Model Learning

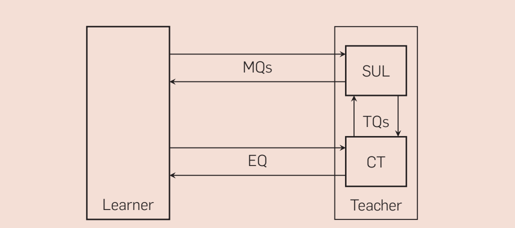
Deepak D'Souza

Department of Computer Science and Automation
Indian Institute of Science, Bangalore.

05 Feb 2020.

Model Learning Problem

Given a (possibly black-box) implementation T ("System Under Learning"), learn a finite-state machine M that "conforms" to T. Learner is allowed to use membership and equivalence queries.



Fritz Vaandrager, CACM 2017

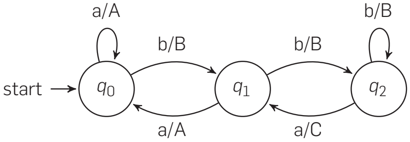
(Mealy) State Machines

$M = (I, O, Q, q_o, \delta, \lambda)$ where

- I, O are finite input and output alphabets
- Q a finite set of states
- $\delta : (Q \times I) \rightarrow Q$ is transition function.
- $\lambda : (Q \times I) \rightarrow O$ is the state output function.

Language of machine M, A_M, is a map from I^* to O^* (same-length string transducer).

Example with input alphabet $= \{a, b\}$, output alphabet $= \{A, B\}$.



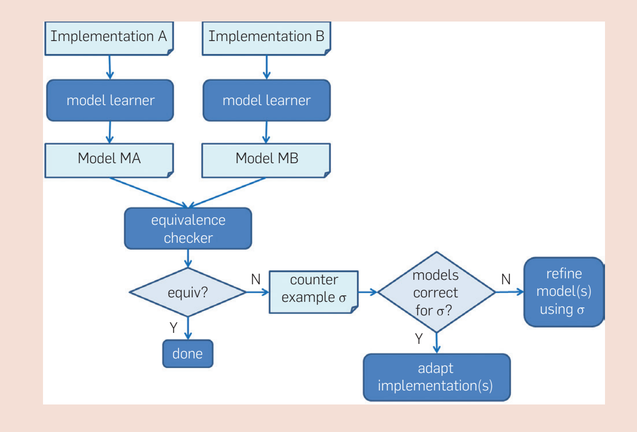
State Machine as a Program

```
enum StateType = {....};
StateType state;

state = ... // initialize

while (true) {
  read(a); // read input
  switch (state) {
    0: switch(a) {
         a: output(...); state = ...;
         b: output(...); state = ...;
       }
    1: switch(a) {
         a: output(...); state = ...;
         b: output(...); state = ...;
       }
    ...
  }
}
```

Applications: Refactoring Equivalence



Fritz Vaandrager, CACM 2017

Other Applications

- Checking for unxepected interactions allowed by Transport Layer Security (TLS) protocol between Client and Server. Learn model for both client and server, and model-check for unwanted interactions.
- Client-Server TCP protocol, unwanted interactions allowed.
- Reverse-Engineer a smart card reader, and model-check for security issues.

Other Applications

- Checking for unxepected interactions allowed by Transport Layer Security (TLS) protocol between Client and Server. Learn model for both client and server, and model-check for unwanted interactions.
- Client-Server TCP protocol, unwanted interactions allowed.
- Reverse-Engineer a smart card reader, and model-check for security issues.

Similarities with PBE and CEGIS frameworks.

Angluin's L^* Algorithm

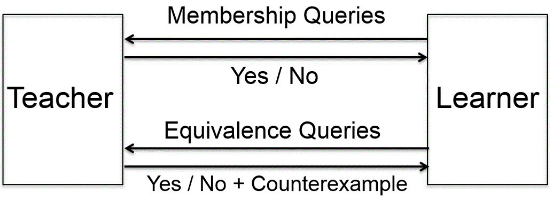
Basic approach is to use Dana Angluin's L^* algorithm to efficiently learn an FSM.

Learning Regular Sets from Queries and Counterexamples, in *Information & Computation*, 1987.

Angluin's L^* algorithm

Teacher has a regular language U in mind.
The Learner can ask two types of queries:

- Is a given string w in U? Teacher answers "Yes" or "No".
- Does a given DFA $\mathcal{A}$ accept the language U? Teacher answers "Yes" or gives a counterexample x.



Angluin's algorithm for the Learner finds the canonical DFA for U, in a number of steps polynomial in the number of states of the canonical DFA for U and the length of the longest counterexample returned by the teacher.

Angluin's Algorithm by Example

Suppose the Teacher has in mind the language

$U = \{w \in \{a, b\}^* \mid$ number of a's is even and number of b's is even$\}$

The Learner asks the Teacher if ϵ, a, and b belong to U, and obtains the following Observation Table:

		ϵ
S	ϵ	1
$S.\{a, b\}$	a	0
	b	0

The set of strings S represents the states of the automaton constructed by the Learner.
Entry (s, e) of the table represents the fact that from state s the automaton accepts/rejects the string e.

Angluin's Algorithm by Example

Suppose the Teacher has in mind the language

$U = \{w \in \{a, b\}^* \mid$ number of a's is even and number of b's is even$\}$

The Learner asks the Teacher if ϵ, a, and b belong to U, and obtains the following Observation Table:

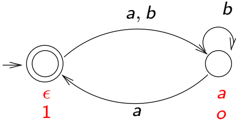
		ϵ
S	ϵ	1
$S.\{a, b\}$	a	0
	b	0

The set of strings S represents the states of the automaton constructed by the Learner.
Entry (s, e) of the table represents the fact that from state s the automaton accepts/rejects the string e.
This table is not "closed" as there are no states (or "rows") corresponding to $\epsilon \cdot a$ and $\epsilon \cdot b$.

Angluin's Algorithm by Example: 2

Learner closes table by adding string a to S, and asking membership queries for aa and ab.
He now gets the observation table:

		ϵ
S	ϵ	1
	a	0
$S.\{a, b\}$	b	0
	aa	1
	ab	0



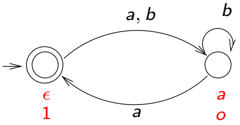
$\mathcal{A}_1$

This table is closed and consistent, and represents the DFA $\mathcal{A}_1$.

Angluin's Algorithm by Example: 2

Learner closes table by adding string a to S, and asking membership queries for aa and ab.
He now gets the observation table:

		ϵ
S	ϵ	1
	a	0
$S.\{a,b\}$	b	0
	aa	1
	ab	0



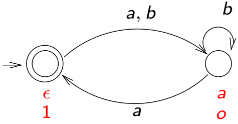
$\mathcal{A}_1$

This table is closed and consistent, and represents the DFA $\mathcal{A}_1$.
Learner now asks the Teacher if $\mathcal{A}_1$ represents the language she has in mind.

Angluin's Algorithm by Example: 2

Learner closes table by adding string *a* to *S*, and asking membership queries for *aa* and *ab*.
He now gets the observation table:

		ϵ
S	ϵ	1
	a	0
$S.\{a, b\}$	b	0
	aa	1
	ab	0



$\mathcal{A}_1$

This table is closed and consistent, and represents the DFA $\mathcal{A}_1$.
Learner now asks the Teacher if $\mathcal{A}_1$ represents the language she has in mind. Teacher replies with counterexample *bb* which is in U but is not accepted by $\mathcal{A}_1$.

Angluin's Algorithm by Example: 3

Learner adds *bb* and its prefixes to his set S, makes membership queries for *ba*, *bba*, and *bbb* to obtain the observation table:

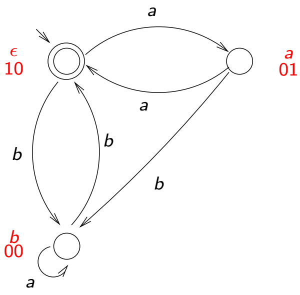
		ϵ
S	ϵ	1
	a	0
	b	0
	bb	1
$S.\{a,b\}$	aa	1
	ab	0
	ba	0
	bba	0
	bbb	0

This table is closed but not consistent. The rows for *a* and *b* are identical, but *aa* and *ba* have different rows.

Angluin's Algorithm by Example: 4

Learner adds $\epsilon \cdot a$ (that is, a) and its suffixes to the set E, and makes membership queries to obtain the observation table:

		ϵ	a
S	ϵ	1	0
	a	0	1
	b	0	0
	bb	1	0
$S.\{a, b\}$	aa	1	0
	ab	0	0
	ba	0	0
	bba	0	1
	bbb	0	0



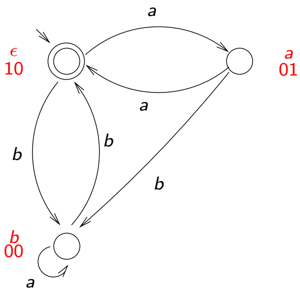
$\mathcal{A}_2$

This table is closed and consistent. So Learner conjectures the automaton $\mathcal{A}_2$.

Angluin's Algorithm by Example: 4

Learner adds $\epsilon \cdot a$ (that is, a) and its suffixes to the set E, and makes membership queries to obtain the observation table:

		ϵ	a
S	ϵ	1	0
	a	0	1
	b	0	0
	bb	1	0
$S.\{a,b\}$	aa	1	0
	ab	0	0
	ba	0	0
	bba	0	1
	bbb	0	0



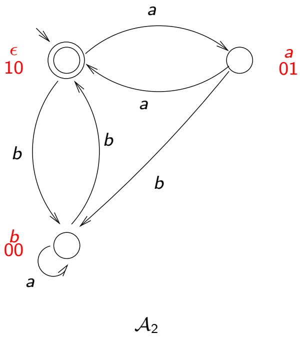
$\mathcal{A}_2$

This table is closed and consistent. So Learner conjectures the automaton $\mathcal{A}_2$. Teacher responds with counterexample abb.

Angluin's Algorithm by Example: 5

Learner adds *abb* its prefixes to S, makes membership queries to obtain the observation table:

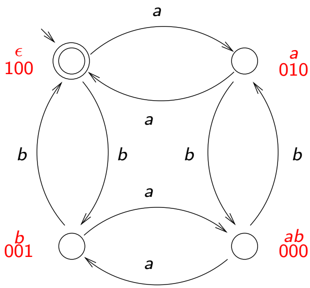
		ϵ	a
S	ϵ	1	0
	a	0	1
	b	0	0
	ab	0	0
	bb	1	0
	abb	0	1
$S.\{a,b\}$	aa	1	0
	ba	0	0
	aba	0	0
	bba	0	1
	bbb	0	0
	$abba$	1	0
	$abbb$	0	0



Angluin's Algorithm by Example: 6

Learner adds b and its suffixes to E, and makes membership queries to obtain the observation table:

		ϵ	a	b
	ϵ	1	0	0
S	a	0	1	0
	b	0	0	1
	ab	0	0	0
	bb	1	0	0
	abb	0	1	0
	aa	1	0	0
$S.\{a,b\}$	ba	0	0	0
	aba	0	0	1
	bba	0	1	0
	bbb	0	0	1
	$abba$	1	0	0
	$abbb$	0	0	1



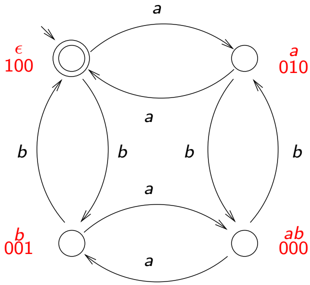
$\mathcal{A}_3$

Table is closed and consistent, so Learner conjectures DFA $\mathcal{A}_3$.

Angluin's Algorithm by Example: 6

Learner adds b and its suffixes to E, and makes membership queries to obtain the observation table:

		ϵ	a	b
S	ϵ	1	0	0
	a	0	1	0
	b	0	0	1
	ab	0	0	0
	bb	1	0	0
	abb	0	1	0
$S.\{a,b\}$	aa	1	0	0
	ba	0	0	0
	aba	0	0	1
	bba	0	1	0
	bbb	0	0	1
	$abba$	1	0	0
	$abbb$	0	0	1



$\mathcal{A}_3$

Table is closed and consistent, so Learner conjectures DFA $\mathcal{A}_3$. Teacher responds with "Yes!".

Angluin's L* Algorithm

Initialize S and E to $\{\lambda\}$.
Ask membership queries for λ and each $a \in A$.
Construct the initial observation table (S, E, T).

Repeat:
  While (S, E, T) is not closed or not consistent:
    If (S, E, T) is not consistent,
      then find s_1 and s_2 in S, $a \in A$, and $e \in E$ such that
      $row(s_1) = row(s_2)$ and $T(s_1 \cdot a \cdot e) \neq T(s_2 \cdot a \cdot e)$,
      add $a \cdot e$ to E,
      and extend T to $(S \cup S \cdot A) \cdot E$ using membership queries.
    If (S, E, T) is not closed,
      then find $s_1 \in S$ and $a \in A$ such that
      $row(s_1 \cdot a)$ is different from $row(s)$ for all $s \in S$,
      add $s_1 \cdot a$ to S,
      and extend T to $(S \cup S \cdot A) \cdot E$ using membership queries.

  Once (S, E, T) is closed and consistent, let $M = M(S, E, T)$.
  Make the conjecture M.
  If the Teacher replies with a counter-example t, then
    add t and all its prefixes to S
    and extend T to $(S \cup S \cdot A) \cdot E$ using membership queries.
Until the Teacher replies *yes* to the conjecture M.
Halt and output M.

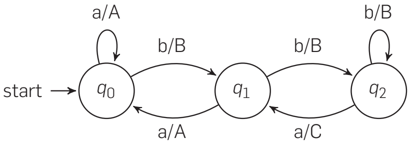
FIG. 1. The Learner L^*.

Complexity of Angluin's Algorithm

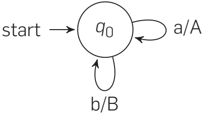
Let minimal automaton for U have n states, and length of longest counter-example given by Teacher be m. Then:

- Makes at most n equivalence conjectures. (Since counterexample increases number of states by at least 1.)
- Total number of strings in E is at most n.
- Table can be represented by table of size $O(m^2n^2 + mn^3)$.
- At most n closed/consistency checks, each done in poly time in size of table.
- Total running time is polynomial in n and m.

Extending to Mealy State Machines



$\mathcal{O}_1$	a	b
ϵ	A	B
a	A	B
b	A	B



$\mathcal{O}_2$	a	b
ϵ	A	B
b	A	B
bb	C	B
bba	A	B
a	A	B
ba	A	B
bbb	C	B
bbaa	A	B
bbab	C	B

$\mathcal{O}_3$	a	b	ba
ϵ	A	B	A
b	A	B	C
bb	C	B	C
bba	A	B	C
a	A	B	A
ba	A	B	A
bbb	C	B	C
bbaa	A	B	A
bbab	C	B	C