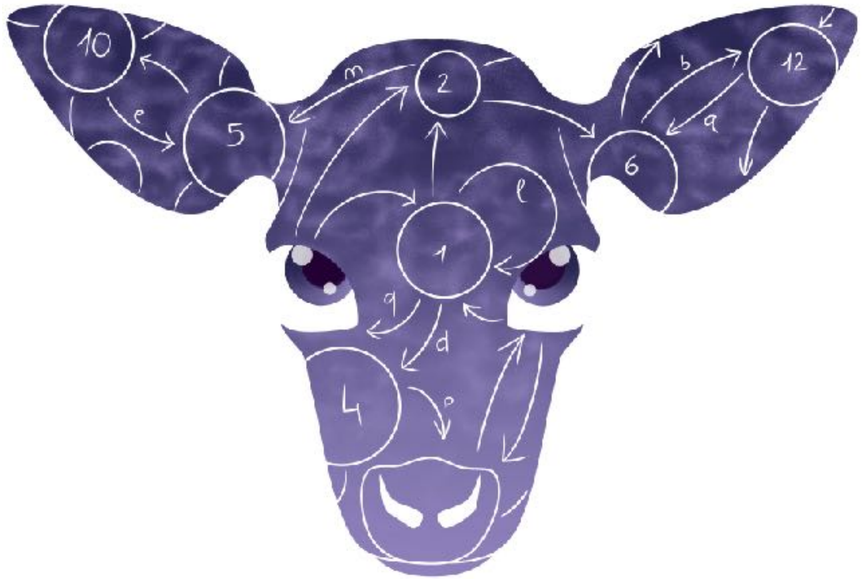


Categorical Automata Learning Framework

calf-project.org

Matteo Sammartino

S-REPLS 6

UCL ENGINEERING
Change the world

UCL

Our team

Alexandra Silva
UCL

Matteo Sammartino
UCL

Gerco van Heerdt
UCL

Joshua Moerman
Radboud University

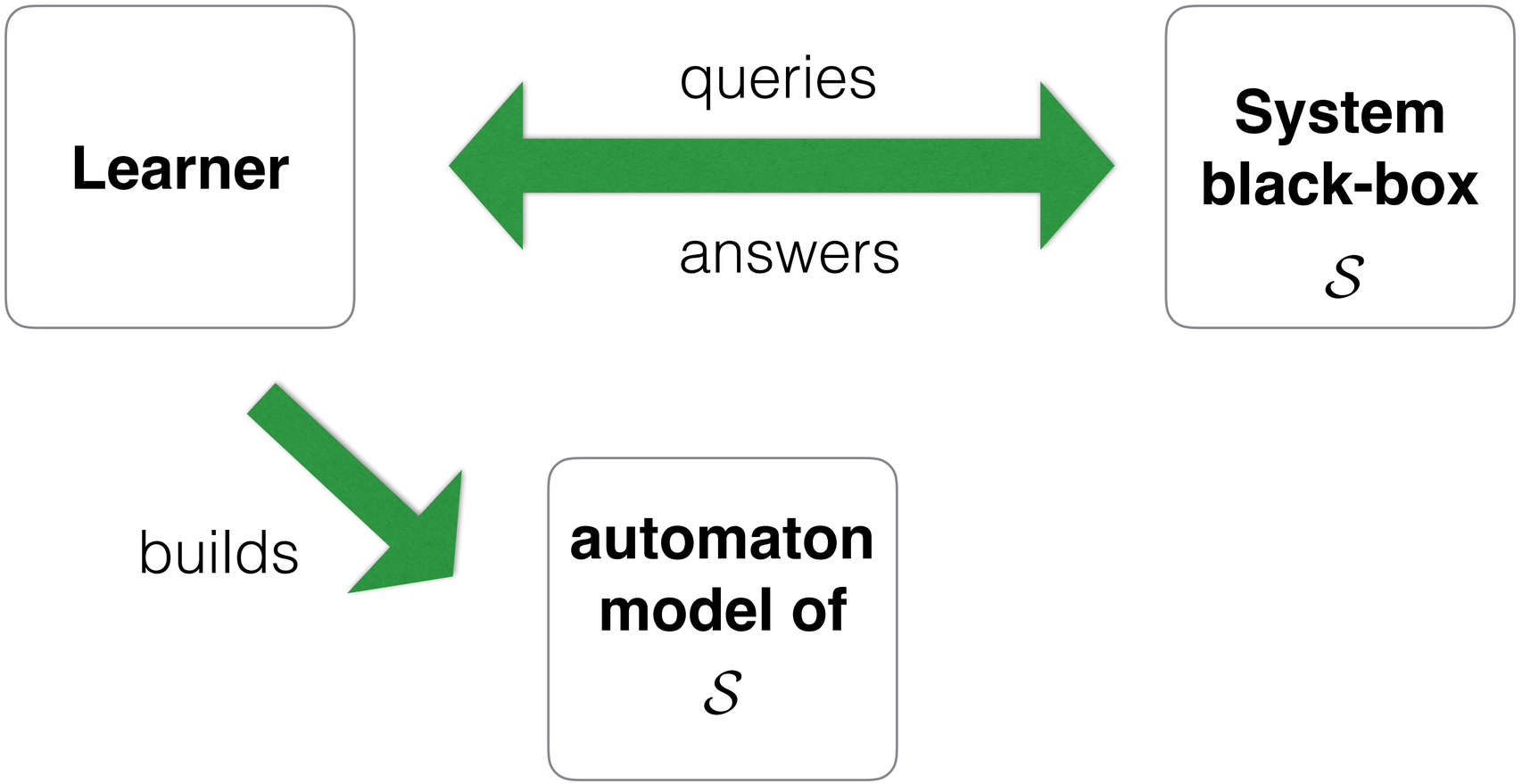
Maverick Chardet
ENS Lyon

Collaborators:

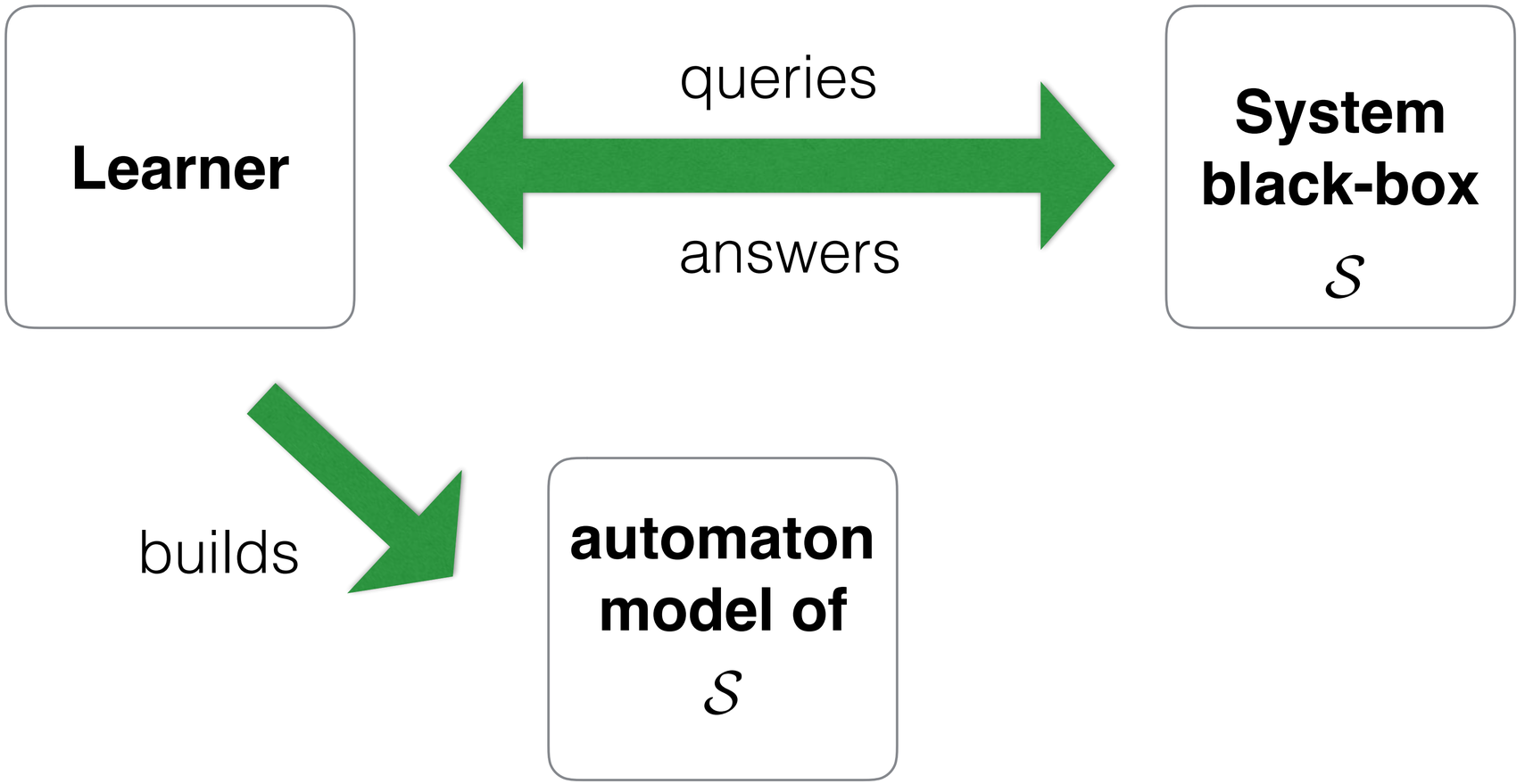
Bartek Klin
Warsaw University

Michal Szynwelski
Warsaw University

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** $\mathcal{L} \subseteq A^\star$

L* algorithm (D.Angluin ’87)

Finite alphabet of system’s actions A

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

Learner

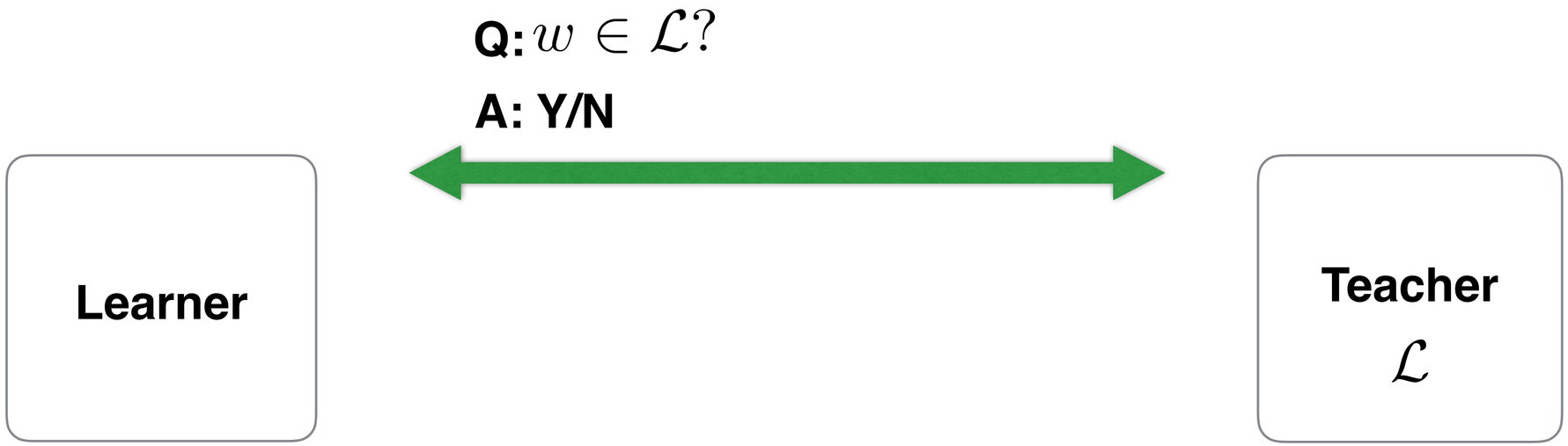
Teacher

$\mathcal{L}$

L* algorithm (D.Angluin ’87)

Finite alphabet of system’s actions A

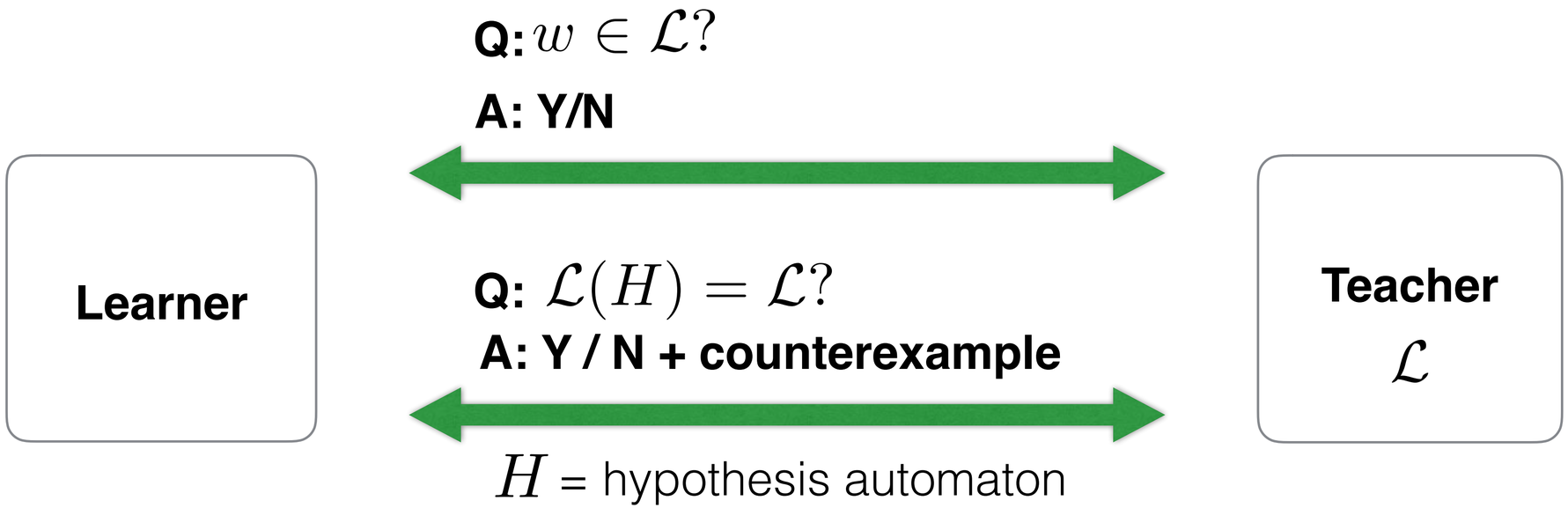
set of system behaviors is a **regular language** $\mathcal{L} \subseteq A^\star$



L* algorithm (D.Angluin ’87)

Finite alphabet of system’s actions A

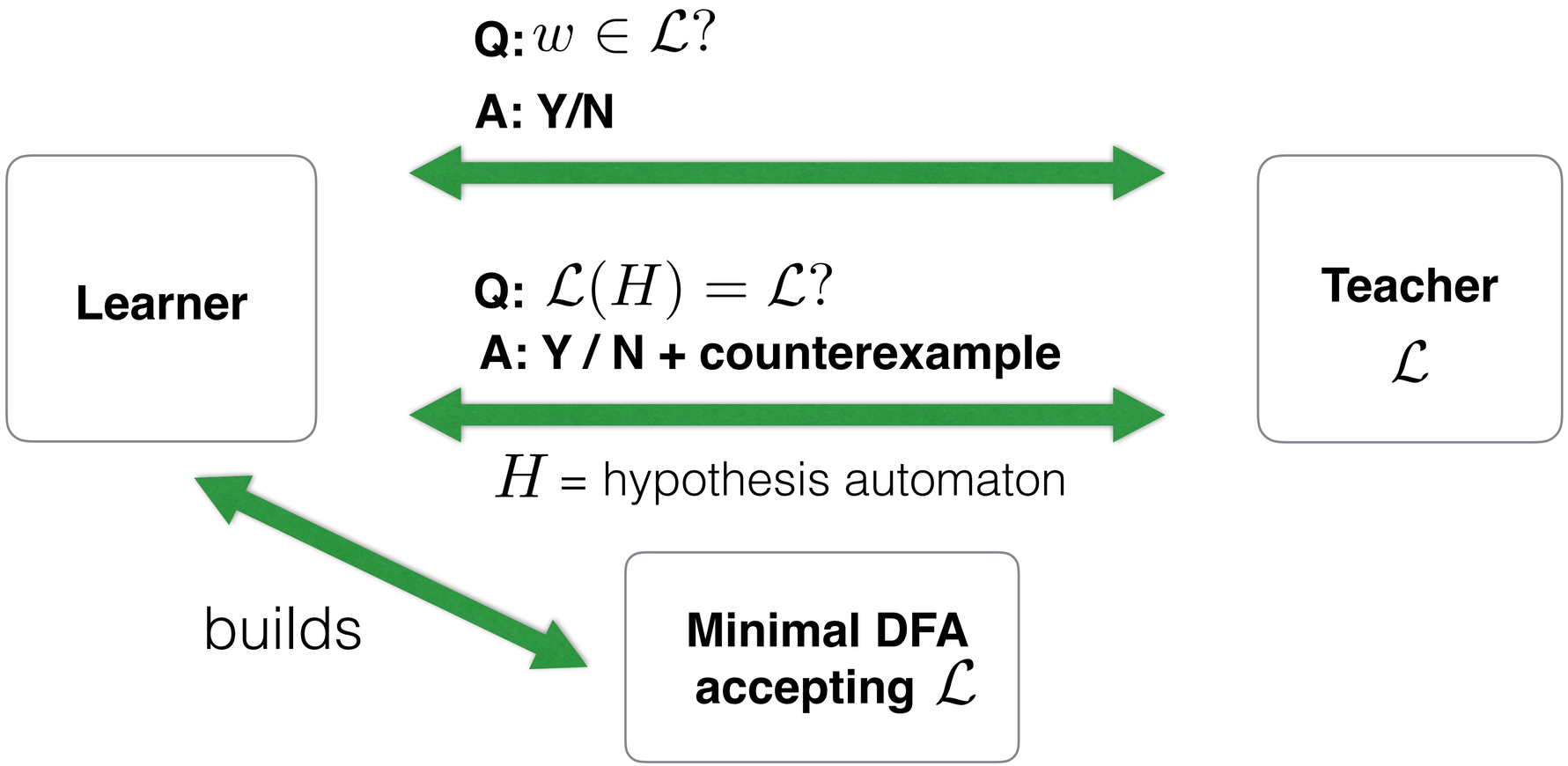
set of system behaviors is a **regular language** $\mathcal{L} \subseteq A^\star$



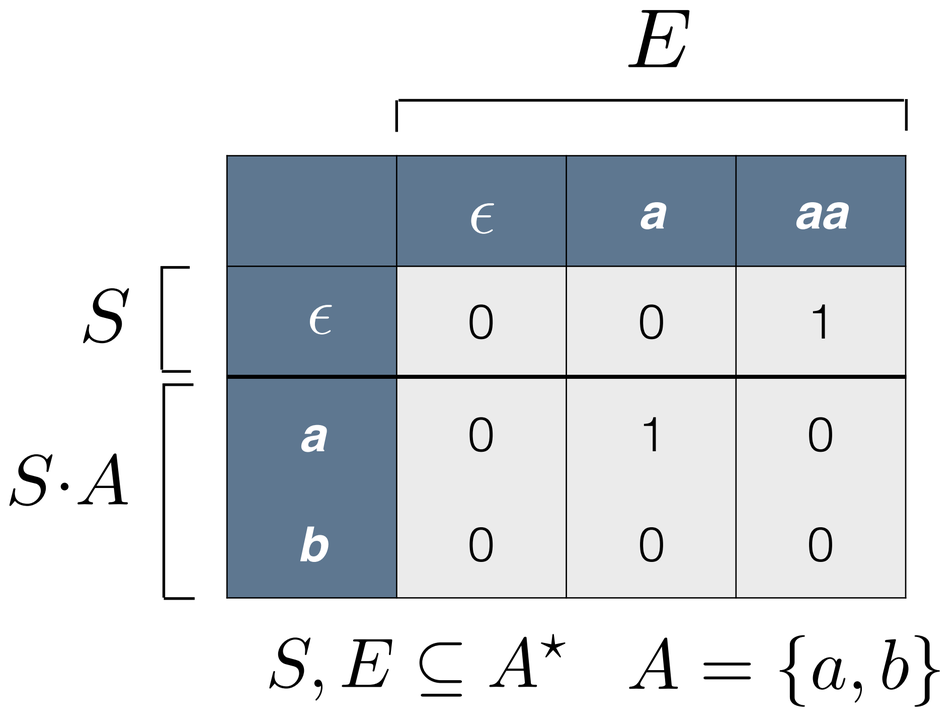
L* algorithm (D.Angluin ’87)

Finite alphabet of system’s actions A

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



Observation table



$S, E \subseteq A^\star \quad A = \{a, b\}$

$row\colon S \cup S{\cdot}A \to 2^E$

$row(s)(e) = 1 \iff se \in \mathcal{L}$

Observation table

E

		ϵ	*a*	*aa*
S	ϵ	0	0	1
$S \cdot A$	*a*	0	1	0
	b	0	0	0

$S, E \subseteq A^\star \quad A = \{a, b\}$

$$row \colon S \cup S \cdot A \to 2^E$$

$$row(s)(e) = 1 \iff se \in \mathcal{L}$$

Hypothesis automaton

$$\begin{cases} \text{states} = & \{row(s) \mid s \in S\} \\ \text{final states} = & \{row(s) \mid s \in S, row(s)(\epsilon) = 1\} \\ \text{initial state} = & row(\epsilon) \\ \text{transition function} & row(s) \xrightarrow{a} row(sa) \end{cases}$$

Observation table

E

		ϵ	*a*	*aa*
S	ϵ	0	0	1
$S \cdot A$	*a*	0	1	0
	b	0	0	0

$S, E \subseteq A^\star \quad A = \{a, b\}$

$$row \colon S \cup S \cdot A \to 2^E$$

$$row(s)(e) = 1 \iff se \in \mathcal{L}$$

Hypothesis automaton

$$\left\{ \begin{array}{rl} \text{states} = & \{row(s) \mid s \in S\} \\ \text{final states} & \{row(s) \mid s \in S, row(s)(\epsilon) = 1\} \\ \text{initial state} & \\ \text{transition function} & row(s) \xrightarrow{a} row(sa) \end{array} \right.$$

Why is this correct?

Table properties

Table properties

Closed

$$\forall t \in S \cdot A \quad \exists s \in S \qquad row(t) = row(s).$$

Table properties

$$row(s) \xrightarrow{a} row(sa)$$

Closed

$$\forall t \in S \cdot A \quad \exists s \in S \qquad row(t) = row(s).$$

Table properties

$$row(s) \xrightarrow{a} row(sa)$$

next state exists

Closed

$$\forall t \in S \cdot A \quad \exists s \in S \qquad row(t) = row(s).$$

Table properties

$$row(s) \xrightarrow{a} row(sa)$$

next state exists

Closed

$$\forall t \in S \cdot A \quad \exists s \in S \qquad row(t) = row(s).$$

Consistent

$$\forall s_1, s_2 \in S \quad row(s_1) = row(s_2) \implies \forall a \in A \;\; row(s_1 a) = row(s_2 a)$$

Table properties

$$row(s) \xrightarrow{a} row(sa)$$

next state exists

determinism

Closed

$$\forall t \in S \cdot A \quad \exists s \in S \qquad row(t) = row(s).$$

Consistent

$$\forall s_1, s_2 \in S \quad row(s_1) = row(s_2) \implies \forall a \in A \;\; row(s_1 a) = row(s_2 a)$$

Table properties

$$row(s) \xrightarrow{a} row(sa)$$

next state exists

determinism

Closed

$$\forall t \in S \cdot A \quad \exists s \in S \quad row(t) = row(s).$$

Fixed by extending the table

$$\forall s_1, s_2 \in S \quad row(s_1) = row(s_2) \implies \forall a \in A \;\; row(s_1 a) = row(s_2 a)$$

Pros of L* …

simple is beautiful

&

POWERFUL

Applications : Hardware verification, security/network protocols…

Generalizations : Mealy machines, I/O automata, …

A zoo of automata

Probabilistic

Weighted

Alternating

Register

Universal

Non-deterministic

Mealy Machines

A zoo of automata

Probabilistic

Weighted

Alternating

Register

Universal

Non-deterministic

Mealy Machines

Algorithms

Correctness proofs

involved and hard to check

A zoo of automata

Probabilistic

Weighted

Alternating

Register

Universal

Non-deterministic

Category theory comes to the rescue!

Algorithms

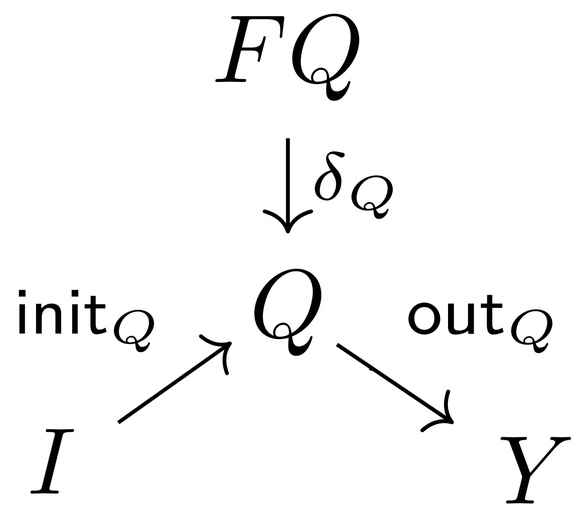
Correctness proofs

involved and hard to check

Abstract automata

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

Endofunctor $F\colon \mathbf{C} \to \mathbf{C}$ = automaton type



Abstract automata

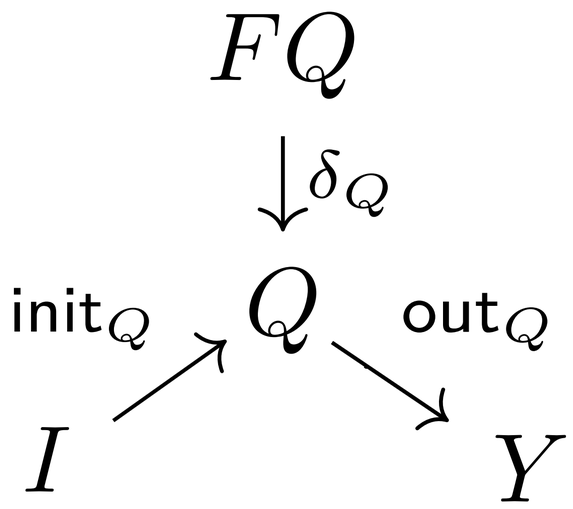
Category $\mathbf{C}$ = universe of state-spaces

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

DFAs

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

$F = (-) \times A$



Abstract automata

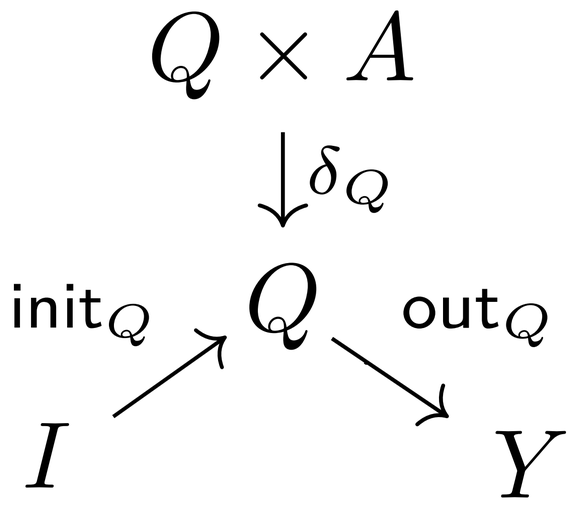
Category $\mathbf{C}$ = universe of state-spaces

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

DFAs

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

$F = (-) \times A$



Abstract automata

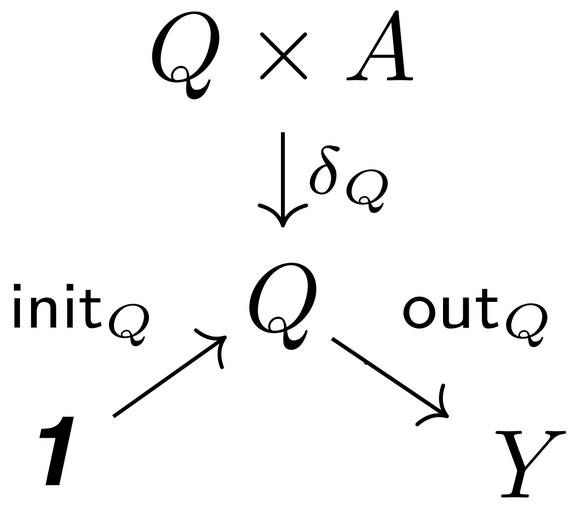
Category $\mathbf{C}$ = universe of state-spaces

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

DFAs

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

$F = (-) \times A$



Abstract automata

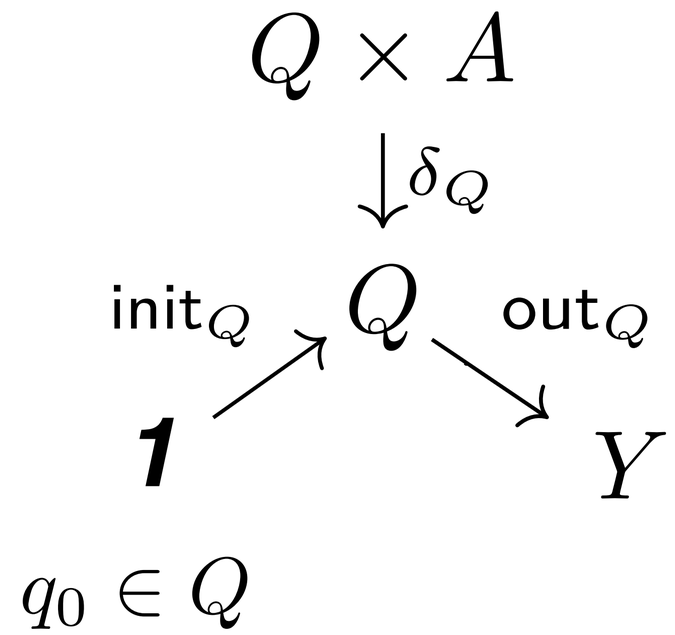
Category $\mathbf{C}$ = universe of state-spaces

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

DFAs

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

$F = (-) \times A$



$q_0 \in Q$

Abstract automata

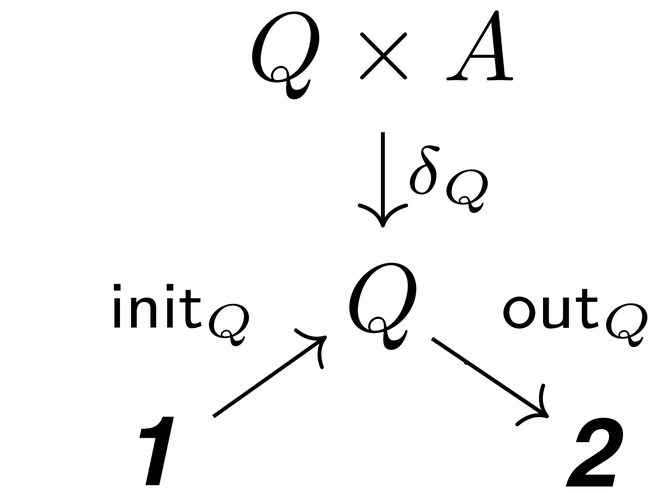
Category $\mathbf{C}$ = universe of state-spaces

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

DFAs

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

$F = (-) \times A$



$q_0 \in Q$

Abstract automata

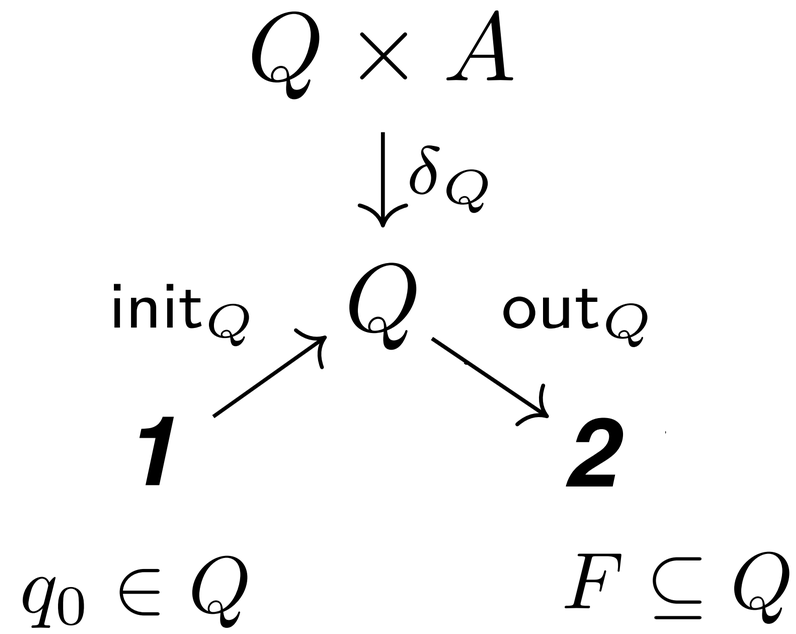
Category $\mathbf{C}$ = universe of state-spaces

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

DFAs

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

$F = (-) \times A$



Abstract learning

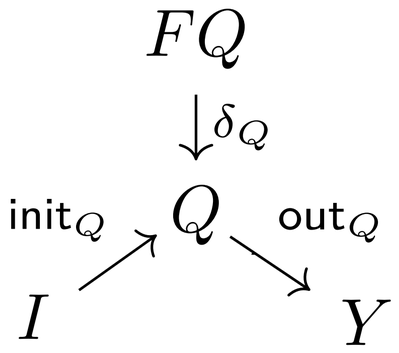
Abstract observation data structure

Abstract learning

Abstract observation data structure

approximates

Target minimal automaton

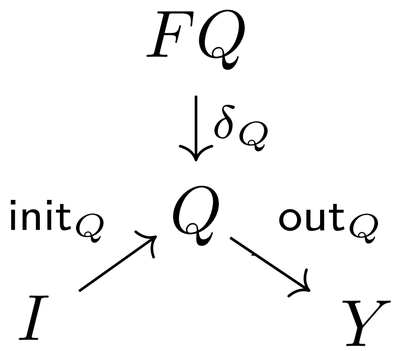


Abstract learning

Abstract observation data structure

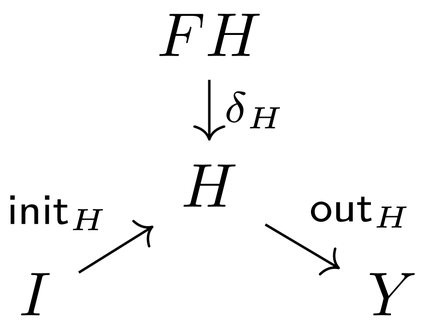
approximates

Target minimal automaton



abstract closedness and consistency

Hypothesis automaton

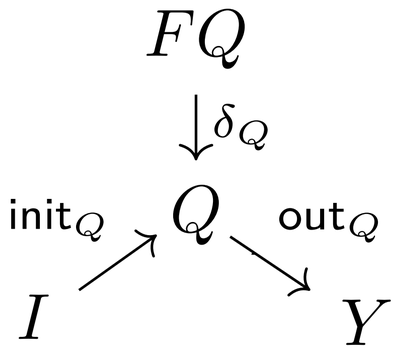


Abstract learning

Abstract observation data structure

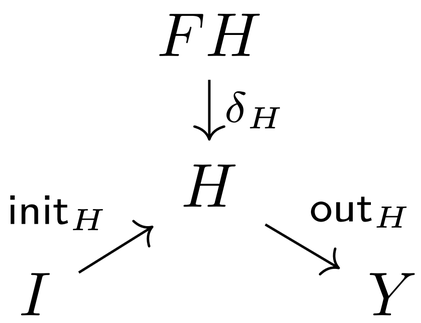
approximates

Target minimal automaton



abstract closedness and consistency

Hypothesis automaton



General correctness theorem

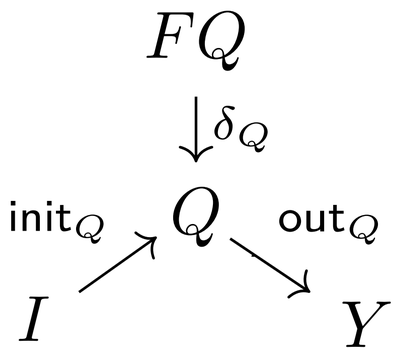
Guidelines for implementation

Abstract learning

Abstract observation data structure

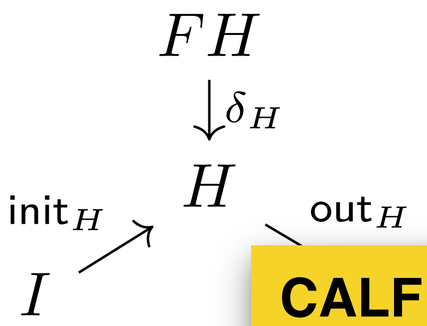
approximates

Target minimal automaton



abstract closedness and consistency

Hypothesis automaton



General correctness theorem

Guidelines for implementation

CALF: Categorical Automata Learning Framework (arXiv:1704.05676)

Gerco van Heerdt, Matteo Sammartino, Alexandra Silva

Degrees of freedom

Degrees of freedom

Change base category

Set	**DFAs**
Nom	**Nominal automata**
Vect	**Weighted automata**

Degrees of freedom

Change base category

Set	**DFAs**
Nom	**Nominal automata**
Vect	**Weighted automata**

Side-effects (via monads)

Powerset	**NFAs**
Powerset with intersection	**Universal automata**
Maybe monad	**Partial automata**

Degrees of freedom

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**
Maybe monad	**Partial automata**

Degrees of freedom

Change base category

Set	**DFAs**
Nom	**Nominal automata**
Vect	**Weighted automata**

Change main data structure

Discrimination trees

Learning Nominal Automata (POPL ’17)

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

Side-effects (via monads)

Powerset	**NFAs**
Powerset with intersection	**Universal automata**
Maybe monad	**Partial automata**

Degrees of freedom

Change base category

Change main data structure

Learning Nominal Automata (POPL '17)

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

Set	**DFAs**
Nom	**Nominal automata**
Vect	**Weighted automata**

Discrimination trees

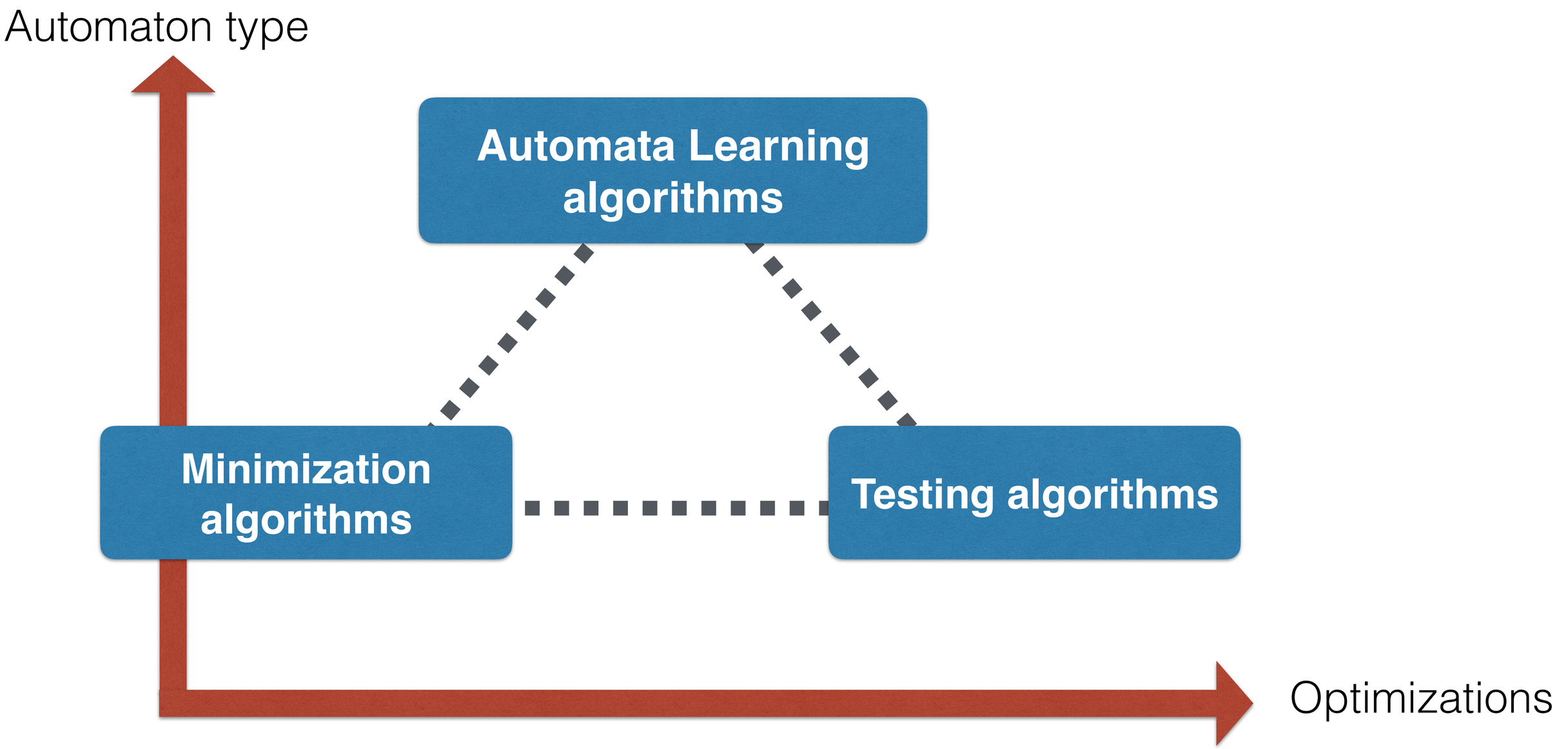
Learning Automata with Side-effects (**arXiv:1704.08055**)

Gerco van Heerdt, Matteo Sammartino, Alexandra Silva

Side-effects (via monads)

Powerset	**NFAs**
Powerset with intersection	**Universal automata**
Maybe monad	**Partial automata**

Connections with other algorithms



Abstraction helps a lot …

Abstraction helps a lot …

but there is no free lunch!

Ongoing and future work

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