Learning Nominal Automata

Joshua Moerman
(Radboud University)

Matteo Sammartino, Alexandra Silva
(University College London)

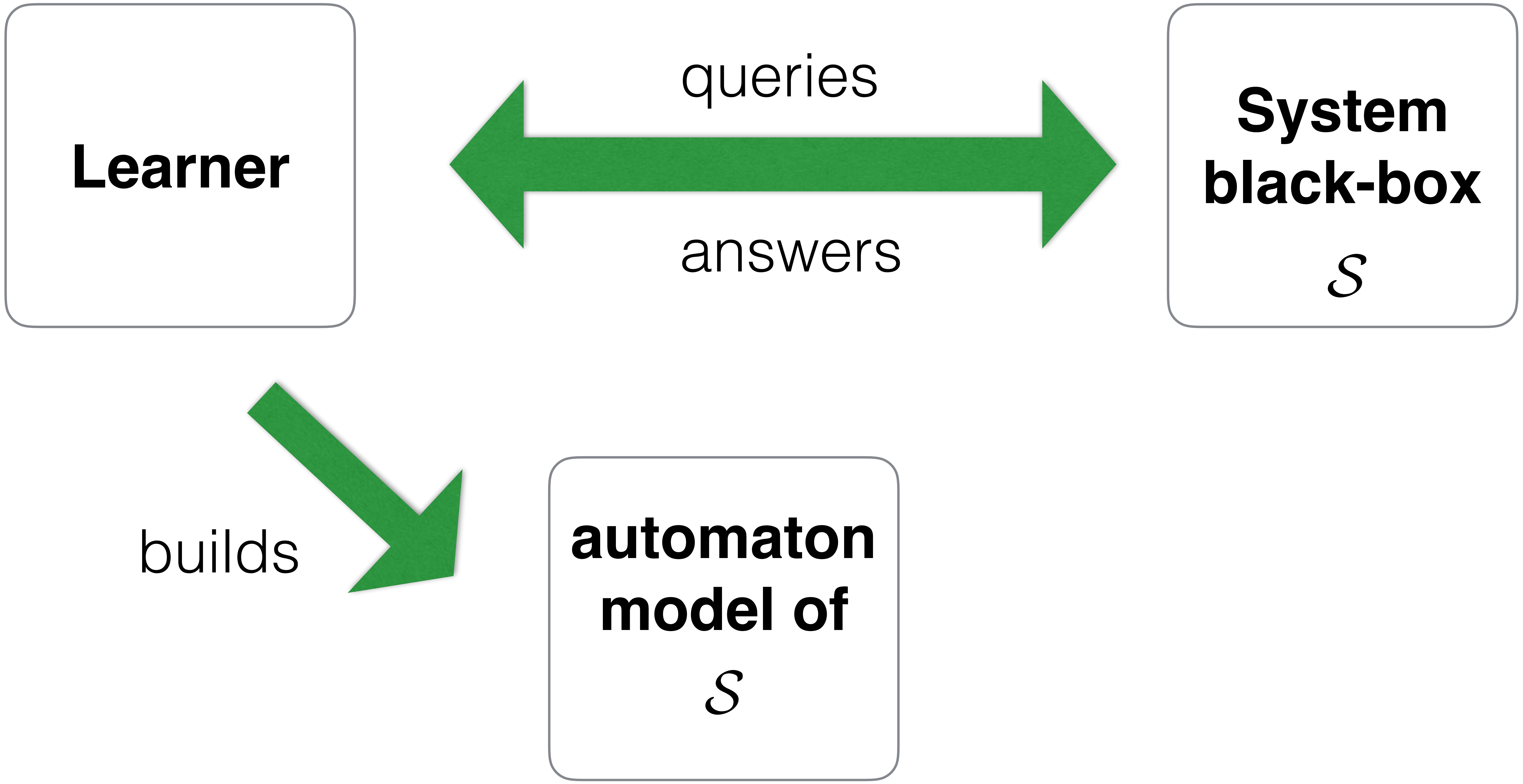
Bartek Klin, Michał Szynwelski
(Warsaw University)

POPL 2017
Paris

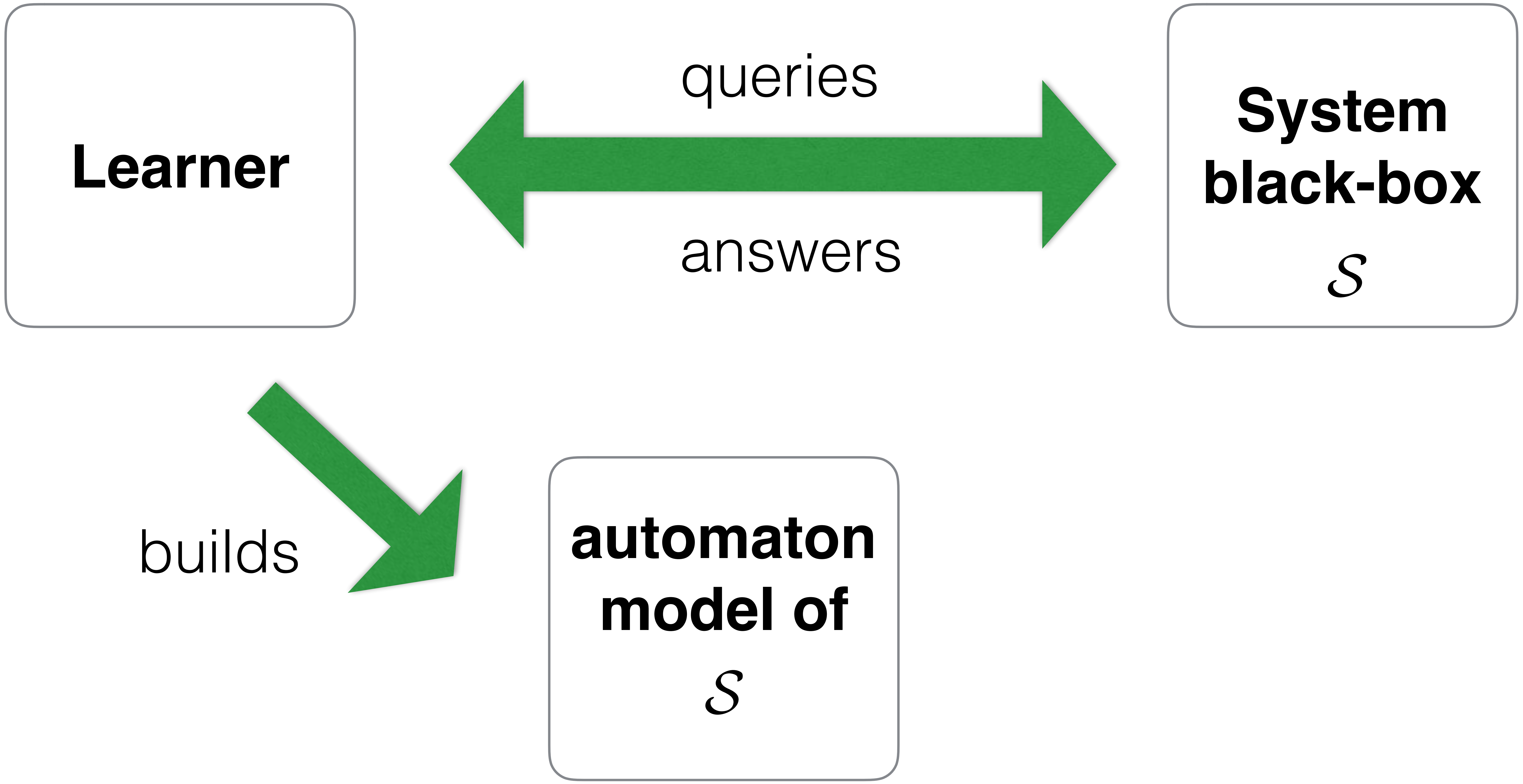
UCL ENGINEERING
Change the world

UCL

Active learning



Active 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

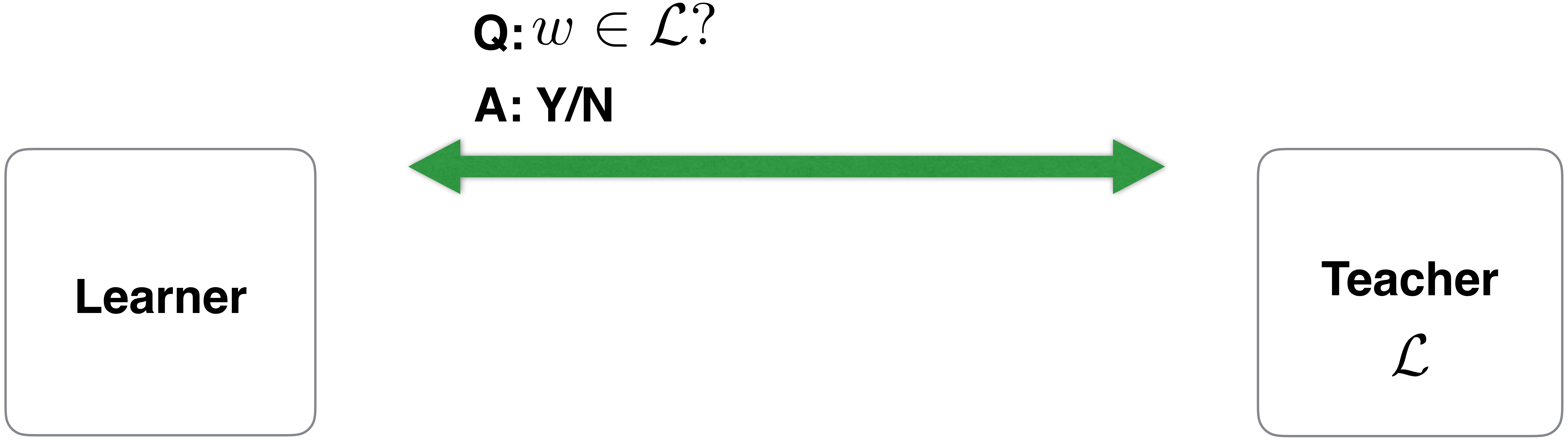
Teacher

$\mathcal{L}$

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

Q: $w \in \mathcal{L}?$

A: Y/N

Learner

Q: $\mathcal{L}(H) = \mathcal{L}?$

A: Y / N + counterexample

Teacher

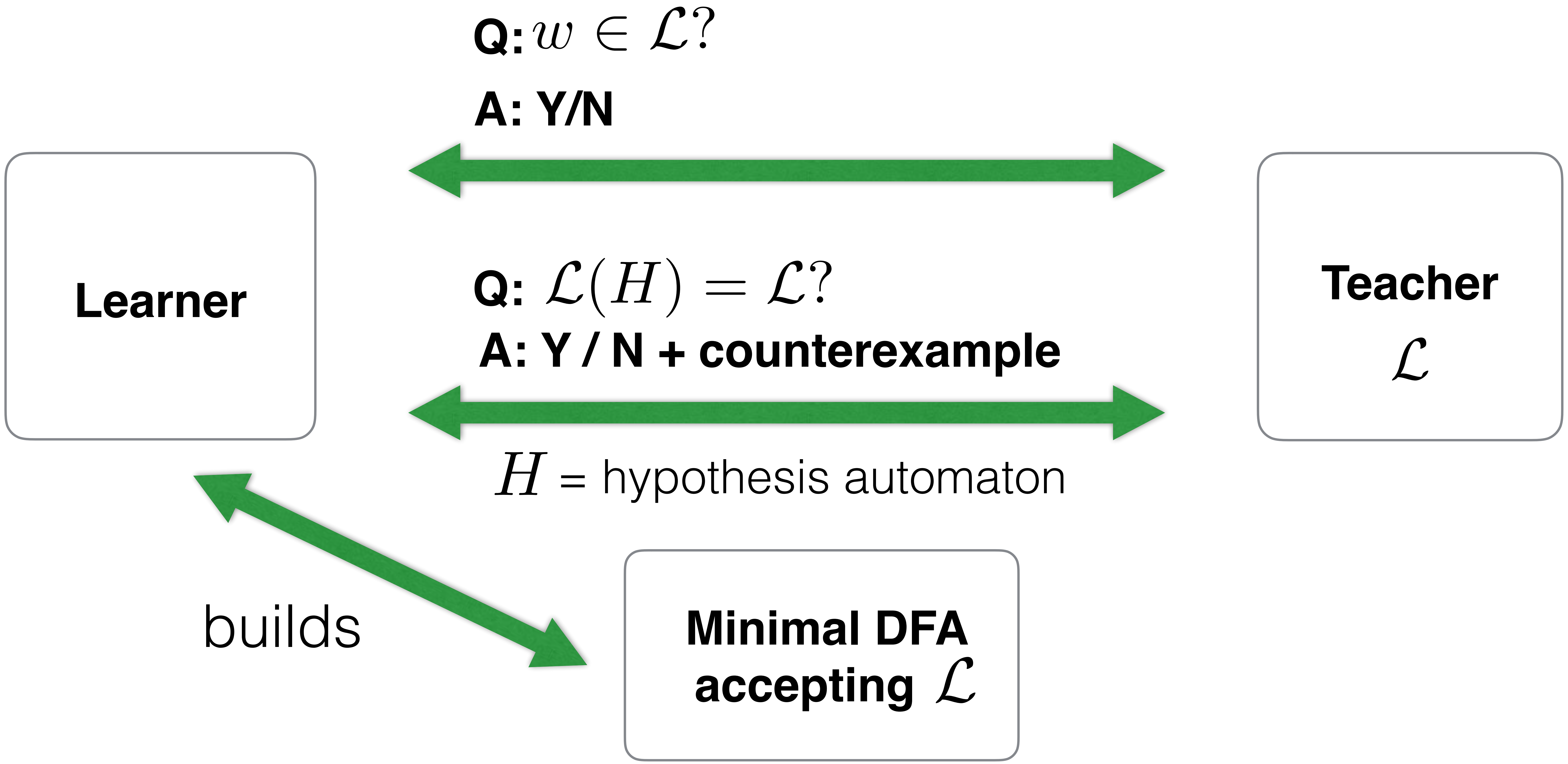
$\mathcal{L}$

H = hypothesis automaton

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

E

	ϵ	**a**	**aa**
ϵ	0	0	1
a	0	1	0
b	0	0	0

S

$S{\cdot}A$

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

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

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

Observation table

E

	ϵ	***a***	***aa***
ϵ	0	0	1
a	0	1	0
b	0	0	0

S: ϵ; $S{\cdot}A$: ***a***, ***b***

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

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

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

Hypothesis automaton

states = $\{row(s) \mid s \in S\}$

final states = $\{row(s) \mid s \in S, row(s)(\epsilon) = 1\}$

initial state = $row(\epsilon)$

transition function $\quad row(s) \xrightarrow{a} row(sa)$

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

states = $\{row(s) \mid s \in S\}$

final states ... $)(\epsilon) = 1\}$

initial state

Why is this correct?

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

Table properties

Closed

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

next state exists

Consistent

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

next state is unique

Table properties

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

Closed

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

next state exists

Consistent

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

next state is unique

Table properties

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

Closed

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

Fixed by extending the table

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

next state is unique

Pros of L* …

simple is beautiful

&

POWERFUL

Applications : Hardware verification, security/network protocols…

Generalizations : Mealy machines, I/O automata, …

… and shortcomings

L* learns **control-flow**

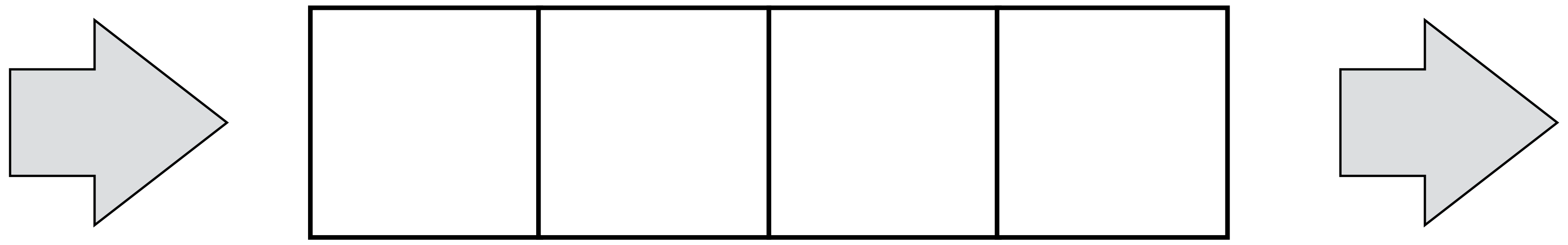
What if program model needs to express **data-flow?**

operations on **data values**

comparisons between data values

FIFO

push(x)



pop(y)

y = front element

Automata over infinite alphabets
(nominal automata)

Automata over infinite alphabets (nominal automata)

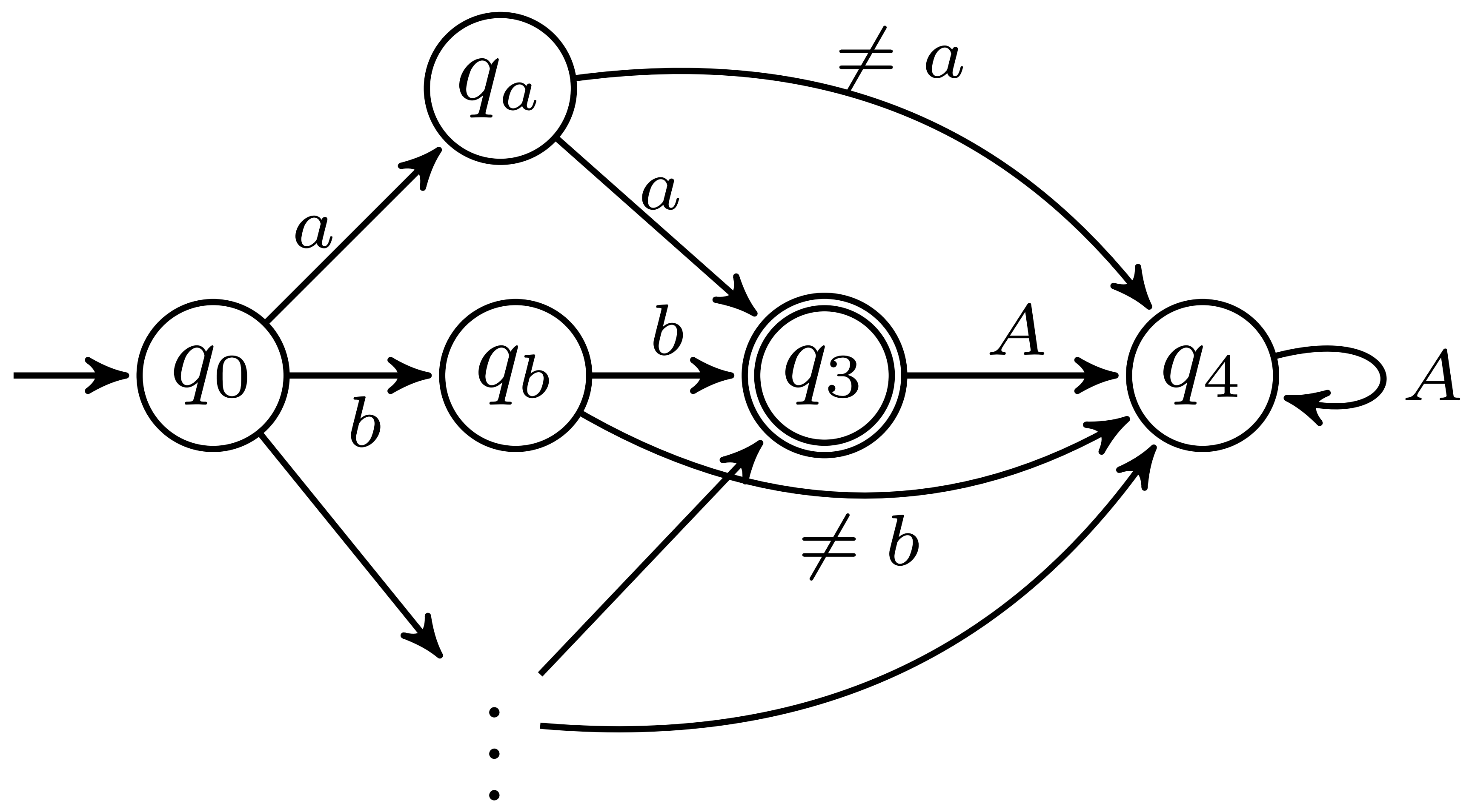
$$A = \{a, b, c, d, \dots\}$$ **infinite alphabet**

$$\mathcal{L} = \{aa, bb, cc, dd, \dots\}$$

Automata over infinite alphabets (nominal automata)

$$A = \{a, b, c, d, \ldots\}$$ **infinite alphabet**

$$\mathcal{L} = \{aa, bb, cc, dd, \ldots\}$$

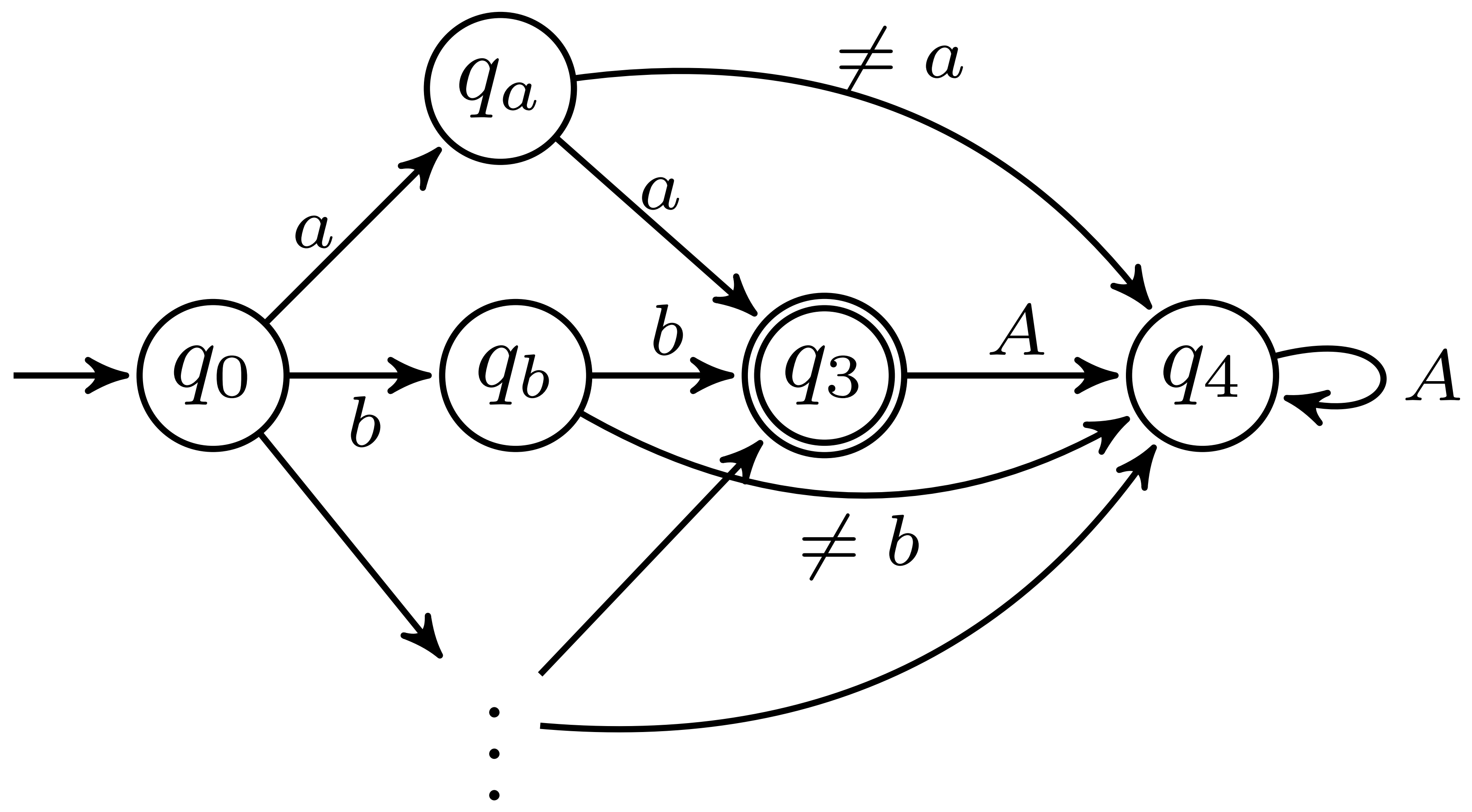


infinite automaton

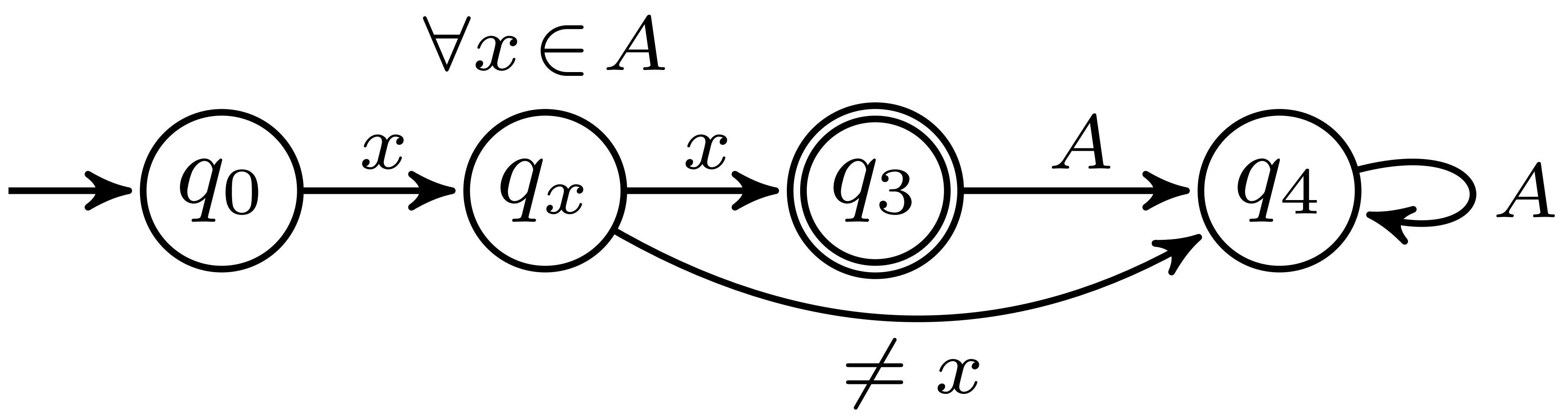
Automata over infinite alphabets (nominal automata)

$$A = \{a, b, c, d, \ldots\}$$ **infinite alphabet**

$$\mathcal{L} = \{aa, bb, cc, dd, \ldots\}$$



infinite automaton



but with a finite representation

How to learn them?

How to learn them?

Ad-hoc algorithm? **NO!**

How to learn them?

Ad-hoc algorithm? **NO!**

Challenges:

- table needs to be **infinite**
- code operates on **infinite sets**

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

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

How to learn them?

Ad-hoc algorithm? **NO!**

Challenges:

- table needs to be **infinite**
- code operates on **infinite sets**

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

Everything is "finitely representable"

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

A paradigm shift

(finite) sets ➡ (orbit-finite) nominal sets

functions ➡ equivariant functions

(change category from **Set** to **Nom**)

Nominal automata theory

Nominal Programming languages

Mikolaj Bojanczyk, Bartek Klin, Slawomir Lasota:
Automata with Group Actions. LICS 2011

Bartek Klin, Michal Szynwelski:
SMT Solving for Functional Programming over Infinite Structures. MSFP 2016

A paradigm shift

(finite) sets ➡ (orbit-finite) nominal sets

functions ➡ equivariant functions

(change category from **Set** to **Nom**)

Nominal L*

Nominal automata theory

Nominal Programming languages

Mikolaj Bojanczyk, Bartek Klin, Slawomir Lasota: **Automata with Group Actions**. LICS 2011

Bartek Klin, Michal Szynwelski: **SMT Solving for Functional Programming over Infinite Structures**. MSFP 2016

A paradigm shift

(finite) sets ➡ (orbit-finite) nominal sets

functions ➡ equivariant functions

(change category from **Set** to **Nom**)

Nominal L*

Nominal automata theory

N

First non-trivial application of a new programming paradigm (**NLambda**)

Mikolaj Bojanczyk, Bartek Klin, Slawomir Lasota: **Automata with Group Actions**. LICS 2011

Bartek [illegible]
SMT Solving for Functional Programming over Infinite Structures. MSFP 2016

A paradigm shift

(finite) sets ➡ (orbit-finite) nominal sets

functions ➡ equivariant functions

(change category from **Set** to **Nom**)

Nominal L*

Works with any (suitable) data domain

Nominal automata theory

Mikolaj Bojanczyk, Bartek Klin, Slawomir Lasota: **Automata with Group Actions**. LICS 2011

N

First non-trivial application of a new programming paradigm (**NLambda**)

Bartek

SMT Solving for Functional Programming over Infinite Structures. MSFP 2016

A paradigm shift

(finite) sets → (orbit-finite) nominal sets

functions → equivariant functions

(change category from **Set** to **Nom**)

Nominal L*

Works with any (suitable) data domain

Nominal automata theory

N

First non-trivial application of a new programming paradigm (**NLambda**)

Mikolaj Bojanczyk, Bartek Klin, Slawomir Lasota: **Automata with Group Actions**. LICS 2011

Bartek Klin, Michal Szynwelski: **SMT Solving for Functional Programming over Infinite Structures**. MSFP 2016

Eryk Kopczynski, Szymon Torunczyk: **LOIS: syntax and semantics.** POPL 2017

Correctness and termination

Correctness and termination

NLambda guarantees that **each line of code terminates**

Correctness and termination

NLambda guarantees that **each line of code terminates**

Algorithm correctness and termination **from scratch**?

Correctness and termination

NLambda guarantees that **each line of code terminates**

Algorithm correctness and termination **from scratch**?

Not really

Set-based proofs as **guidelines**

L* enjoys a nice **category-theoretic** generalization

Bart Jacobs, Alexandra Silva
Automata Learning: A Categorical Perspective, Horizons of the Minds 2014

What we've done

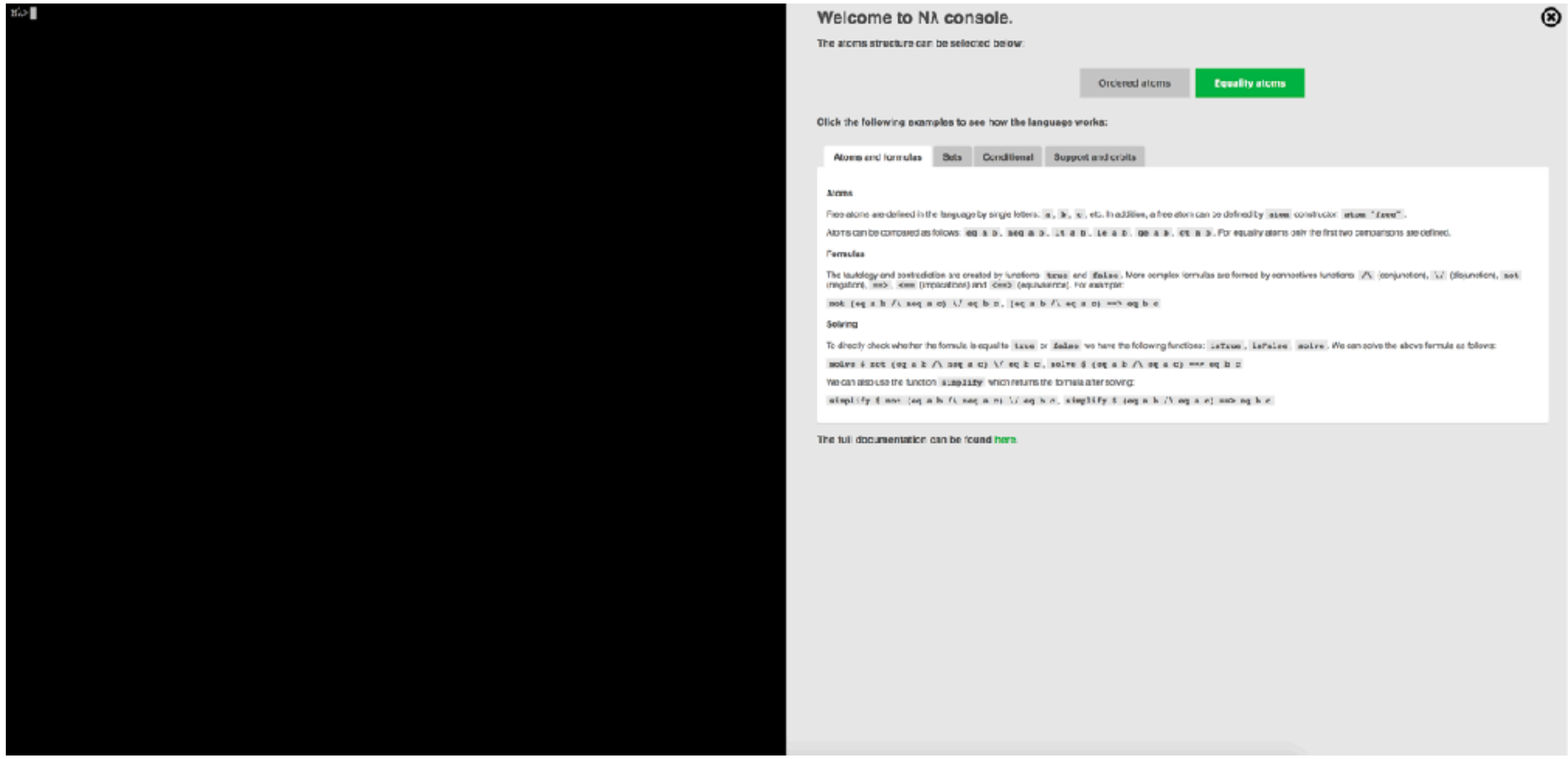
- Nominal L*
- **More in the paper**: variations, Nominal NL*
- NLambda (Haskell) Implementation
- Experimental results

What's next…

- Improve NLambda
- Other active learning algorithms
- Other optimizations
- Applications: large-scale software, crypto protocols…

Try it yourself

https://www.mimuw.edu.pl/~szynwelski/nlambda/



https://github.com/Jaxan/nominal-lstar