Formalized Run-Time Analysis of Active Learning Coalgebraically in Agda

Thorsten Wissmann
thorsten-wissmann.de

FAU Erlangen-Nürnberg, Germany

July 14, 2026

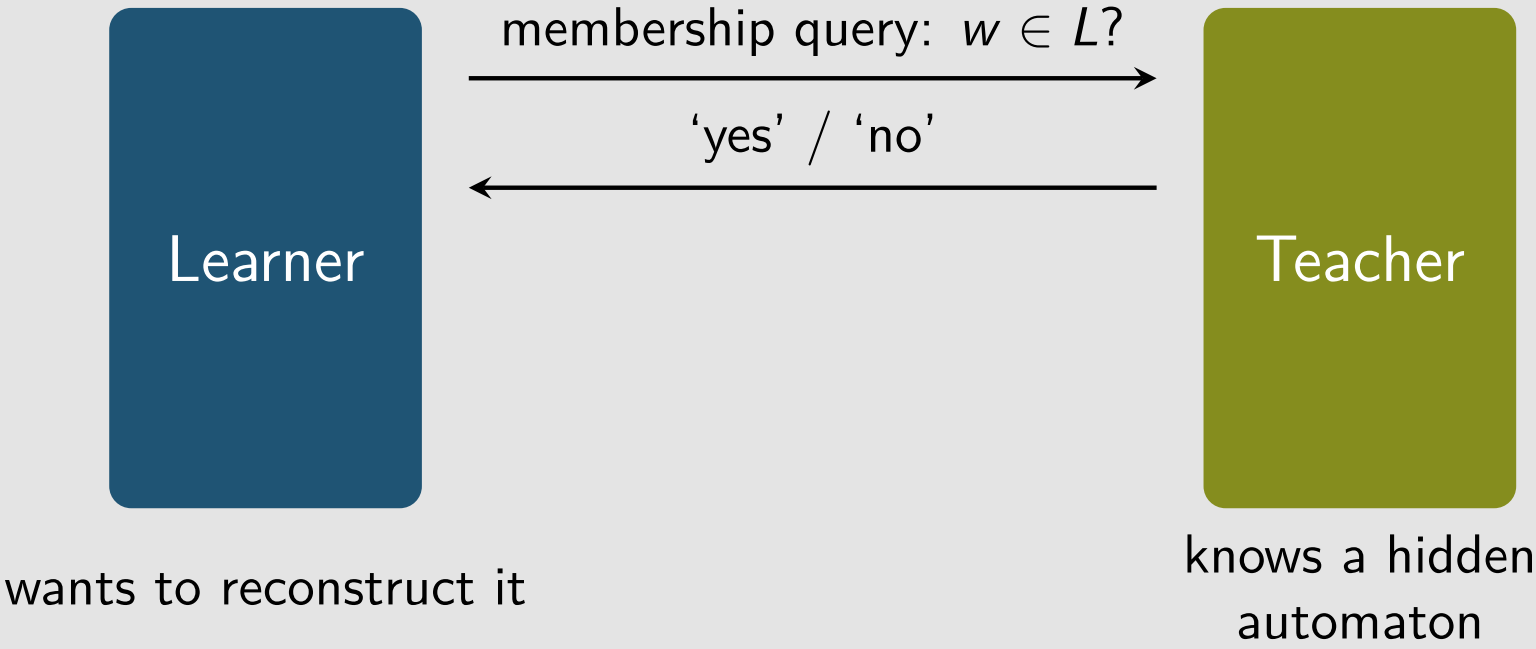
Active Automata Learning

Angluin '87

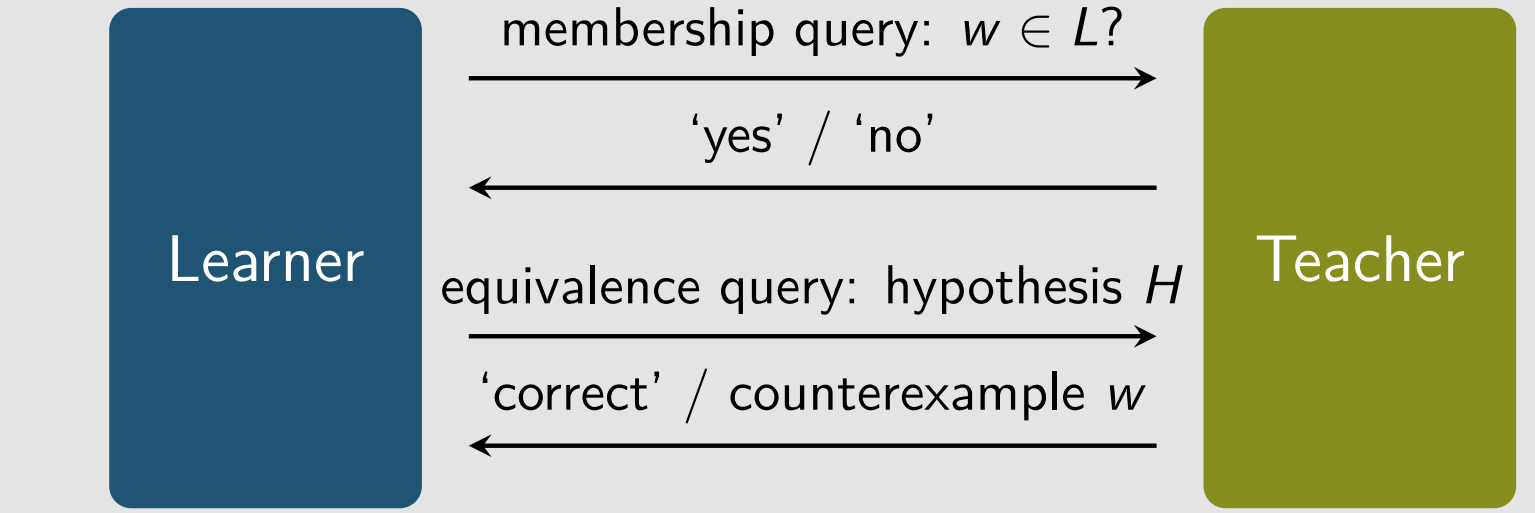
wants to reconstruct it

knows a hidden automaton

Active Automata Learning Angluin '87



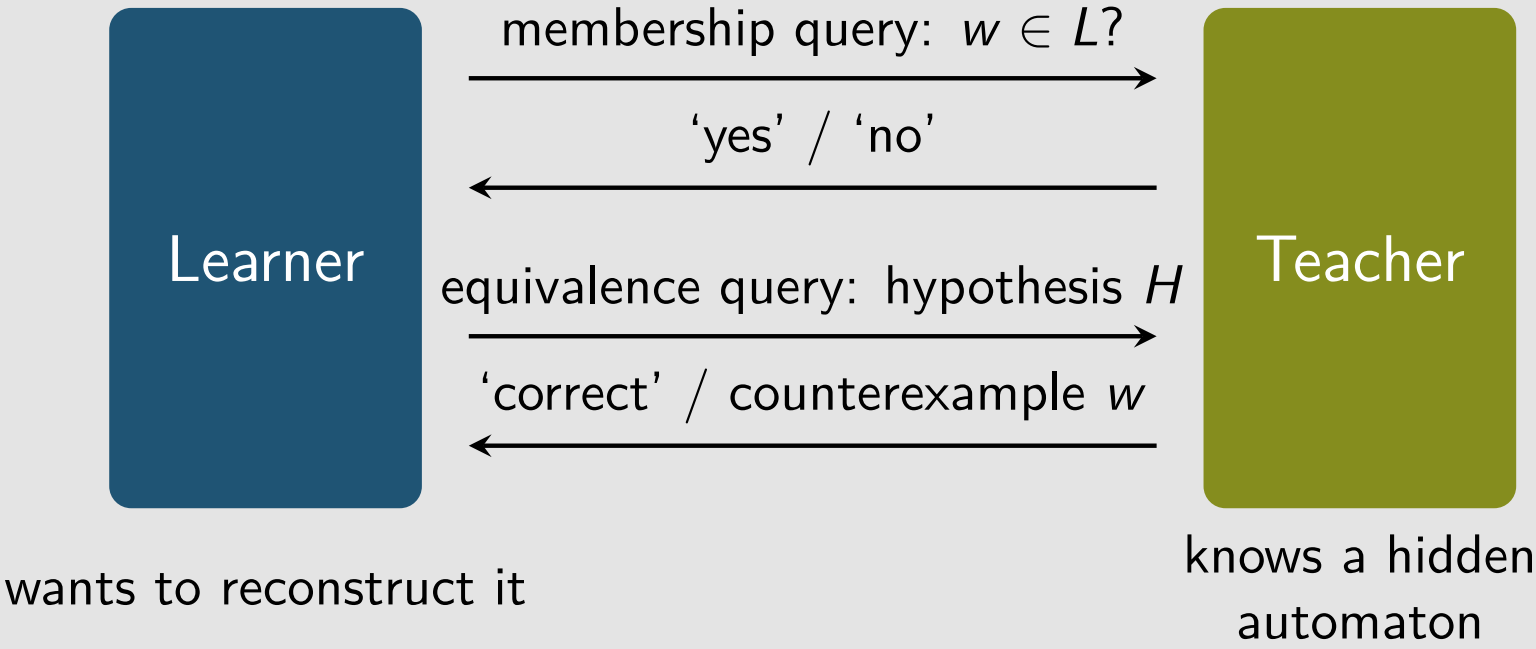
Active Automata Learning — Angluin '87



wants to reconstruct it

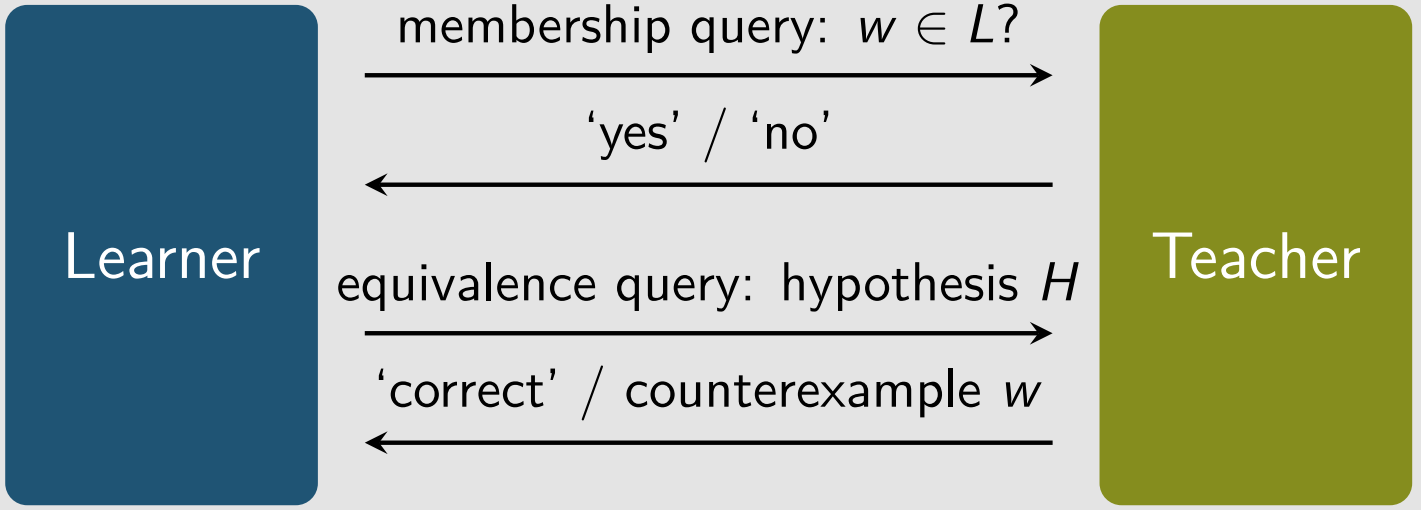
knows a hidden automaton

Active Automata Learning — Angluin '87



- 'correct' ends the game. Learner wins.

Active Automata Learning — Angluin '87



wants to reconstruct it

knows a hidden automaton

- 'correct' ends the game. Learner wins.
- Run-time = number of queries.

Question 1

In automata learning, what are the definitions of learner & teacher?

Question 1

In automata learning, what are the definitions of learner & teacher?

Angluin, 1987

Note that we have avoided giving formal definitions of Teachers, Learners, questions, dialogs and so on, preferring not to obscure the intuitive simplicity of the present paper.

Question 1

In automata learning, what are the definitions of learner & teacher?

Angluin, 1987

Note that we have avoided giving formal definitions of Teachers, Learners, questions, dialogs and so on, preferring not to obscure the intuitive simplicity of the present paper.

Question 2

Why do we(?) want formal definitions?

Question 1

In automata learning, what are the definitions of learner & teacher?

Angluin, 1987

Note that we have avoided giving formal definitions of Teachers, Learners, questions, dialogs and so on, preferring not to obscure the intuitive simplicity of the present paper.

Question 2

Why do we(?) want formal definitions?

Answer of this paper

Correctness & complexity proofs should operate on mathematically well-defined objects

Contributions

- Compact definitions: game type, learner, teacher, semantics.

Contributions

- Compact definitions: game type, learner, teacher, semantics.
- Generic proof principle for upper bounds. No hidden automaton needed.

Contributions

- Compact definitions: game type, learner, teacher, semantics.
- Generic proof principle for upper bounds. No hidden automaton needed.
- Fully formalized in Agda.

Contributions

- Compact definitions: game type, learner, teacher, semantics.
- Generic proof principle for upper bounds. No hidden automaton needed.
- Fully formalized in Agda.

Notion	Formal Definition	Example
Concept Class	a set $\mathbb{D}$	all DFAs over A
Game Type	functor $F\colon \mathsf{Set} \times \mathsf{Set} \to \mathsf{Set}$	DFA learning
Learner	C + map $C \to F(R, C)$	Angluin's L^*
Teacher	T + maps $F(R, X) \to (R + T \times X)^T$	adversarial teacher
Semantics	maps $F(R, X) \to F(R, 2^{\mathbb{D}} \times X)$	'on a query for w ...'

Running Example: Number Guessing

The game

Teacher thinks of $d \in \mathbb{N}$. Learner guesses. Teacher answers: 'correct', 'too-high', or 'too-low'.

Running Example: Number Guessing

The game

Teacher thinks of $d \in \mathbb{N}$. Learner guesses. Teacher answers: 'correct', 'too-high', or 'too-low'.

Query	Response
1	'too-low'
3	'too-high'
2	'correct'

learner wins

Running Example: Number Guessing

The game

Teacher thinks of $d \in \mathbb{N}$. Learner guesses. Teacher answers: 'correct', 'too-high', or 'too-low'.

Query	Response
1	'too-low'
3	'too-high'
2	'correct'

learner wins

Query	Response
5	'too-low'
8	'too-high'
7	'too-low'

learner wins:
teacher inconsistent!

Running Example: Number Guessing

The game

Teacher thinks of $d \in \mathbb{N}$. Learner guesses. Teacher answers: 'correct', 'too-high', or 'too-low'.

Query	Response
1	'too-low'
3	'too-high'
2	'correct'

learner wins

Query	Response
5	'too-low'
8	'too-high'
7	'too-low'

learner wins:
teacher inconsistent!

Query	Response
10	'too-low'
10^{10}	'too-low'
10^{1000}	'too-low'

teacher wins
... so far

Running Example: Number Guessing

The game

Teacher thinks of $d \in \mathbb{N}$. Learner guesses. Teacher answers: 'correct', 'too-high', or 'too-low'.

Query	Response
1	'too-low'
3	'too-high'
2	'correct'

learner wins

Query	Response
5	'too-low'
8	'too-high'
7	'too-low'

learner wins:
teacher inconsistent!

Query	Response
10	'too-low'
10^{10}	'too-low'
10^{1000}	'too-low'

teacher wins
... so far

Simple. But: who *is* the learner? Who *is* the teacher?

Learners, Formally

Definition ($W := \{\text{'too-high'}, \text{'too-low'}\}$)

A *learner* consists of: states C, results R, initial state $q_0 \in C$, and

$$c \colon C \longrightarrow \mathbb{N} \times R \times C^W$$

Learners, Formally

Definition ($W := \{\text{'too-high'}, \text{'too-low'}\}$)

A *learner* consists of: states C, results R, initial state $q_0 \in C$, and

$$c \colon C \longrightarrow \mathbb{N} \times R \times C^W$$

In state q with $c(q) = (n, r, f)$:

- guess $n \in \mathbb{N}$

Learners, Formally

Definition ($W := \{\text{‘too-high’}, \text{‘too-low’}\}$)

A *learner* consists of: states C, results R, initial state $q_0 \in C$, and

$$c \colon C \longrightarrow \mathbb{N} \times R \times C^W$$

In state q with $c(q) = (n, r, f)$:

- guess $n \in \mathbb{N}$
- wrong guess: continue in $f(\text{‘too-high’})$ or $f(\text{‘too-low’})$

Learners, Formally

Definition ($W := \{\text{'too-high'}, \text{'too-low'}\}$)

A *learner* consists of: states C, results R, initial state $q_0 \in C$, and

$$c\colon C \longrightarrow \mathbb{N} \times R \times C^W$$

In state q with $c(q) = (n, r, f)$:

- guess $n \in \mathbb{N}$
- wrong guess: continue in $f(\text{'too-high'})$ or $f(\text{'too-low'})$
- correct guess: game over, result r. No successor state!

Learners, Formally

Definition ($W := \{\text{'too-high'}, \text{'too-low'}\}$)

A *learner* consists of: states C, results R, initial state $q_0 \in C$, and

$$c\colon C \longrightarrow \mathbb{N} \times R \times C^W$$

In state q with $c(q) = (n, r, f)$:

- guess $n \in \mathbb{N}$
- wrong guess: continue in $f(\text{'too-high'})$ or $f(\text{'too-low'})$
- correct guess: game over, result r. No successor state!

Observation

A learner is a (possibly infinite) Moore automaton: input W, output $\mathbb{N} \times R$.

Example Learner: Binary Search

States = intervals. 'Where must the secret be?'

$$C := \{[n, \infty) \mid n \in \mathbb{N}\} \cup \{[n, m] \mid n, m \in \mathbb{N}\} \qquad q_0 = [0, \infty)$$

Example Learner: Binary Search

States = intervals. 'Where must the secret be?'

$$C := \{[n, \infty) \mid n \in \mathbb{N}\} \cup \{[n, m] \mid n, m \in \mathbb{N}\} \qquad q_0 = [0, \infty)$$

- Phase 1: on $[n, \infty)$, guess $2n + 1$. Doubling, until 'too-high'.

Example Learner: Binary Search

States = intervals. 'Where must the secret be?'

$$C := \{[n, \infty) \mid n \in \mathbb{N}\} \cup \{[n, m] \mid n, m \in \mathbb{N}\} \qquad q_0 = [0, \infty)$$

- Phase 1: on $[n, \infty)$, guess $2n + 1$. Doubling, until 'too-high'.
- Phase 2: on $[n, m]$, guess the middle. Halving.

Example Learner: Binary Search

States = intervals. 'Where must the secret be?'

$$C := \{[n, \infty) \mid n \in \mathbb{N}\} \cup \{[n, m] \mid n, m \in \mathbb{N}\} \qquad q_0 = [0, \infty)$$

- Phase 1: on $[n, \infty)$, guess $2n + 1$. Doubling, until 'too-high'.
- Phase 2: on $[n, m]$, guess the middle. Halving.

Knowing $\neq$ having guessed

In state $[n, n]$: learner *knows* the secret is n. Still needs one more guess.

Teachers, Formally

Definition

A *teacher* consists of: states T, initial state $s_0 \in T$, and

$$\delta \colon T \times \mathbb{N} \longrightarrow \{\text{‘correct’}\} + W \times T$$

Teachers, Formally

Definition

A *teacher* consists of: states T, initial state $s_0 \in T$, and

$$\delta\colon\ T \times \mathbb{N} \longrightarrow \{\text{`correct'}\} + W \times T$$

- T: the teacher's memory. *Stateless* if $|T| = 1$.

Teachers, Formally

Definition

A *teacher* consists of: states T, initial state $s_0 \in T$, and

$$\delta\colon T \times \mathbb{N} \longrightarrow \{\text{‘correct’}\} + W \times T$$

- T: the teacher's memory. *Stateless* if $|T| = 1$.
- ‘correct’ ends the game. Correct guess – or surrender.

Teachers, Formally

Definition

A *teacher* consists of: states T, initial state $s_0 \in T$, and

$$\delta\colon T \times \mathbb{N} \longrightarrow \{\text{`correct'}\} + W \times T$$

- T: the teacher's memory. *Stateless* if $|T| = 1$.
- 'correct' ends the game. Correct guess – or surrender.

Examples (stateless)

- The honest teacher for a fixed secret $d \in \mathbb{N}$.

Teachers, Formally

Definition

A *teacher* consists of: states T, initial state $s_0 \in T$, and

$$\delta\colon T \times \mathbb{N} \longrightarrow \{\text{'correct'}\} + W \times T$$

- T: the teacher's memory. *Stateless* if $|T| = 1$.
- 'correct' ends the game. Correct guess – or surrender.

Examples (stateless)

- The honest teacher for a fixed secret $d \in \mathbb{N}$.
- The teacher that always answers 'too-low'.
  Indistinguishable from secret $10^{1000} + 1$. Like learning a non-regular language.

The Adversarial Teacher

Idea: don't commit to a number. Keep an interval.

The Adversarial Teacher

Idea: don't commit to a number. Keep an interval.

- State: interval $[n, m]$. Everything in it: still plausible.

The Adversarial Teacher

Idea: don't commit to a number. Keep an interval.

- State: interval $[n, m]$. Everything in it: still plausible.
- Guess in the lower half: answer 'too-low', keep the upper half.
- Guess in the upper half: answer 'too-high', keep the lower half.

The Adversarial Teacher

Idea: don't commit to a number. Keep an interval.

- State: interval $[n, m]$. Everything in it: still plausible.
- Guess in the lower half: answer 'too-low', keep the upper half.
- Guess in the upper half: answer 'too-high', keep the lower half.
- 'correct' only when forced: $[k, k]$ and guess k.

The Adversarial Teacher

Idea: don't commit to a number. Keep an interval.

- State: interval $[n, m]$. Everything in it: still plausible.
- Guess in the lower half: answer 'too-low', keep the upper half.
- Guess in the upper half: answer 'too-high', keep the lower half.
- 'correct' only when forced: $[k, k]$ and guess k.

Consequence

Each answer keeps $\geq$ half of the interval alive.
Any learner needs more than $\log_2(m - n)$ guesses. (formal: later)

Generic Game Types

Same idea, other games? Abstract over the query interface.

Generic Game Types

Same idea, other games? Abstract over the query interface.

Definition

A *game type* is a functor $F\colon \mathrm{Set} \times \mathrm{Set} \to \mathrm{Set}$.

Just a construction: sets R, $X \mapsto$ set $F(R, X)$.

Generic Game Types

Same idea, other games? Abstract over the query interface.

Definition

A *game type* is a functor $F\colon \mathrm{Set} \times \mathrm{Set} \to \mathrm{Set}$.

Just a construction: sets R, $X \mapsto$ set $F(R, X)$.

Example

Number guessing: $F(R, X) = \mathbb{N} \times R \times X^W$

Generic Game Types

Same idea, other games? Abstract over the query interface.

Definition

A *game type* is a functor $F\colon \mathrm{Set} \times \mathrm{Set} \to \mathrm{Set}$.

Just a construction: sets R, $X \mapsto$ set $F(R, X)$.

Example

Number guessing: $F(R, X) = \mathbb{N} \times R \times X^W$

Definition

A *learner* for F: states C, results R, $q_0 \in C$, and

$$c\colon C \longrightarrow F(R, C)$$

Game Type for DFA Learning

$$F(R, X) := \underbrace{A^* \times X^2}_{\text{MQ}} + \underbrace{\text{DFA} \times R \times X^{A^*}}_{\text{EQ}}$$

Game Type for DFA Learning

$$F(R, X) := \underbrace{A^* \times X^2}_{\text{MQ}} + \underbrace{\text{DFA} \times R \times X^{A^*}}_{\text{EQ}}$$

- MQ(w, f): is w in the language? Continue in $f(0)$ or $f(1)$.

Game Type for DFA Learning

$$F(R, X) := \underbrace{A^* \times X^2}_{\text{MQ}} + \underbrace{\text{DFA} \times R \times X^{A^*}}_{\text{EQ}}$$

- MQ(w, f): is w in the language? Continue in $f(0)$ or $f(1)$.
- EQ(H, r, f): hypothesis H. On counterexample w: continue in $f(w)$.

Game Type for DFA Learning

$$F(R, X) := \underbrace{A^* \times X^2}_{\text{MQ}} + \underbrace{\text{DFA} \times R \times X^{A^*}}_{\text{EQ}}$$

- MQ(w, f): is w in the language? Continue in $f(0)$ or $f(1)$.
- EQ(H, r, f): hypothesis H. On counterexample w: continue in $f(w)$.

Note

No R in the MQ summand.

So: membership queries cannot end the game.

Generic Teachers

Definition

A *teacher* for F: states T, initial state $s_0 \in T$, and a *natural* family

$$\alpha_{R,X} \colon F(R,X) \longrightarrow (R + X \times T)^T$$

Generic Teachers

Definition

A *teacher* for F: states T, initial state $s_0 \in T$, and a *natural* family

$$\alpha_{R,X} \colon F(R,X) \longrightarrow (R + X \times T)^T$$

Sanity check (essentially Yoneda)

For number guessing, such α correspond exactly to

$$\delta \colon T \times \mathbb{N} \longrightarrow \{\text{`correct'}\} + W \times T$$

Similarly for DFA learning: one map for MQ, one for EQ.

Generic Teachers

Definition

A *teacher* for F: states T, initial state $s_0 \in T$, and a *natural* family

$$\alpha_{R,X} \colon F(R,X) \longrightarrow (R + X \times T)^T$$

Sanity check (essentially Yoneda)

For number guessing, such α correspond exactly to

$$\delta \colon T \times \mathbb{N} \longrightarrow \{\text{`correct'}\} + W \times T$$

Similarly for DFA learning: one map for MQ, one for EQ.

Playing the game

Compose learner and teacher: $g(q,s) = \alpha(c(q))(s)$,
$g \colon C \times T \longrightarrow R + C \times T$.
Iterate. Stop when the result lands in R.

Game Semantics

Fix a concept class $\mathbb{D}$: what the teacher might hide.
Number guessing: $\mathbb{D} = \mathbb{N}$. DFA learning: $\mathbb{D} = \mathrm{DFA}$.

Game Semantics

Fix a concept class $\mathbb{D}$: what the teacher might hide.
Number guessing: $\mathbb{D} = \mathbb{N}$. DFA learning: $\mathbb{D} = \mathrm{DFA}$.

Definition

A *semantics* for F: maps

$$[\![-]\!] \colon F(R, X) \to F(R, 2^{\mathbb{D}} \times X) \qquad \text{(preserving the } X\text{-part)}$$

Each successor state gets a predicate on $\mathbb{D}$:
'which secrets are consistent with this answer?'

Game Semantics

Fix a concept class $\mathbb{D}$: what the teacher might hide.
Number guessing: $\mathbb{D} = \mathbb{N}$. DFA learning: $\mathbb{D} = \mathrm{DFA}$.

Definition

A *semantics* for F: maps

$$[\![-]\!] \colon F(R, X) \to F(R, 2^{\mathbb{D}} \times X) \qquad \text{(preserving the } X\text{-part)}$$

Each successor state gets a predicate on $\mathbb{D}$:
'which secrets are consistent with this answer?'

Example: guess H in number guessing

- 'too-low'-branch: $\{d \in \mathbb{N} \mid d > H\}$
- 'too-high'-branch: $\{d \in \mathbb{N} \mid d < H\}$

When is the Game Won?

Problem: no hidden automaton to compare against.
Instead: judge the teacher by its answers.

When is the Game Won?

Problem: no hidden automaton to compare against.
Instead: judge the teacher by its answers.

Definition

$d \in \mathbb{D}$ is *still possible* after n rounds:
the teacher's first n responses are all consistent with d.

When is the Game Won?

Problem: no hidden automaton to compare against.
Instead: judge the teacher by its answers.

Definition

$d \in \mathbb{D}$ is *still possible* after n rounds:
the teacher's first n responses are all consistent with d.

Definition

The learner *finds d within n rounds*:
d is *not* still possible after n rounds.

Two cases:

When is the Game Won?

Problem: no hidden automaton to compare against.
Instead: judge the teacher by its answers.

Definition

$d \in \mathbb{D}$ is *still possible* after n rounds:
the teacher's first n responses are all consistent with d.

Definition

The learner *finds d within n rounds*:
d is *not* still possible after n rounds.

Two cases:

- d is the secret: teacher accepts within n rounds.
- d is not the secret: some response refutes d.

When is the Game Won?

Problem: no hidden automaton to compare against.
Instead: judge the teacher by its answers.

Definition

$d \in \mathbb{D}$ is *still possible* after n rounds:
the teacher's first n responses are all consistent with d.

Definition

The learner *finds d within n rounds*:
d is *not* still possible after n rounds.

Two cases:

- d is the secret: teacher accepts within n rounds.
- d is not the secret: some response refutes d.

Example: guessing $0, 1, 2, 3, \ldots$ finds every d within $1 + d$ rounds.

Proof Principle: Step-Bounded Learners

Goal: an upper bound $b\colon \mathbb{D} \to \mathbb{N}$ on the queries needed to find d.

Proof Principle: Step-Bounded Learners

Goal: an upper bound $b : \mathbb{D} \to \mathbb{N}$ on the queries needed to find d.

Definition: step-bounded by b

Equip the learner with:

- tick : $C \to \mathbb{N}$ – queries asked so far
- allows : $C \to 2^{\mathbb{D}}$ – concepts not yet refuted

such that:

Proof Principle: Step-Bounded Learners

Goal: an upper bound $b\colon \mathbb{D} \to \mathbb{N}$ on the queries needed to find d.

Definition: step-bounded by b

Equip the learner with:

- $\mathrm{tick}\colon C \to \mathbb{N}$ – queries asked so far
- $\mathrm{allows}\colon C \to 2^{\mathbb{D}}$ – concepts not yet refuted

such that:

- $\mathrm{allows}(q_0) = \mathbb{D}$
- $d \in \mathrm{allows}(q)$ implies $\mathrm{tick}(q) < b(d)$

Proof Principle: Step-Bounded Learners

Goal: an upper bound $b\colon \mathbb{D} \to \mathbb{N}$ on the queries needed to find d.

Definition: step-bounded by b

Equip the learner with:

- $\mathrm{tick}\colon C \to \mathbb{N}$ – queries asked so far
- $\mathrm{allows}\colon C \to 2^{\mathbb{D}}$ – concepts not yet refuted

such that:

- $\mathrm{allows}(q_0) = \mathbb{D}$
- $d \in \mathrm{allows}(q)$ implies $\mathrm{tick}(q) < b(d)$
- every consistent answer: allows preserved, tick strictly increases

Proof Principle: Step-Bounded Learners

Goal: an upper bound $b\colon \mathbb{D} \to \mathbb{N}$ on the queries needed to find d.

Definition: step-bounded by b

Equip the learner with:

- $\mathrm{tick}\colon C \to \mathbb{N}$ – queries asked so far
- $\mathrm{allows}\colon C \to 2^{\mathbb{D}}$ – concepts not yet refuted

such that:

- $\mathrm{allows}(q_0) = \mathbb{D}$
- $d \in \mathrm{allows}(q)$ implies $\mathrm{tick}(q) < b(d)$
- every consistent answer: allows preserved, tick strictly increases

Last item, in number guessing (guess H, $d \in \mathrm{allows}(q)$):

$$d > H \;\Rightarrow\; d \in \mathrm{allows}(f(\text{`too-low'})),\;\; \mathrm{tick}(q) < \mathrm{tick}(f(\text{`too-low'}))$$

Inconsistent answers: no condition at all.

Main Theorem

Theorem

Learner step-bounded by $b\colon \mathbb{D} \to \mathbb{N}$ $\implies$ for *every* teacher, the learner finds every $d \in \mathbb{D}$ within $b(d)$ rounds.

Main Theorem

Theorem

Learner step-bounded by $b\colon \mathbb{D} \to \mathbb{N}$ $\implies$ for *every* teacher, the learner finds every $d \in \mathbb{D}$ within $b(d)$ rounds.

Example: binary search, in Agda

Step-bounded by

$$b(d) = 3 + 2 \cdot \lfloor \log_2(d) \rfloor$$

Main Theorem

Theorem

Learner step-bounded by $b\colon \mathbb{D} \to \mathbb{N}$ $\implies$ for *every* teacher, the learner finds every $d \in \mathbb{D}$ within $b(d)$ rounds.

Example: binary search, in Agda

Step-bounded by

$$b(d) = 3 + 2 \cdot \lfloor \log_2(d) \rfloor$$

- refine the state space, define tick per interval

Main Theorem

Theorem

Learner step-bounded by $b\colon \mathbb{D} \to \mathbb{N}$ $\implies$ for *every* teacher, the learner finds every $d \in \mathbb{D}$ within $b(d)$ rounds.

Example: binary search, in Agda

Step-bounded by

$$b(d) = 3 + 2 \cdot \lfloor \log_2(d) \rfloor$$

- refine the state space, define tick per interval
- main effort: transitions preserve “allows”

Main Theorem

Theorem

Learner step-bounded by $b\colon \mathbb{D} \to \mathbb{N}$ $\implies$ for *every* teacher, the learner finds every $d \in \mathbb{D}$ within $b(d)$ rounds.

Example: binary search, in Agda

Step-bounded by

$$b(d) = 3 + 2 \cdot \lfloor \log_2(d) \rfloor$$

- refine the state space, define tick per interval
- main effort: transitions preserve “allows”

Holds for *all* teachers. Even the adversarial one.

Lower Bounds: Mind the Quantifiers

Proposition

Fix any stateless teacher (not answering ‘too-low’ everywhere).
Then *some* learner finds every secret within 2 rounds.

Lower Bounds: Mind the Quantifiers

Proposition

Fix any stateless teacher (not answering 'too-low' everywhere).
Then *some* learner finds every secret within 2 rounds.

- Teacher fixed first. Some learner just guesses right.

Lower Bounds: Mind the Quantifiers

Proposition

Fix any stateless teacher (not answering 'too-low' everywhere).
Then *some* learner finds every secret within 2 rounds.

- Teacher fixed first. Some learner just guesses right.
- Round 2: at most one query to convict inconsistency.

Lower Bounds: Mind the Quantifiers

Proposition

Fix any stateless teacher (not answering 'too-low' everywhere).
Then *some* learner finds every secret within 2 rounds.

- Teacher fixed first. Some learner just guesses right.
- Round 2: at most one query to convict inconsistency.
- So: quantifier order matters!

Lower Bounds: Mind the Quantifiers

Proposition

Fix any stateless teacher (not answering 'too-low' everywhere).
Then *some* learner finds every secret within 2 rounds.

- Teacher fixed first. Some learner just guesses right.
- Round 2: at most one query to convict inconsistency.
- So: quantifier order matters!

Theorem (lower bound)

Adversarial teacher on $[n, n+m]$:
for *every* learner, some d is still possible after $\lfloor \log_2(m) \rfloor$ rounds.

Binary search is optimal.

Should You Play?

Steve Ballmer's interview question (TV, 2016)

I'm thinking of a number between 1 and 100. [. . .] If you get it the first guess, I give you 5 bucks, [otherwise] 4 bucks, 3, 2, 1, 0, you pay me a buck, you pay me 2, . . . Do you want to play?

`https://youtu.be/svCYbkS0Sjk`

Should You Play?

Steve Ballmer's interview question (TV, 2016)

I'm thinking of a number between 1 and 100. [. . .] If you get it the first guess, I give you 5 bucks, [otherwise] 4 bucks, 3, 2, 1, 0, you pay me a buck, you pay me 2, . . . Do you want to play?

`https://youtu.be/svCYbkS0Sjk`

- Guess 7 or later: you pay.

Should You Play?

Steve Ballmer's interview question (TV, 2016)

I'm thinking of a number between 1 and 100. [...] If you get it the first guess, I give you 5 bucks, [otherwise] 4 bucks, 3, 2, 1, 0, you pay me a buck, you pay me 2, ... Do you want to play?

`https://youtu.be/svCYbkS0Sjk`

- Guess 7 or later: you pay.
- Lower bound theorem: $\lfloor \log_2(99) \rfloor = 6$.
  The teacher can force 7 guesses.

Should You Play?

Steve Ballmer's interview question (TV, 2016)

I'm thinking of a number between 1 and 100. [. . .] If you get it the first guess, I give you 5 bucks, [otherwise] 4 bucks, 3, 2, 1, 0, you pay me a buck, you pay me 2, . . . Do you want to play?

`https://youtu.be/svCYbkS0Sjk`

- Guess 7 or later: you pay.
- Lower bound theorem: $\lfloor \log_2(99) \rfloor = 6$.
  The teacher can force 7 guesses.

- So: don't play.
  (Unless the teacher is honest and uniformly random: expected +\$0.20. Graham-Cumming '24)

Variation 1: Long Counterexamples

New variant = new game type + semantics. Everything else stays generic.

Variation 1: Long Counterexamples

New variant = new game type + semantics. Everything else stays generic.

- No guarantee: teachers may return *long* counterexamples.

Variation 1: Long Counterexamples

New variant = new game type + semantics. Everything else stays generic.

- No guarantee: teachers may return *long* counterexamples.
- So run-times are also parametric in the counterexample length. Rivest, Schapire '93

Variation 1: Long Counterexamples

New variant = new game type + semantics. Everything else stays generic.

- No guarantee: teachers may return *long* counterexamples.
- So run-times are also parametric in the counterexample length. Rivest, Schapire '93

Adjustment

$\mathbb{D} := \mathrm{DFA} \times \mathbb{N}$ – automaton + bound on counterexample length.
Semantics of EQ: counterexample w is consistent with (M, ℓ) if

$$L(M)(w) \neq L(H)(w) \quad \text{and} \quad |w| \leq \ell$$

Bounds $b\colon \mathbb{D} \to \mathbb{N}$ now see size *and* counterexample length.
The learner does not know the maximum counterexample length!

Variation 2: Restricted Learning

Angluin '87

Equivalence queries answer only 'yes'/'no'. No counterexample.

Variation 2: Restricted Learning — Angluin '87

Equivalence queries answer only 'yes'/'no'. No counterexample.

Adjustment

Only one successor state on EQ:

$$F(R, X) := A^* \times X^2 + \mathrm{DFA} \times R \times X$$

Variation 2: Restricted Learning — Angluin '87

Equivalence queries answer only 'yes'/'no'. No counterexample.

Adjustment

Only one successor state on EQ:

$$F(R, X) := A^* \times X^2 + \mathrm{DFA} \times R \times X$$

Proposition

Alphabet size ≥ 2, any polynomial P:
there is a teacher that keeps, for every learner at least one DFA with n states possible after $P(n)$ queries by the learner.

So: learning is not possible within polynomial many queries.

Variation 3: Mealy Machines

Output queries instead of membership queries:
input word in A^+ is sent to the last output symbol O.

Variation 3: Mealy Machines

Output queries instead of membership queries:
input word in A^+ is sent to the last output symbol O.

Functor

$$F(R, X) = A^+ \times X^O + \text{Mealy} \times R \times X^{A^+}$$

Variation 3: Mealy Machines

Output queries instead of membership queries:
input word in A^+ is sent to the last output symbol O.

Functor

$$F(R, X) = A^+ \times X^O + \text{Mealy} \times R \times X^{A^+}$$

Generic teacher notion instantiates to:

MQ: $T \times A^+ \longrightarrow O \times T$ EQ: $T \times \text{Mealy} \longrightarrow 1 + A^+ \times T$

Theorem

The $L^\#$ learning algorithm needs at most

$$(|A| + 1) \cdot n \cdot (n + 1) + (n + 1) \cdot \lfloor \log_2 m \rfloor + 1$$

queries to learn a hidden Mealy machines with n states and counter-examples of length at most m.

Variation 4: Full Trace

Adjustment

Return full trace of output symbols along the run:

$$F(R, X) = \coprod_{n \in \mathbb{N}} A^n \times X^{O^n} + \text{Mealy} \times R \times X^{A^*}$$

Variation 4: Full Trace

Adjustment

Return full trace of output symbols along the run:

$$F(R, X) = \coprod_{n \in \mathbb{N}} A^n \times X^{O^n} + \text{Mealy} \times R \times X^{A^*}$$

Generic teacher notion instantiates to:

$$\text{MQ}_n \colon T \times A^n \longrightarrow O^n \times T \qquad \text{EQ} \colon T \times \text{Mealy} \longrightarrow 1 + A^* \times T$$

Conclusions & Future Work

Summary

- Learner = coalgebra. Teacher = natural transformation.
- Step-counting for upper bounds. Adversarial teachers for lower bounds.
- Number guessing: matching bounds $\mathcal{O}(\log d)$, in Agda.

Conclusions & Future Work

Summary

- Learner = coalgebra. Teacher = natural transformation.
- Step-counting for upper bounds. Adversarial teachers for lower bounds.
- Number guessing: matching bounds $\mathcal{O}(\log d)$, in Agda.

Future work

Formalization: `p8.cs.fau.de/active-learning-agda/`

Conclusions & Future Work

Summary

- Learner = coalgebra. Teacher = natural transformation.
- Step-counting for upper bounds. Adversarial teachers for lower bounds.
- Number guessing: matching bounds $\mathcal{O}(\log d)$, in Agda.

Future work

- $(R + X \times T)^T$ is a monad. Swap it: e.g. probabilistic teachers. Moggi '91

Formalization: `p8.cs.fau.de/active-learning-agda/`

Conclusions & Future Work

Summary

- Learner = coalgebra. Teacher = natural transformation.
- Step-counting for upper bounds. Adversarial teachers for lower bounds.
- Number guessing: matching bounds $\mathcal{O}(\log d)$, in Agda.

Future work

- $(R + X \times T)^T$ is a monad. Swap it: e.g. probabilistic teachers. Moggi '91
- Counters beyond $\mathbb{N}$: count MQ and EQ separately.

Formalization: `p8.cs.fau.de/active-learning-agda/`


Angluin, Dana. “Learning regular sets from queries and counterexamples”. *Information and Computation* 75.2 (1987), pp. 87–106. ISSN: 0890-5401. DOI: https://doi.org/10.1016/0890-5401(87)90052-6. URL: http://www.sciencedirect.com/science/article/pii/0890540187900526.


Angluin, Dana. “Queries and Concept Learning”. *Mach. Learn.* 2.4 (1987), pp. 319–342. DOI: 10.1007/BF00116828. URL: https://doi.org/10.1007/BF00116828.


Graham-Cumming, John. “Steve Ballmer’s incorrect binary search interview question”. (2024). URL: https://blog.jgc.org/2024/09/steve-ballmers-binary-search-interview.html.


Moggi, Eugenio. “Notions of Computation and Monads”. *Inf. Comput.* 93.1 (1991), pp. 55–92. DOI: 10.1016/0890-5401(91)90052-4. URL: https://doi.org/10.1016/0890-5401(91)90052-4.


Rivest, R.L., R.E. Schapire. “Inference of Finite Automata Using Homing Sequences”. *Inf. Comput.* 103.2 (1993), pp. 299–347. DOI: 10.1006/inco.1993.1021.