Angluin's Algorithm For Learning Regular Languages

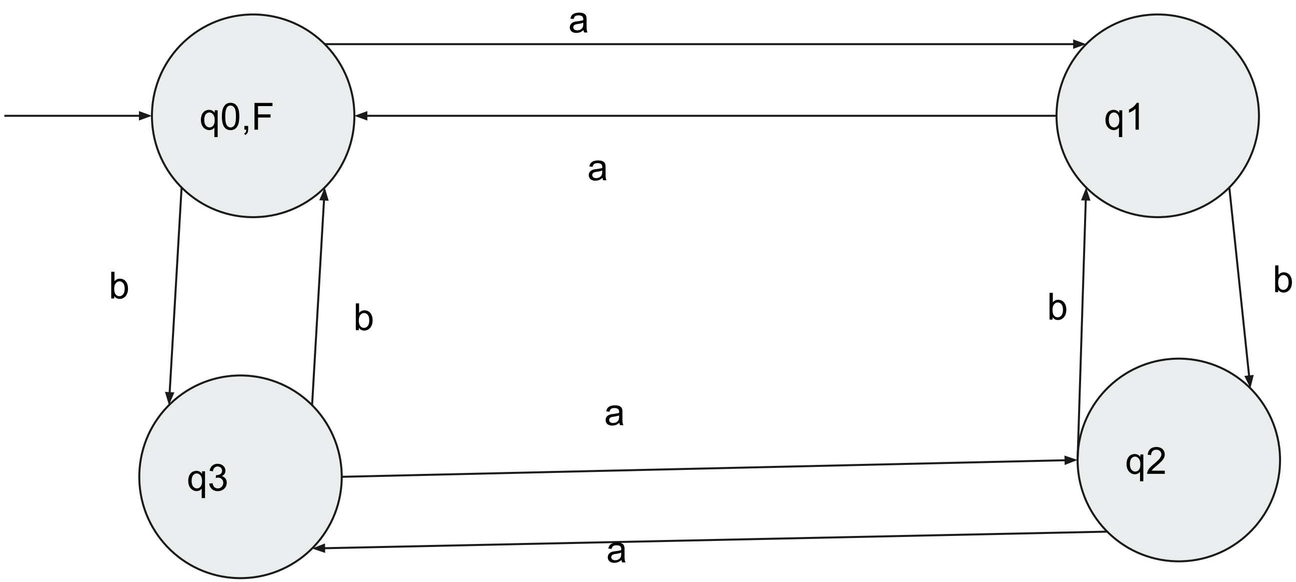
Presented By:
Addrija
Mehul Shankhapal
Dhruv Piyush Shah

Content

1) Motivation
2) Introduction
3) Terms
4) Examples
5) Running Example of Algorithm
6) Conclusion & Drawbacks
7) References

Motivation

Construct a DFA for language of strings over {a,b} such that the string has even number of a's and even number of b's.



Motivation (Points to Ponder on)

1) How do we construct a DFA for a given Regular Language L?
2) Do we use an algorithm in constructing the DFA for a Regular Language?
3) How many of us use smart tricks or intelligence in constructing a DFA ?
4) Have you encountered with a DFA problem that was tough to crack in an exam?

Introduction

Solution for above problem is:

Angluin’s (L*) Algorithm (By Dana Angluin):

- Learning Algorithm that uses Queries And Counterexamples in learning automaton.
- It’s a two party algorithm involving a Teacher and a Learner.
- * in L* stands for regular languages.

Terms

- Teacher
  - Teacher knows the regular set corresponding to the language.
  - He has a DFA in his mind.
- Learner
  - Learner wants to make a DFA for the given regular language.
  - It takes help from the Teacher to construct the DFA who already know it by asking only two type of queries / question.
- Minimally Adequate Teacher
  - This is no different from the above definition of the Teacher.
  - It is just called minimally adequate because teacher can answer only two type of questions.
- Membership Queries
  - This query consists of a string w.
  - Teacher reply with “YES” if w belongs to the regular set and “NO” otherwise.

Terms

- Equivalence Queries
  - It consists of a description of a regular set, i.e. DFA made by the learner.
  - Teacher reply with “YES” if given DFA accepts the regular set. If doesn’t, then teacher reply with the counterexample in form of a string.
- Prefix Closed Set
  - Any set S is prefix-closed if for every w ∈ S, prefix(w) ∈ S.
- Suffix Closed Set
  - Any set S is suffix-closed if for every w ∈ S, suffix(w) ∈ S

Examples on Prefix Closed Set

1.	{ 0, λ, 10, 010 }	X
2.	{ 110, 1, 0, λ, 11 }	✔
3.	{ 1110, 10, 1 }	X
4.	{ 011, 0, λ, 11, 01 }	X
5.	{ 111, λ, 11, 1, 0 }	✔

Examples on Suffix Closed Set

1. { 0, λ, 10, 010 } ✔
2. { 110, 1, 0, λ, 11 } X
3. { 1110, 10, 1 } X
4. { 011, 0, λ, 11, 01 } X
5. { 111, λ, 11, 1, 0 } ✔

Notations

U: Unknown language to be learnt

A: Alphabet Set

S: Non-empty finite prefix-closed set of strings

E: Non-empty finite suffix-closed set of strings

T: Mapping from finite set of strings to {0, 1}

$T(w) = 1$, iff $w \in U$

Row(s): Row of table corresponding to s.

OBSERVATION TABLE(Hankel Matrix)

Hf(p,s)	λ	a	b	aa	ab	ba	bb
λ	Hf(λ,λ)	Hf(λ,a)	Hf(λ,b)	Hf(λ,aa)	Hf(λ,ab)	Hf(λ,ba)	Hf(λ,bb)
a	Hf(a,λ)	Hf(a,a)	Hf(a,b)	Hf(a,aa)	Hf(a,ab)	Hf(a,ba)	Hf(a,bb)
b	Hf(b,λ)	Hf(b,a)	Hf(b,b)	Hf(b,aa)	Hf(b,ab)	Hf(b,ba)	Hf(b,bb)
aa	Hf(aa,λ)	Hf(aa,a)	Hf(aa,b)	Hf(aa,aa)	Hf(aa,ab)	Hf(aa,ab)	Hf(aa,bb)
ab	Hf(ab,λ)	Hf(ab,a)	Hf(ab,b)	Hf(ab,aa)	Hf(ab,ab)	Hf(ab,ba)	Hf(ab,bb)
ba	Hf(ba,λ)	Hf(ba,a)	Hf(ba,b)	Hf(ba,aa)	Hf(ba,ab)	Hf(ba,ba)	Hf(ba,bb)
bb	Hf(bb,λ)	Hf(bb,a)	Hf(bb,b)	Hf(bb,aa)	Hf(bb,ab)	Hf(bb,ba)	Hf(bb,bb)

OBSERVATION TABLE(Hankel Matrix)

Hankel Matrix(H):

1) The index of row are called Prefixes
2) The index of columns are called Suffixes
3) Constraints that Hankel matrix follow:
   a) If p.s=p'.s' then Hf(p,s)=Hf(p',s')
4) Hf(p,s)=f(p.s) (f : (sigma)*->R)(In case of L* f(p.s)is T(p.s))

One can represent DFA in form of a Hankel Matrix where the f(p.s) takes value either 0 or 1 if p.s belongs to Language L then entry is 1 otherwise 0.

Angluin's L * Algorithm converts Observation Table(Closed and Consistent Finite Hankel Matrix) to a DFA.

OBSERVATION TABLE

1) The learner learns DFA by using the Observation Table and the entries in the observation table which are filled by asking membership queries to the teacher.

2) The **columns** of the table is **the suffix-closed set E** and the **row** is prefix closed set **S union S.a(for all a in A)**

3) row(s) denotes a state in the DFA that will be obtained while converting observation table to DFA.

4) The observation table in L * algorithm has rows indexed by S and S.a for each a in A(The rows are divided into two parts)

L * Algorithm

Two Crucial Properties of Observation table:

1) **Closed**: An observation table is called closed if for all t in S.A there exist an s' in S such that row(s')=row(t).This states that every row(s.a) must be present in row(s) .

2) **Consistent**: An observation table is said to be consistent if, whenever s1,s2 in S satisfy row(s1)=row(s2) then for every a in A it must satisfy row(s1.a)=row(s2.a).

Exercise

1) E={λ,a} S={λ,a,b} Is this table closed?

Not closed as no s exists such that row(S)=row(ab)

	λ	a
λ	0	1
a	1	1
b	0	0

	λ	a
aa	0	0
ab	1	0
ba	0	1
bb	1	1

Exercise

2) E={λ,a} S={λ,a} Is this table closed?

Yes , it is closed

	λ	a
λ	0	0
a	1	1

	λ	a
b	1	1
aa	0	0
ab	0	0

Exercise

3) E={λ,a} S={λ,a,b} Is this table consistent?

Not consistent because row(λ.a)!= row(a.a)

	λ	a
λ	0	1
a	0	1
b	0	0

	λ	a
aa	0	0
ab	0	1
ba	1	1
bb	1	0

Exercise

4) E={λ,a} S={λ,a} Is this table consistent?

Yes, it is consistent..

	λ	a
λ	0	0
a	0	0

	λ	a
b	0	1
aa	0	0
ab	0	1

L * Algorithm

The Algorithm starts with S={λ} E={λ} And then constructs observation table in each iteration and checks for its closure and consistency property and takes appropriate action to make the table both closed and consistent and fires an equivalence query to the teacher.

Based on the teacher's response appropriate action is taken.

We will illustrate the working of the algorithm by taking an example run

L * Algorithm on second last symbol is b in string

E={λ} S={λ} A={a,b}

T1	λ
λ	

T1	λ
a	
b	

T1 Observation Table

T1 table entry filled based on membership queries answers..

T1	λ
λ	0

T1	λ
a	0
b	0

Is table closed?

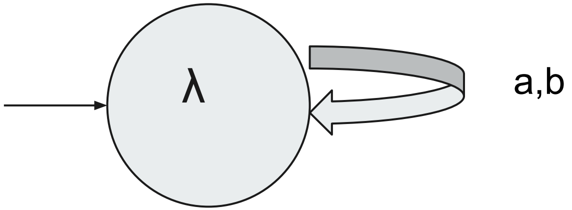
Is table consistent?

T1 observation table

1) The table T1 is both consistent and closed so now we make DFA out of the observation table.
2) Procedure to make DFA D(Q,A,Q0,delta,F)from observation table M(S,E,T).
   a) Q={row(s) | s belongs to S}
   b) A={a,b}
   c) Q0=row(λ)
   d) Transition function is : δ(row(s),a)=row(s.a)
   e) F={row(s) | s belongs to S and T(s)=1}

DFA of T1

DFA given to teacher in equivalence query..



The teacher says No and gives “ba” as counterexample .

Now the learner appends S by ba and its prefixes and then asks membership queries to the teacher.

T2 after asking membership queries

T2:

T2	λ
λ	0
b	0
ba	1

a	0
bb	1
baa	0
bab	0

Is the table closed?

Is the table consistent?

T3 observation table

1) T2 is closed but not consistent because row(λ)=row(b) but row(a)!=row(ba).
2) So we add a to the E set.

Is T3 closed?

T3	λ	a
λ	0	0
b	0	1
ba	1	0
a	0	0
bb	1	1
baa	0	0
bab	0	1

T4 Observation Table

1) T3 is not closed as there is no row s in S such that row(bb)=row(s).So we add bb and its prefix in set S
2) Is the table closed?
3) Is the table consistent?

T4	λ	a
λ	0	0
b	0	1
ba	1	0
bb	1	1

T4	λ	a
a	0	0
baa	0	0
bab	0	1
bba	1	0
bbb	1	1

The table is both closed and consistent

T4 Observation Table

1) T3 is not closed as there is no row s in S such that row(bb)=row(s).So we add bb and its prefix in set S
2) Is the table closed?
3) Is the table consistent?

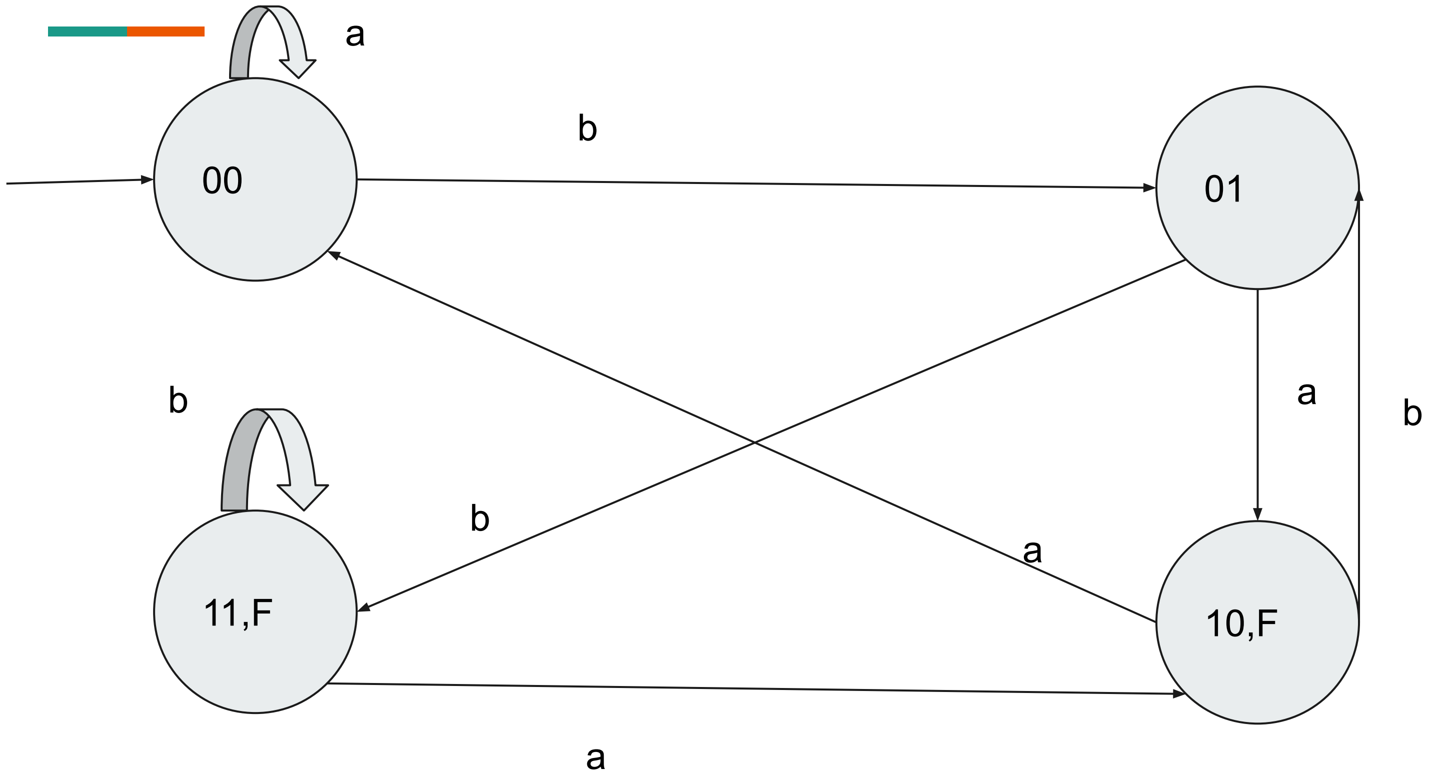
	T4	λ	a
Q0	λ	0	0
Q1	b	0	1
Q2	ba	1	0
Q3	bb	1	1

T4	λ	a
a	0	0
baa	0	0
bab	0	1
bba	1	0
bbb	1	1

The table is both closed and consistent

T4 observation table

DFA of T4 The learner asks equivalence query for the below DFA



The teacher replies yes.

L * Algorithm Pseudocode

Initialize S and E to {λ}

Ask membership queries for λ and a , for all 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

find s1,s2 belonging to S , a belonging to A, e belonging to E ,such that row(s1)=row(s2)

and T(s1.a.e)!=T(s2.a.e),

add (a.e) to E

extend T to (S U S.A).E using membership queries

L * Algorithm Pseudocode Continued

If (S,E,T) is not closed

find s1 belonging to S, a belonging to A such that row(s1.a)!=row(s) for all s belonging to S

add s1.a to S

extend T to (S U S.A).E using membership queries

Perform an equivalence query with M =M(S,E,T)

If answer is “no” with counterexample t

add t and it’s prefixes to S

extend T to (S U S.A).E using membership queries

Until answer is “yes" from equivalence query

Conclusion

- Angluin's automata learning framework
- Learns the smallest automaton that satisfies given constraints.
- Does it efficiently in polynomial time in smallest automaton that satisfies the given constraints.

Drawbacks ?

Conclusion

- Angluin's automata learning framework
- Learns the smallest automaton that satisfies given constraints.
- Does it efficiently in polynomial time in smallest automaton that satisfies the given constraints.

Drawbacks ?

- It is a two party algorithm, which means it needs a teacher who knows the regular set corresponding to the language and a DFA for it.

References

1) https://www.csa.iisc.ac.in/~deepakd/atc-2015/L_Star_Algo.pdf
2) Insights to Angluin's Learning - ScienceDirect
3) Angluin's Algorithm for Learning DFAs | Ullas Aparanji | CSAUSS17 - YouTube
4) https://events.csa.iisc.ac.in/summerschool2017/wp-content/uploads/...
5) Learning regular sets from queries and counterexamples - ScienceDirect
6) Learning Automata with Hankel Matrices - Borja Balle, Amazon Research Cambridge - YouTube
7) Exact query learning of regular and context-free grammars. (logic-data-science.github.io)

THANK YOU !