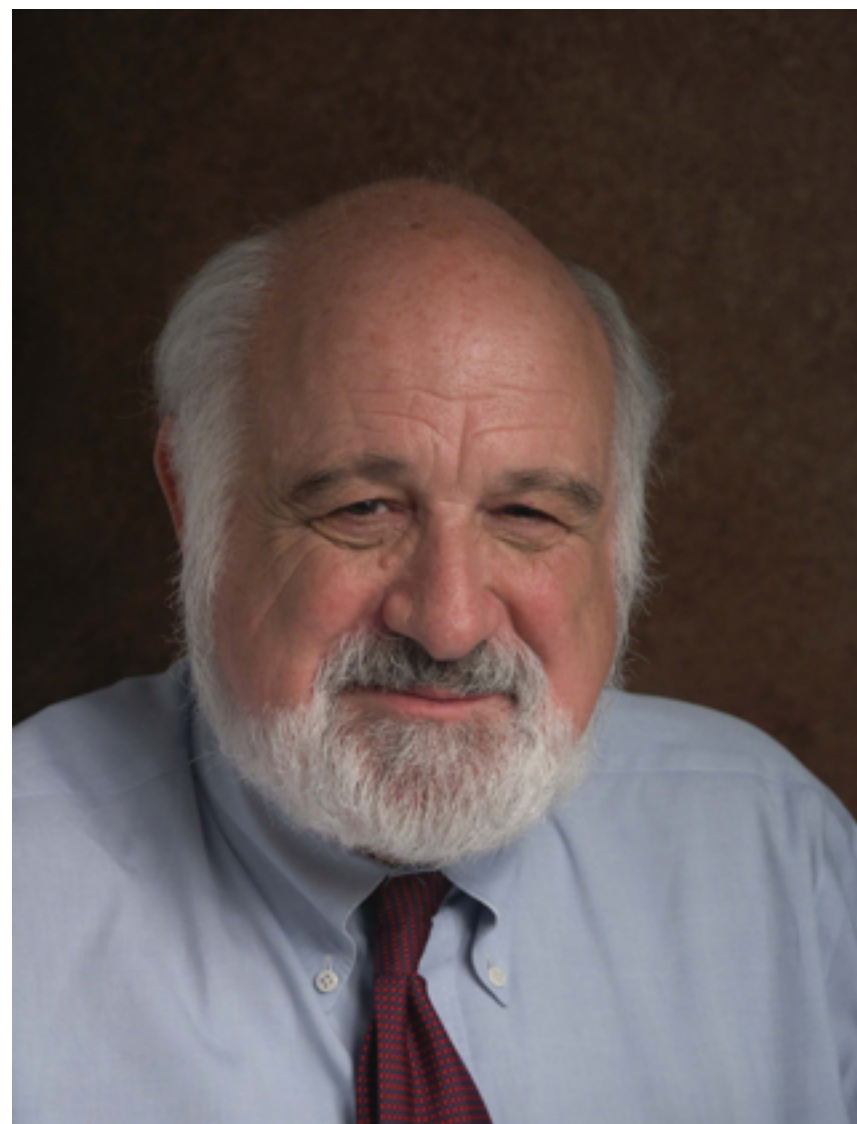
What have

we gained

A general picture to understand
reverse determinization

What have we gained

A general picture to understand
reverse determinization

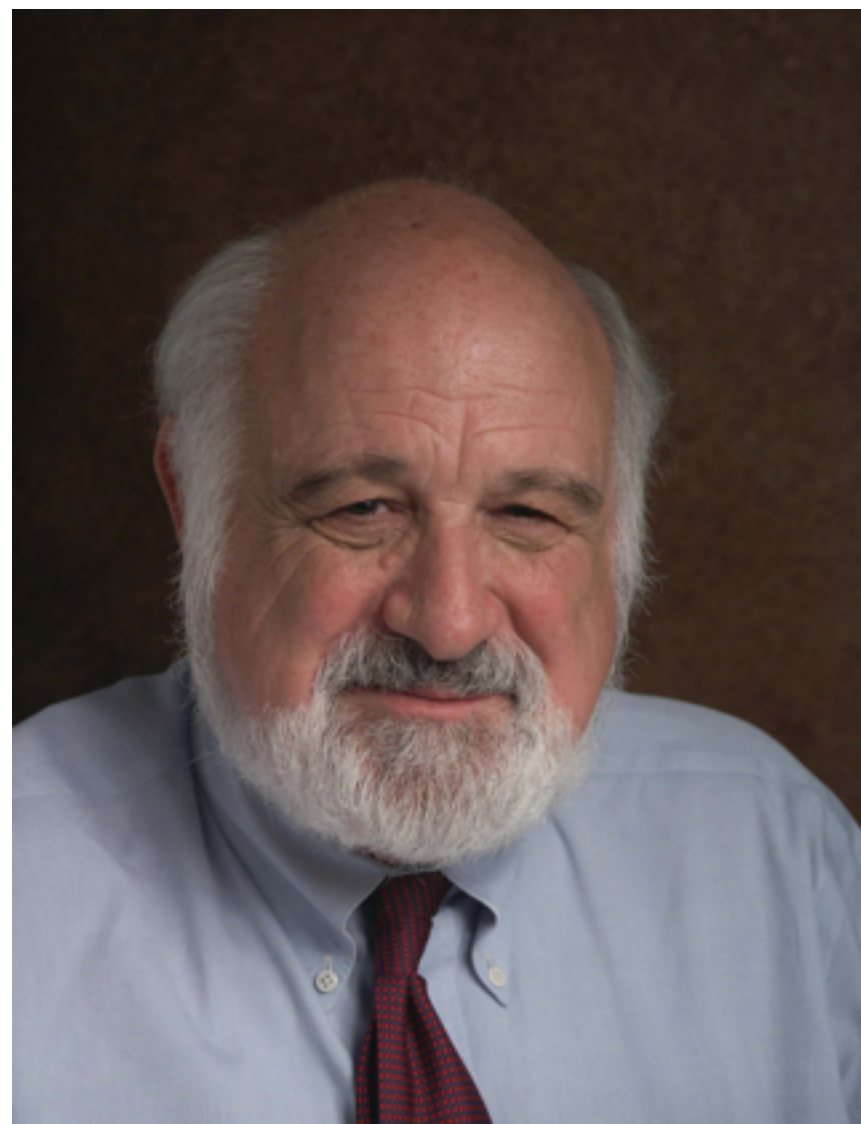


Fuzzy machines

M. Arbib and E. Manes

What have **not** we gained

A general picture to understand
reverse determinization



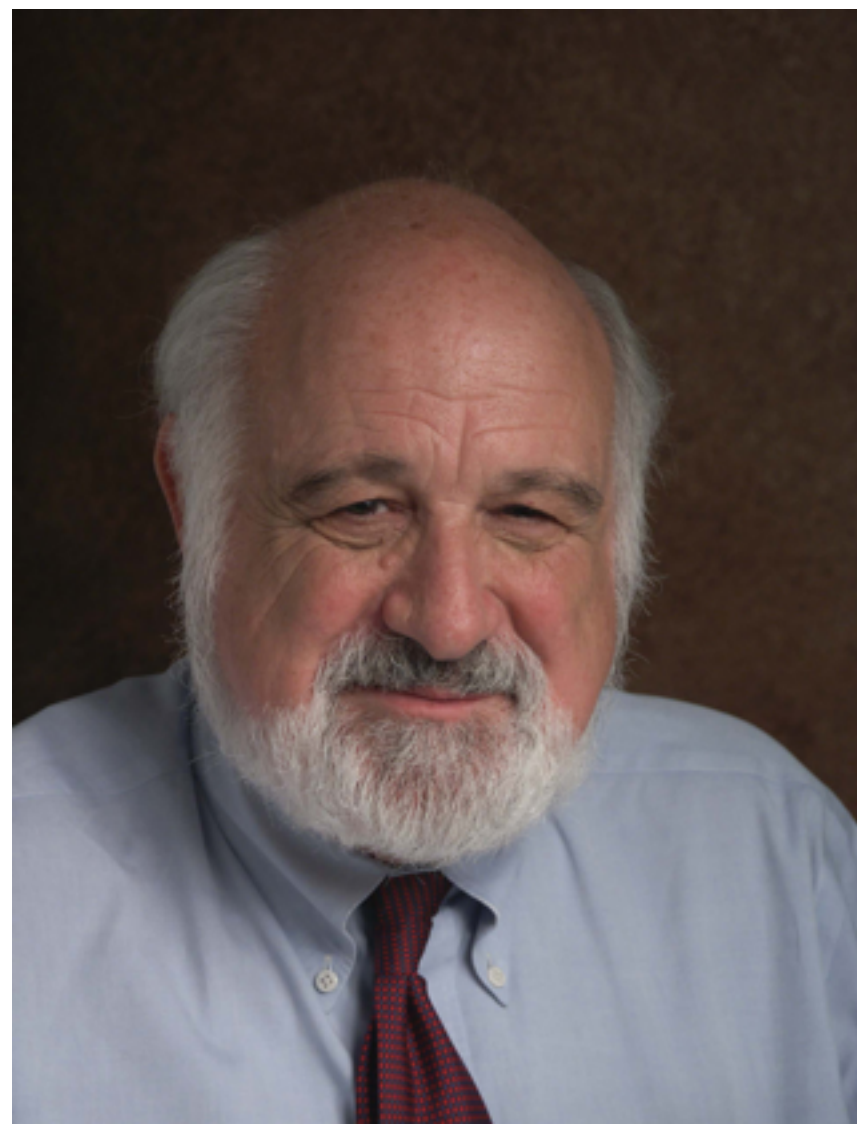
Fuzzy machines

M. Arbib and E. Manes

What have **not** we gained

A general picture to understand
reverse determinization

An algorithm to compute
the *generators*



Fuzzy machines

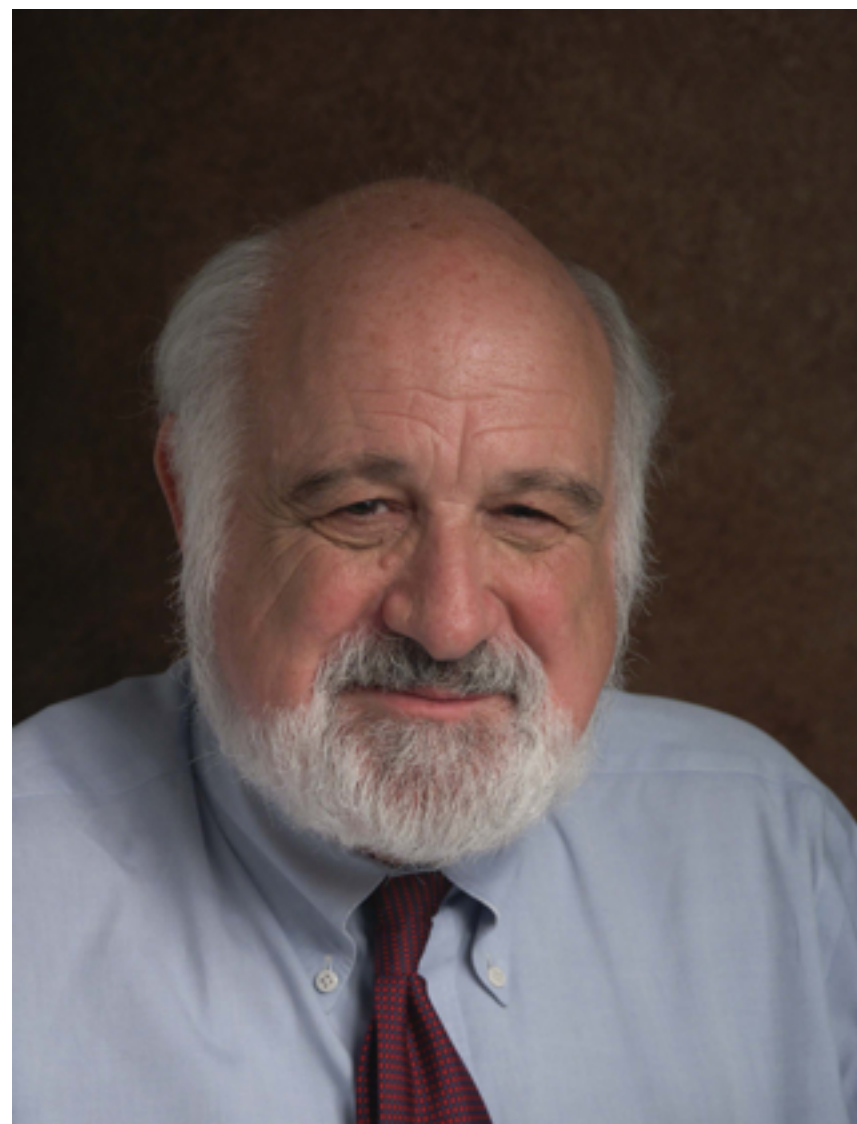
M. Arbib and E. Manes

What have **not** we gained

A general picture to understand
reverse determinization

Not immediate — ad-hoc
general algorithm;
efficient versions for
specific instances

An algorithm to compute
the *generators*



Fuzzy machines

M. Arbib and E. Manes

Conclusions

Algebraic structure of languages is very important

For semantics

For algorithms

Conclusions

Algebraic structure of languages is very important

For semantics

For algorithms

Coalgebraic structure of languages is very important

Questions?

