A categorical approach to automata learning and minimization – part 3

Daniela Petrişan

Université Paris Cité, IRIF, France

TACL'24, Barcelona, 24-28 June 2024

INSTITUT DE RECHERCHE EN INFORMATIQUE FONDAMENTALE

Université de Paris

Automata Learning

Automata Learning

- A classical subject with a wide range of applications: adaptive model checking, verification, learning network invariants and interface specifications.
  (see, e.g., *Martin Leucker. Learning meets verification, 2007*)

Automata Learning

- A classical subject with a wide range of applications: adaptive model checking, verification, learning network invariants and interface specifications.
  (see, e.g., *Martin Leucker. Learning meets verification, 2007*)
- The most famous learning algorithm for automata is the L^*-algorithm of Dana Angluin.
  D. Angluin, Learning Regular Sets from Queries and Counterexamples, Information and Computation, 1978

The L^*-algorithm

- **Goal:** learn a regular language of words L.

The L^*-algorithm

- **Goal:** learn a regular language of words L.
- The algorithm interacts with a teacher who knows L by asking two types of queries:

The L^*-algorithm

- **Goal:** learn a regular language of words L.
- The algorithm interacts with a teacher who knows L by asking two types of queries:
    1. Membership queries: Does a word belong to the language ?

The L^*-algorithm

- **Goal:** learn a regular language of words L.
- The algorithm interacts with a teacher who knows L by asking two types of queries:
  1. Membership queries: Does a word belong to the language ?
  2. Equivalence queries: It gives the teacher a hypothesis automaton and asks whether it accepts the language L.

The L^*-algorithm

- **Goal:** learn a regular language of words L.
- The algorithm interacts with a teacher who knows L by asking two types of queries:
  1. Membership queries: Does a word belong to the language ?
  2. Equivalence queries: It gives the teacher a hypothesis automaton and asks whether it accepts the language L. If no, the teacher provides a counter-example.

The L^*-algorithm

- **Goal:** learn a regular language of words L.
- The algorithm interacts with a teacher who knows L by asking two types of queries:
  1. Membership queries: Does a word belong to the language ?
  2. Equivalence queries: It gives the teacher a hypothesis automaton and asks whether it accepts the language L. If no, the teacher provides a counter-example.
- The algorithm stops when the teacher agrees that the hypothesis automaton accepts the language L.

The L^*-algorithm: some definitions

- At each step, we maintain a pair of sets of words (Q, T), starting with $(\{\epsilon\}, \{\epsilon\})$.
  - Q —> potential states for the hypothesis automaton
  - T —> test words used to define an equivalence relation coarser than the Myhill-Nerode equivalence.

The L^*-algorithm: some definitions

- At each step, we maintain a pair of sets of words (Q, T), starting with $(\{\epsilon\}, \{\epsilon\})$.
  - Q —> potential states for the hypothesis automaton
  - T —> test words used to define an equivalence relation coarser than the Myhill-Nerode equivalence.
- the T-equivalence relation: $w \sim_T v$ iff $\forall u \in T. \quad wu \in L \Leftrightarrow vu \in L$

The L^*-algorithm: some definitions

- At each step, we maintain a pair of sets of words (Q, T), starting with $(\{\epsilon\}, \{\epsilon\})$.
  - Q —> potential states for the hypothesis automaton
  - T —> test words used to define an equivalence relation coarser than the Myhill-Nerode equivalence.
- the T-equivalence relation: $w \sim_T v$ iff $\forall u \in T. \quad wu \in L \Leftrightarrow vu \in L$
- closedness : $\forall q \in Q \,.\forall a \in A \,.\exists p \in Q \,.\, p \sim_T qa$.

The L^*-algorithm: some definitions

- At each step, we maintain a pair of sets of words (Q, T), starting with $(\{\epsilon\}, \{\epsilon\})$.
  - Q —> potential states for the hypothesis automaton
  - T —> test words used to define an equivalence relation coarser than the Myhill-Nerode equivalence.
- the T-equivalence relation: $w \sim_T v$ iff $\forall u \in T. \quad wu \in L \Leftrightarrow vu \in L$
- closedness : $\forall q \in Q . \forall a \in A . \exists p \in Q . \; p \sim_T qa.$
- consistency : $\forall q, q' \in Q . \forall a \in A . \; q \sim_T q' \Rightarrow qa \sim_T q'a$

The L^*-algorithm: some definitions

- At each step, we maintain a pair of sets of words (Q, T), starting with $(\{\epsilon\}, \{\epsilon\})$.
  - Q —> potential states for the hypothesis automaton
  - T —> test words used to define an equivalence relation coarser than the Myhill-Nerode equivalence.
- the T-equivalence relation: $w \sim_T v$ iff $\forall u \in T. \quad wu \in L \Leftrightarrow vu \in L$
- closedness : $\forall q \in Q \,.\forall a \in A \,.\exists p \in Q \,.\ p \sim_T qa$.
- consistency : $\forall q, q' \in Q \,.\forall a \in A \,.\ \ q \sim_T q' \Rightarrow qa \sim_T q'a$
- When (Q, T) is closed and consistent it is possible to build a hypothesis automaton $\mathcal{H}(Q, T)$

The L^*-algorithm

$Q = T := \{\varepsilon\}$
repeat
  while (Q, T) not closed and consistent
    if (Q, T) is not closed enlarge Q
      $(\forall q \in Q \,.\forall a \in A \,.\exists p \in Q \,.\, p \sim_T qa)$
    if (Q, T) is not consistent enlarge T
      $(\forall q, q' \in Q \,.\forall a \in A \,.\ q \sim_T q' \Rightarrow qa \sim_T q'a)$
  ask an equivalence query for $\mathcal{H}(Q, T)$
  if the answer is no then
    add the counterexample and its
    prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

The L^*-algorithm

$Q = T := \{\varepsilon\}$
repeat
  while (Q, T) not closed and consistent
    if (Q, T) is not closed enlarge Q
      $(\forall q \in Q \,.\, \forall a \in A \,.\, \exists p \in Q \,.\, p \sim_T qa)$
    if (Q, T) is not consistent enlarge T
      $(\forall q, q' \in Q \,.\, \forall a \in A \,.\; q \sim_T q' \Rightarrow qa \sim_T q'a)$
  ask an equivalence query for $\mathcal{H}(Q, T)$
  if the answer is no then
    add the counterexample and its prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

The L^*-algorithm

$Q = T := \{\varepsilon\}$
repeat
  while (Q, T) not closed and consistent
    if (Q, T) is not closed enlarge Q
      $(\forall q \in Q . \forall a \in A . \exists p \in Q . p \sim_T qa)$
    if (Q, T) is not consistent enlarge T
      $(\forall q, q' \in Q . \forall a \in A . q \sim_T q' \Rightarrow qa \sim_T q'a)$
  ask an equivalence query for $\mathcal{H}(Q, T)$
  if the answer is no then
    add the counterexample and its prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon\}$	$\{\varepsilon\}$

The L^*-algorithm

$Q = T := \{\varepsilon\}$
repeat
while (Q, T) not closed and consistent
if (Q, T) is not closed enlarge Q
($\forall q \in Q\,.\forall a \in A\,.\exists p \in Q\,.\; p \sim_T qa$)
if (Q, T) is not consistent enlarge T
($\forall q, q' \in Q\,.\forall a \in A\,.\;\; q \sim_T q' \Rightarrow qa \sim_T q'a$)
ask an equivalence query for $\mathcal{H}(Q, T)$
if the answer is no then
add the counterexample and its prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon\}$	$\{\varepsilon\}$

The L^*-algorithm

$Q = T := \{\varepsilon\}$

repeat

  while (Q, T) not closed and consistent

    if (Q, T) is not closed enlarge Q

      $(\forall q \in Q \,.\, \forall a \in A \,.\, \exists p \in Q \,.\, p \sim_T qa)$

    if (Q, T) is not consistent enlarge T

      $(\forall q, q' \in Q \,.\, \forall a \in A \,.\; q \sim_T q' \Rightarrow qa \sim_T q'a)$

  ask an equivalence query for $\mathcal{H}(Q, T)$

  if the answer is no then

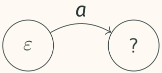
    add the counterexample and its prefixes to Q

until the answer is yes

return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon\}$	$\{\varepsilon\}$

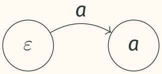


The L^*-algorithm

$Q = T := \{\varepsilon\}$
repeat
  while (Q, T) not closed and consistent
    if (Q, T) is not closed enlarge Q
      $(\forall q \in Q\,.\forall a \in A\,.\exists p \in Q\,.\ p \sim_T qa)$
    if (Q, T) is not consistent enlarge T
      $(\forall q, q' \in Q\,.\forall a \in A\,.\ \ q \sim_T q' \Rightarrow qa \sim_T q'a)$
  ask an equivalence query for $\mathcal{H}(Q, T)$
  if the answer is no then
    add the counterexample and its
    prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon, a\}$	$\{\varepsilon\}$

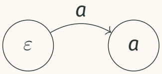


The L^*-algorithm

$Q = T := \{\varepsilon\}$
repeat
  while (Q, T) not closed and consistent
    if (Q, T) is not closed enlarge Q
      $(\forall q \in Q \,.\, \forall a \in A \,.\, \exists p \in Q \,.\, p \sim_T qa)$
    if (Q, T) is not consistent enlarge T
      $(\forall q, q' \in Q \,.\, \forall a \in A \,.\; q \sim_T q' \Rightarrow qa \sim_T q'a)$
  ask an equivalence query for $\mathcal{H}(Q, T)$
  if the answer is no then
    add the counterexample and its
    prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon, a\}$	$\{\varepsilon\}$

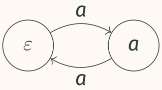


The L^*-algorithm

$Q = T := \{\varepsilon\}$
repeat
  while (Q, T) not closed and consistent
    if (Q, T) is not closed enlarge Q
      $(\forall q \in Q . \forall a \in A . \exists p \in Q . \, p \sim_T qa)$
    if (Q, T) is not consistent enlarge T
      $(\forall q, q' \in Q . \forall a \in A . \; q \sim_T q' \Rightarrow qa \sim_T q'a)$
  ask an equivalence query for $\mathcal{H}(Q, T)$
  if the answer is no then
    add the counterexample and its prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon, a\}$	$\{\varepsilon\}$



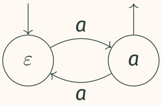
The L^*-algorithm

$Q = T := \{\varepsilon\}$
repeat
  while (Q, T) not closed and consistent
    if (Q, T) is not closed enlarge Q
      $(\forall q \in Q . \forall a \in A . \exists p \in Q . \ p \sim_T qa)$
    if (Q, T) is not consistent enlarge T
      $(\forall q, q' \in Q . \forall a \in A . \ q \sim_T q' \Rightarrow qa \sim_T q'a)$
  ask an equivalence query for $\mathcal{H}(Q, T)$
  if the answer is no then
    add the counterexample and its prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon, a\}$	$\{\varepsilon\}$

$\mathcal{H}(\{\varepsilon, a\}, \{\varepsilon\})$



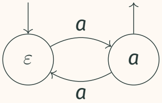
The L^*-algorithm

$Q = T := \{\varepsilon\}$
repeat
  while (Q, T) not closed and consistent
    if (Q, T) is not closed enlarge Q
      $(\forall q \in Q \,.\, \forall a \in A \,.\, \exists p \in Q \,.\, p \sim_T qa)$
    if (Q, T) is not consistent enlarge T
      $(\forall q, q' \in Q \,.\, \forall a \in A \,.\; q \sim_T q' \Rightarrow qa \sim_T q'a)$
  ask an equivalence query for $\mathcal{H}(Q, T)$
  if the answer is no then
    add the counterexample and its prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon, a\}$	$\{\varepsilon\}$

$\mathcal{H}(\{\varepsilon, a\}, \{\varepsilon\})$



Teacher: "No! *aaa* is a counterex."

The L^*-algorithm

$Q = T := \{\varepsilon\}$

repeat

while (Q, T) not closed and consistent

if (Q, T) is not closed enlarge Q

$(\forall q \in Q \,.\, \forall a \in A \,.\, \exists p \in Q \,.\; p \sim_T qa)$

if (Q, T) is not consistent enlarge T

$(\forall q, q' \in Q \,.\, \forall a \in A \,.\; q \sim_T q' \Rightarrow qa \sim_T q'a)$

ask an equivalence query for $\mathcal{H}(Q, T)$

if the answer is no then

add the counterexample and its prefixes to Q

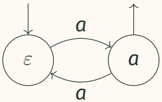
until the answer is yes

return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon, a\}$	$\{\varepsilon\}$

$\mathcal{H}(\{\varepsilon, a\}, \{\varepsilon\})$



Teacher: “No! *aaa* is a counterex.”

The L^*-algorithm

$Q = T := \{\varepsilon\}$

repeat

  while (Q, T) not closed and consistent

    if (Q, T) is not closed enlarge Q

      $(\forall q \in Q \,.\, \forall a \in A \,.\, \exists p \in Q \,.\, p \sim_T qa)$

    if (Q, T) is not consistent enlarge T

      $(\forall q, q' \in Q \,.\, \forall a \in A \,.\; q \sim_T q' \Rightarrow qa \sim_T q'a)$

  ask an equivalence query for $\mathcal{H}(Q, T)$

  if the answer is no then

    add the counterexample and its prefixes to Q

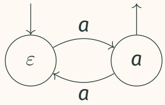
until the answer is yes

return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon, a, aa, aaa\}$	$\{\varepsilon\}$

$\mathcal{H}(\{\varepsilon, a\}, \{\varepsilon\})$



Teacher: “No! *aaa* is a counterex.”

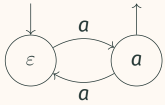
The L^*-algorithm

$Q = T := \{\varepsilon\}$
repeat
  while (Q, T) not closed and consistent
    if (Q, T) is not closed enlarge Q
      $(\forall q \in Q \,.\, \forall a \in A \,.\, \exists p \in Q \,.\, p \sim_T qa)$
    if (Q, T) is not consistent enlarge T
      $(\forall q, q' \in Q \,.\, \forall a \in A \,.\; q \sim_T q' \Rightarrow qa \sim_T q'a)$
  ask an equivalence query for $\mathcal{H}(Q, T)$
  if the answer is no then
    add the counterexample and its prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon, a, aa, aaa\}$	$\{\varepsilon\}$

$\mathcal{H}(\{\varepsilon, a\}, \{\varepsilon\})$



Teacher: “No! *aaa* is a counterex.”

The L^*-algorithm

$Q = T := \{\varepsilon\}$

repeat

  while (Q, T) not closed and consistent

    if (Q, T) is not closed enlarge Q

      $(\forall q \in Q \,.\, \forall a \in A \,.\, \exists p \in Q \,.\, p \sim_T qa)$

    if (Q, T) is not consistent enlarge T

      $(\forall q, q' \in Q \,.\, \forall a \in A \,.\; q \sim_T q' \Rightarrow qa \sim_T q'a)$

  ask an equivalence query for $\mathcal{H}(Q, T)$

  if the answer is no then

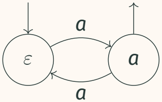
    add the counterexample and its prefixes to Q

until the answer is yes

return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon, a, aa, aaa\}$	$\{\varepsilon\}$

$\mathcal{H}(\{\varepsilon, a\}, \{\varepsilon\})$ 

Teacher: “No! *aaa* is a counterex.”

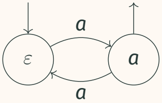
The L^*-algorithm

$Q = T := \{\varepsilon\}$
repeat
  while (Q, T) not closed and consistent
    if (Q, T) is not closed enlarge Q
      $(\forall q \in Q \,.\, \forall a \in A \,.\, \exists p \in Q \,.\; p \sim_T qa)$
    if (Q, T) is not consistent enlarge T
      $(\forall q, q' \in Q \,.\, \forall a \in A \,.\; q \sim_T q' \Rightarrow qa \sim_T q'a)$
  ask an equivalence query for $\mathcal{H}(Q, T)$
  if the answer is no then
    add the counterexample and its
    prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon, a, aa, aaa\}$	$\{\varepsilon\}$

$\mathcal{H}(\{\varepsilon, a\}, \{\varepsilon\})$



Teacher: "No! *aaa* is a counterex."

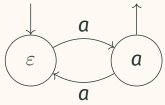
$\varepsilon \sim_{\{\varepsilon\}} aa$, but $a \not\sim_{\{\varepsilon\}} aaa$

The L^*-algorithm

$Q = T := \{\varepsilon\}$
repeat
  while (Q, T) not closed and consistent
    if (Q, T) is not closed enlarge Q
      $(\forall q \in Q\,.\forall a \in A\,.\exists p \in Q\,.\ p \sim_T qa)$
    if (Q, T) is not consistent enlarge T
      $(\forall q, q' \in Q\,.\forall a \in A\,.\ q \sim_T q' \Rightarrow qa \sim_T q'a)$
  ask an equivalence query for $\mathcal{H}(Q, T)$
  if the answer is no then
    add the counterexample and its prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon, a, aa, aaa\}$	$\{\varepsilon, a\}$

$\mathcal{H}(\{\varepsilon, a\}, \{\varepsilon\})$ 

Teacher: "No! *aaa* is a counterex."

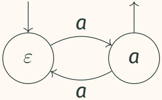
The L^*-algorithm

$Q = T := \{\varepsilon\}$
repeat
  while (Q, T) not closed and consistent
    if (Q, T) is not closed enlarge Q
      $(\forall q \in Q \,.\, \forall a \in A \,.\, \exists p \in Q \,.\, p \sim_T qa)$
    if (Q, T) is not consistent enlarge T
      $(\forall q, q' \in Q \,.\, \forall a \in A \,.\; q \sim_T q' \Rightarrow qa \sim_T q'a)$
  ask an equivalence query for $\mathcal{H}(Q, T)$
  if the answer is no then
    add the counterexample and its prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

Q	T
$\{\varepsilon, a, aa, aaa\}$	$\{\varepsilon, a\}$

$\mathcal{H}(\{\varepsilon, a\}, \{\varepsilon\})$



Teacher: "No! *aaa* is a counterex."

The L^*-algorithm

$Q = T := \{\varepsilon\}$

repeat

while (Q, T) not closed and consistent

if (Q, T) is not closed enlarge Q

$(\forall q \in Q . \forall a \in A . \exists p \in Q . \; p \sim_T qa)$

if (Q, T) is not consistent enlarge T

$(\forall q, q' \in Q . \forall a \in A . \;\; q \sim_T q' \Rightarrow qa \sim_T q'a)$

ask an equivalence query for $\mathcal{H}(Q, T)$

if the answer is no then

add the counterexample and its prefixes to Q

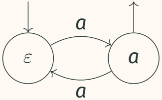
until the answer is yes

return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

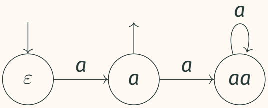
Q	T
$\{\varepsilon, a, aa, aaa\}$	$\{\varepsilon, a\}$

$\mathcal{H}(\{\varepsilon, a\}, \{\varepsilon\})$



Teacher: “No! *aaa* is a counterex.”

$\mathcal{H}(Q, T)$



The L^*-algorithm

$Q = T := \{\varepsilon\}$

repeat

  while (Q, T) not closed and consistent

    if (Q, T) is not closed enlarge Q

      $(\forall q \in Q\,.\,\forall a \in A\,.\,\exists p \in Q\,.\ p \sim_T qa)$

    if (Q, T) is not consistent enlarge T

      $(\forall q, q' \in Q\,.\,\forall a \in A\,.\ \ q \sim_T q' \Rightarrow qa \sim_T q'a)$

  ask an equivalence query for $\mathcal{H}(Q, T)$

  if the answer is no then

    add the counterexample and its prefixes to Q

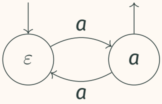
until the answer is yes

return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

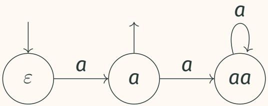
Q	T
$\{\varepsilon, a, aa, aaa\}$	$\{\varepsilon, a\}$

$\mathcal{H}(\{\varepsilon, a\}, \{\varepsilon\})$



Teacher: “No! *aaa* is a counterex.”

$\mathcal{H}(Q, T)$



Teacher: “Yes!”

The L^*-algorithm

$Q = T := \{\varepsilon\}$

repeat

  while (Q, T) not closed and consistent

    if (Q, T) is not closed enlarge Q

      $(\forall q \in Q \,.\, \forall a \in A \,.\, \exists p \in Q \,.\; p \sim_T qa)$

    if (Q, T) is not consistent enlarge T

      $(\forall q, q' \in Q \,.\, \forall a \in A \,.\;\; q \sim_T q' \Rightarrow qa \sim_T q'a)$

  ask an equivalence query for $\mathcal{H}(Q, T)$

  if the answer is no then

    add the counterexample and its prefixes to Q

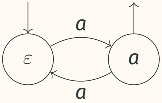
until the answer is yes

return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

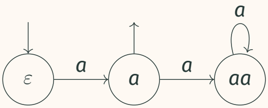
Q	T
$\{\varepsilon, a, aa, aaa\}$	$\{\varepsilon, a\}$

$\mathcal{H}(\{\varepsilon, a\}, \{\varepsilon\})$



Teacher: “No! *aaa* is a counterex.”

$\mathcal{H}(Q, T)$



Teacher: “Yes!”

The L^*-algorithm

$Q = T := \{\varepsilon\}$

repeat

  while (Q, T) not closed and consistent

    if (Q, T) is not closed enlarge Q

      $(\forall q \in Q \,.\, \forall a \in A \,.\, \exists p \in Q \,.\, p \sim_T qa)$

    if (Q, T) is not consistent enlarge T

      $(\forall q, q' \in Q \,.\, \forall a \in A \,.\; q \sim_T q' \Rightarrow qa \sim_T q'a)$

  ask an equivalence query for $\mathcal{H}(Q, T)$

  if the answer is no then

    add the counterexample and its prefixes to Q

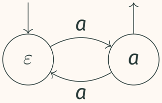
until the answer is yes

return $\mathcal{H}(Q, T)$

Example: Learning the language $\{a\}$ over the alphabet $\{a\}$.

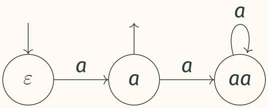
Q	T
$\{\varepsilon, a, aa, aaa\}$	$\{\varepsilon, a\}$

$\mathcal{H}(\{\varepsilon, a\}, \{\varepsilon\})$



Teacher: "No! *aaa* is a counterex."

$\mathcal{H}(Q, T)$



Teacher: "Yes!"

return $\mathcal{H}(Q, T)$

The L^*-algorithm: Variations

The L^*-algorithm has been extended to various other forms of automata

- weighted automata over fields (Bergadano and Varricchio, 1996)
- ~~sub~~sequential transducers (Vilar, 1996)
- nominal automata (van Heerd et al., 2017)
- symbolic automata (Drews et al., 2017)
- non-deterministic automata (Bollig et al., 2009)
- alternating automata (Angluin et al., 2015)

The L^*-algorithm: Variations

The L^*-algorithm has been extended to various other forms of automata

- weighted automata over fields (Bergadano and Varricchio, 1996)
- ~~sub~~sequential transducers (Vilar, 1996)
- nominal automata (van Heerd et al., 2017)
- symbolic automata (Drews et al., 2017)
- non-deterministic automata (Bollig et al., 2009)
- alternating automata (Angluin et al., 2015)

"The need for a unifying framework collecting various types of learning techniques is, thus, beyond all questions." Bollig et al., 2010

The L^*-algorithm: Variations

The L^*-algorithm has been extended to various other forms of automata

- weighted automata over fields (Bergadano and Varricchio, 1996)
- ~~sub~~sequential transducers (Vilar, 1996)
- nominal automata (van Heerd et al., 2017)
- symbolic automata (Drews et al., 2017)
- non-deterministic automata (Bollig et al., 2009)
- alternating automata (Angluin et al., 2015)

"The need for a unifying framework collecting various types of learning techniques is, thus, beyond all questions." Bollig et al., 2010

Other category theoretic generalizations (van Heerd et al., 2017; Urbat and Schröder, 2019)

Back to learning...
automata, not categories!

L^* algorithm categorically??

$Q = T := \{\varepsilon\}$

repeat

  while (Q, T) not closed and consistent

    if (Q, T) is not closed enlarge Q

      $(\forall q \in Q . \forall a \in A . \exists p \in Q . \, p \sim_T qa)$

    if (Q, T) is not consistent enlarge T

      $(\forall q, q' \in Q . \forall a \in A . \; q \sim_T q' \Rightarrow qa \sim_T q'a)$

  ask an equivalence query for $\mathcal{H}(Q, T)$

  if the answer is no then

    add the counterexample and its prefixes to Q

until the answer is yes

return $\mathcal{H}(Q, T)$

L^* algorithm categorically??

$Q = T := \{\varepsilon\}$
repeat
  while (Q, T) not closed and consistent
    if (Q, T) is not closed enlarge Q
      $(\forall q \in Q . \forall a \in A . \exists p \in Q . \; p \sim_T qa)$
    if (Q, T) is not consistent enlarge T
      $(\forall q, q' \in Q . \forall a \in A . \; q \sim_T q' \Rightarrow qa \sim_T q'a)$
  ask an equivalence query for $\mathcal{H}(Q, T)$
  if the answer is no then
    add the counterexample and its
    prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Thomas Colcombet, Daniela Petrisan, Riccardo Stabile: Learning Automata and Transducers: A Categorical Approach. CSL 2021

L^*-revisited

- At the (Q, T) stage of the algorithm the learner only has access to a fragment of the language:

$$L_{Q,T} : QAT \cup QT \rightarrowtail A^* \xrightarrow{L} 2$$

L^*-revisited

- At the (Q, T) stage of the algorithm the learner only has access to a fragment of the language:

$$L_{Q,T} : \; QAT \cup QT \rightarrowtail A^* \xrightarrow{L} 2$$

- This can be represented by a notion of (Q, T)-biautomaton

$$1 \xrightarrow[(q \in Q)]{\triangleright q} Q_1 \underset{\varepsilon}{\overset{a\ (a \in A)}{\rightrightarrows}} Q_2 \xrightarrow[(t \in T)]{t \triangleleft} 2$$

such that the following coherence diagrams commute

$$\begin{array}{ccccc} & & Q_1 & & \\ & \overset{\triangleright q}{\nearrow} & & \overset{a}{\searrow} & \\ 1 & & & & Q_2 \\ & \underset{\triangleright qa}{\searrow} & & \underset{\varepsilon}{\nearrow} & \\ & & Q_1 & & \end{array} \qquad \begin{array}{ccccc} & & Q_2 & & \\ & \overset{a}{\nearrow} & & \overset{t \triangleleft}{\searrow} & \\ Q_1 & & & & 2 \\ & \underset{\varepsilon}{\searrow} & & \underset{at \triangleleft}{\nearrow} & \\ & & Q_2 & & \end{array}$$

Minimal (Q, T)-biautomaton and the hypothesis automaton

We can compute the minimal (Q, T)-biautomaton in an arbitrary category* using off-the-shelf results from (Colcombet,P., 2017).

$$1 \xrightarrow{\triangleright q_{min}} Q/{\sim_{T \cup AT}} \underset{\varepsilon_{min}}{\overset{a_{min}}{\rightrightarrows}} (Q \cup QA)/{\sim_T} \xrightarrow{t\triangleleft_{min}} 2$$

Recall $w \sim_T v$ iff $\forall u \in T. \quad wu \in L \Leftrightarrow vu \in L$

* under mild assumptions

Minimal (Q, T)-biautomaton and the hypothesis automaton

We can compute the minimal (Q, T)-biautomaton in an arbitrary category* using off-the-shelf results from (Colcombet,P., 2017).

$$1 \xrightarrow{\triangleright q_{min}} Q/{\sim_{T \cup AT}} \underset{\varepsilon_{min}}{\overset{a_{min}}{\rightrightarrows}} (Q \cup QA)/{\sim_T} \xrightarrow{t\triangleleft_{min}} 2$$

Recall $w \sim_T v$ iff $\forall u \in T. \quad wu \in L \Leftrightarrow vu \in L$

$\triangleright q_{min}(*) = [q]_{\sim_{T \cup AT}}$

Minimal (Q, T)-biautomaton and the hypothesis automaton

We can compute the minimal (Q, T)-biautomaton in an arbitrary category* using off-the-shelf results from (Colcombet,P., 2017).

$$1 \xrightarrow{\triangleright q_{min}} Q/{\sim_{T \cup AT}} \underset{\varepsilon_{min}}{\overset{a_{min}}{\rightrightarrows}} (Q \cup QA)/{\sim_T} \xrightarrow{t \triangleleft_{min}} 2$$

Recall $w \sim_T v$ iff $\forall u \in T. \quad wu \in L \Leftrightarrow vu \in L$

$$\triangleright q_{min}(*) = [q]_{\sim_{T \cup AT}} \qquad a_{min}([q]_{\sim_{T \cup AT}}) = [qa]_{\sim_T}$$

$$\varepsilon_{min}([q]_{\sim_{T \cup AT}}) = [q]_{\sim_T}$$

Minimal (Q,T)-biautomaton and the hypothesis automaton

We can compute the minimal (Q,T)-biautomaton in an arbitrary category* using off-the-shelf results from (Colcombet,P., 2017).

$$1 \xrightarrow{\triangleright q_{min}} Q/{\sim_{T\cup AT}} \underset{\varepsilon_{min}}{\overset{a_{min}}{\rightrightarrows}} (Q \cup QA)/{\sim_T} \xrightarrow{t\triangleleft_{min}} 2$$

Recall $w \sim_T v$ iff $\forall u \in T.\quad wu \in L \Leftrightarrow vu \in L$

$$\triangleright q_{min}(*) = [q]_{\sim_{T\cup AT}} \qquad a_{min}([q]_{\sim_{T\cup AT}}) = [qa]_{\sim_T} \qquad t\triangleleft_{min}([q]_{\sim_T}) = L_{Q,T}(qt)$$

$$\varepsilon_{min}([q]_{\sim_{T\cup AT}}) = [q]_{\sim_T} \qquad t\triangleleft_{min}([qa]_{\sim_T}) = L_{Q,T}(qat)$$

Minimal (Q, T)-biautomaton and the hypothesis automaton

We can compute the minimal (Q, T)-biautomaton in an arbitrary category* using off-the-shelf results from (Colcombet,P., 2017).

$$1 \xrightarrow{\triangleright q_{min}} Q/{\sim_{T \cup AT}} \underset{\varepsilon_{min}}{\overset{a_{min}}{\rightrightarrows}} (Q \cup QA)/{\sim_T} \xrightarrow{t \triangleleft_{min}} 2$$

Recall $w \sim_T v$ iff $\forall u \in T.\quad wu \in L \Leftrightarrow vu \in L$

$$\triangleright q_{min}(*) = [q]_{\sim_{T \cup AT}} \qquad a_{min}([q]_{\sim_{T \cup AT}}) = [qa]_{\sim_T} \qquad t \triangleleft_{min} ([q]_{\sim_T}) = L_{Q,T}(qt)$$

$$\varepsilon_{min}([q]_{\sim_{T \cup AT}}) = [q]_{\sim_T} \qquad t \triangleleft_{min} ([qa]_{\sim_T}) = L_{Q,T}(qat)$$

Minimal (Q, T)-biautomaton and the hypothesis automaton

We can compute the minimal (Q, T)-biautomaton in an arbitrary category* using off-the-shelf results from (Colcombet,P., 2017).

$$1 \xrightarrow{\triangleright q_{min}} Q/{\sim_{T \cup AT}} \underset{\varepsilon_{min}}{\overset{a_{min}}{\rightrightarrows}} (Q \cup QA)/{\sim_T} \xrightarrow{t \triangleleft_{min}} 2$$

Recall $w \sim_T v$ iff $\forall u \in T. \quad wu \in L \Leftrightarrow vu \in L$

$\triangleright q_{min}(*) = [q]_{\sim_{T \cup AT}}$

$a_{min}([q]_{\sim_{T \cup AT}}) = [qa]_{\sim_T}$

$\varepsilon_{min}([q]_{\sim_{T \cup AT}}) = [q]_{\sim_T}$

$t \triangleleft_{min} ([q]_{\sim_T}) = L_{Q,T}(qt)$

$t \triangleleft_{min} ([qa]_{\sim_T}) = L_{Q,T}(qat)$

- ε_{min} is surjective iff (Q, T) is closed
- ε_{min} is injective iff (Q, T) is consistent

$Q = T := \{\varepsilon\}$
repeat
  while (Q, T) not closed and consistent
    if (Q, T) is not closed enlarge Q
      $(\forall q \in Q . \forall a \in A . \exists p \in Q . \ p \sim_T qa)$
    if (Q, T) is not consistent enlarge T
      $(\forall q, q' \in Q . \forall a \in A . \ q \sim_T q' \Rightarrow qa \sim_T q'a)$
  ask an equivalence query for $\mathcal{H}(Q, T)$
  if the answer is no then
    add the counterexample and its prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Minimal (Q, T)-biautomaton and the hypothesis automaton

We can compute the minimal (Q, T)-biautomaton in an arbitrary category* using off-the-shelf results from (Colcombet,P., 2017).

$$1 \xrightarrow{\triangleright q_{min}} Q/{\sim_{T \cup AT}} \underset{\varepsilon_{min}}{\overset{a_{min}}{\rightrightarrows}} (Q \cup QA)/{\sim_T} \xrightarrow{t\triangleleft_{min}} 2$$

Recall $w \sim_T v$ iff $\forall u \in T. \quad wu \in L \Leftrightarrow vu \in L$

$$\triangleright q_{min}(*) = [q]_{\sim_{T \cup AT}} \qquad a_{min}([q]_{\sim_{T \cup AT}}) = [qa]_{\sim_T} \qquad t \triangleleft_{min} ([q]_{\sim_T}) = L_{Q,T}(qt)$$

$$\varepsilon_{min}([q]_{\sim_{T \cup AT}}) = [q]_{\sim_T} \qquad t \triangleleft_{min} ([qa]_{\sim_T}) = L_{Q,T}(qat)$$

- ε_{min} is surjective iff (Q, T) is closed
- ε_{min} is injective iff (Q, T) is consistent
- If ε_{min} is an isomorphism we merge the two states of the (Q, T)-biautomaton and obtain $\mathcal{H}(Q, T)$.

```
Q = T := {ε}
repeat
  while (Q,T) not closed and consistent
    if (Q,T) is not closed enlarge Q
       (∀q ∈ Q.∀a ∈ A.∃p ∈ Q. p ~T qa)
    if (Q,T) is not consistent enlarge T
       (∀q,q' ∈ Q.∀a ∈ A. q ~T q' ⇒ qa ~T q'a)
  ask an equivalence query for H(Q,T)
  if the answer is no then
    add the counterexample and its
    prefixes to Q
until the answer is yes
return H(Q,T)
```

The $FunL^*$-algorithm

The $FunL^*$-algorithm

input: teacher of the target language L — in a catgeory $\mathcal{C}$
output: $Min(L)$
$Q := T := \{\varepsilon\}$
repeat
  while ε_{min} is not an isomorphism **do** — $\mathrm{Iso} = E \cap M$
    if $\varepsilon_{min} \notin E$ **then** — (E, M) fact. system
      add QA to Q
    if $\varepsilon_{min} \notin M$ **then**
      add AT to T
  ask an equivalence query for the hypothesis automaton $\mathcal{H}(Q, T)$
  if the answer is no **then**
    add the counterexample and all its prefixes to Q
until the answer is yes
return $\mathcal{H}(Q, T)$

Correction and termination of the algorithm

Theorem. Assume $\mathcal{C}$ is a category with a factorization system (E, M), having countable copowers and countable powers.

We consider a target language L in the catgeory $\mathcal{C}$ such that the state space of the minimal automaton for L is (E, M)-noetherian* (generalization of finite).

Then the $FunL^*$-algorithm terminates, eventually producing the minimal automaton $Min(L)$ accepting L.

*(E, M)-noetherianity means no infinite chains of E-quotients or of M-subobjects.

Perspectives

Minimization/learning for free ?

What is special about certain monads? And why it works in some cases and not in others?

Minimization/learning for free ?

What is special about certain monads? And why it works in some cases and not in others?

Are there some conditions on a monad T so that $\mathrm{Kl}(T)$ has all the required properties required for the existence of minimization/learning of $\mathrm{Kl}(T)$-automata ?

Minimization/learning for free ?

What is special about certain monads? And why it works in some cases and not in others?

Are there some conditions on a monad T so that $\mathrm{Kl}(T)$ has all the required properties required for the existence of minimization/learning of $\mathrm{Kl}(T)$-automata ?

Some advancement in this direction
Quentin Aristote:
Active Learning of Deterministic Transducers with Outputs in Arbitrary Monoids. CSL 2024.

Minimization/learning for free ?

What is special about certain monads? And why it works in some cases and not in others?

Are there some conditions on a monad T so that $\mathrm{Kl}(T)$ has all the required properties required for the existence of minimization/learning of $\mathrm{Kl}(T)$-automata ?

Some advancement in this direction
Quentin Aristote:
Active Learning of Deterministic Transducers with Outputs in Arbitrary Monoids. CSL 2024.

And what can be done when we know that $\mathrm{Kl}(T)$ is not good enough?

Minimizing NFAs

The category Rel $\simeq$ Kl($\mathcal{P}$) does not have a good factorization system.

Minimizing NFAs

The category Rel $\simeq$ Kl($\mathcal{P}$) does not have a good factorization system.

Move to Eilenberg-Moore algebras, that is, to the category of join sup-semilattices !

We see a Rel-valued automaton as a JSL-valued automaton.

Minimizing NFAs

The category Rel $\simeq$ Kl($\mathcal{P}$) does not have a good factorization system.

Move to Eilenberg-Moore algebras, that is, to the category of join sup-semilattices !

We see a Rel-valued automaton as a JSL-valued automaton.

In JSL we have all the three ingredients for minimization ! Do we retrieve a JSL-automaton that comes from an actual NFA?

Minimizing NFAs

The category Rel $\simeq$ Kl($\mathcal{P}$) does not have a good factorization system.

Move to Eilenberg-Moore algebras, that is, to the category of join sup-semilattices !

We see a Rel-valued automaton as a JSL-valued automaton.

In JSL we have all the three ingredients for minimization ! Do we retrieve a JSL-automaton that comes from an actual NFA?

It turns out yes! The canonical RFSA introduced in
François Denis, Aurélien Lemay, Alain Terlutte:
Residual Finite State Automata. Fundam. Informaticae 51(4): 339-368 (2002)

Minimizing NFAs

The category Rel $\simeq$ Kl($\mathcal{P}$) does not have a good factorization system.

Move to Eilenberg-Moore algebras, that is, to the category of join sup-semilattices !

We see a Rel-valued automaton as a JSL-valued automaton.

In JSL we have all the three ingredients for minimization ! Do we retrieve a JSL-automaton that comes from an actual NFA?

It turns out yes! The canonical RFSA introduced in
François Denis, Aurélien Lemay, Alain Terlutte:
Residual Finite State Automata. Fundam. Informaticae 51(4): 339-368 (2002)

Understand this through a lax functor from JSL to Rel... Ongoing work with Quentin Schroeder and Quentin Aristote.

Further extensions

- Extension to tree automata
- Weighted automata over semirings …
- What about other forms of learning, e.g., nominal automata? We can build on Victor Iwaniack's work on automata in toposes.

And even more importantly ...

And even more importantly ...

TCS4F
THEORETICAL
COMPUTER
SCIENTISTS
FOR FUTURE

https://tcs4f.org/

What we can do : keep a CO_2 budget, choose more sustainable means of transport, spread the word, sign the TCS4F manifesto...

And even more importantly ...

TCS4F
THEORETICAL
COMPUTER
SCIENTISTS
FOR FUTURE

https://tcs4f.org/

What we can do : keep a CO_2 budget, choose more sustainable means of transport, spread the word, sign the TCS4F manifesto...

An estimation of the emissions per person for a return trip Paris - Barcelone

- by train : approx. 6 kg CO_2
- by plane : approx. 680 kg CO_2