Learning with queries

About L* algorithm

Patryk Flama

Department of Computer Science
University of Wrocław

7 kwietnia 2025

Overview

Minimally Adequate Teacher

Minimally Adequate Teacher (MAT) is an Oracle answering two types of queries:

- membership queries (MQ)
- equivalence queries (EQ)

$\mathsf{QUERY} = \{\mathsf{MQ}, \mathsf{EQ}\}$

General idea

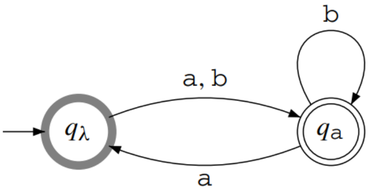
- find a consistent observation table
- construct DFA and submit equivalence query to the Oracle
- use the counter-example to update the table
- submit membership queries to make the table closed and complete
- repeat until Oracle tells us that that the correct language has been reached

Observation table

An observation table is a specific tabular representation of an automaton

	λ	a
λ	0	1
a	1	0
b	1	0
aa	0	1
ab	1	0

(a) An observation table.



(b) The corresponding automaton.

Fig. 13.1. The observation table and the corresponding automaton.

Observation table

Definition

An observation table is a triple $\langle \text{STA, EXP, OT} \rangle$:

- $\text{STA} = \text{Red} \cup \text{Blue}$
- $\text{Red} \subset \Sigma^*$
- $\text{EXP} \subset \Sigma^*$
- $\text{Blue} = \text{Red} \cdot \Sigma \backslash \text{Red}$
- $\text{OT} : \text{STA} \times \text{EXP} \rightarrow \{0, 1, *\}$:

$$\text{OT}[r][c] = \begin{cases} 1 & \text{if } rc \in L \\ 2 & \text{if } rc \notin L \\ * & \text{otherwise} \end{cases}$$

	λ	a
λ	0	1
a	1	0
b	1	0
aa	0	1
ab	1	0

(a) An observation table.

Holes and completeness

Holes

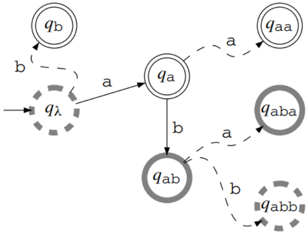
A hole in a table $\langle \text{STA}, \text{EXP}, \text{OT} \rangle$is a pair (r, c) such that $\text{OT}[r][c] = *$

Completeness

A table is complete if it has no **Holes**

	λ	a	b
λ		1	1
a	1	1	0
ab	0	0	
b	1		1
aa	1	1	
aba	0	1	0
abb			1

(a) An incomplete table.



(b) The corresponding automaton.

Fig. 13.2. The automaton corresponding to an incomplete table.

Closed table

We consider here the case where there are no holes in the table

Equivalent prefixes/rows

Two prefixes r and r' are equivalent if $\mathrm{OT}[r] = OT[r']$
We will denote this by $r \equiv_{\mathrm{EXP}} r'$

Closed table

A table $\langle \mathrm{STA}, \mathrm{EXP}, \mathrm{OT} \rangle$is **closed** if given any row r of *Blue* there is some row r' in *Red* such that $r \equiv_{\mathrm{EXP}} r'$

Closed table

It's easy to check if table is closed. But what can the algorithm do if it's not?
Let r be the row from *Blue* that does not appear in *Red*. Add r to *Red* and $\forall a \in \Sigma$ add ra to *Blue*
We can repeat this until the table is **closed** (notice that the number of iterations is bounded by the size of the automaton)

Closed table

Examples

Row *ab* does not appear in *Red*, so we add it there. Then for all $a \in \Sigma$ (a, b) we add new strings to *Blue*. Additionally, all known observations are filled.

	λ	a
λ	0	1
a	1	0
b	1	0
aa	0	1
ab	1	1

(a) A table that is not closed, because of row ab.

	λ	a
λ	0	1
a	1	0
ab	1	1
b	1	0
aa	0	1
aba	1	
abb		

(b) Closing the table.

Fig. 13.4. Closing a table.

Build DFA from table

We can build DFA from observation table, if:

- The set of strings marking the states in STA must be prefix-closed
- The set EXP is suffix-closed
- The table must be complete (have no holes)
- The table must be closed (no 'unique' blue rows)

Build DFA from table

Algorithm 13.1: LSTAR-BUILDAUTOMATON.

Input: a closed and complete observation table (STA, EXP, OT)
Output: DFA $\langle \Sigma, Q, q_\lambda, \mathbb{F}_\mathbb{A}, \mathbb{F}_\mathbb{R}, \delta \rangle$
$Q \leftarrow \{q_u : u \in \textsc{Red} \wedge \forall v < u \ \mathrm{OT}[v] \neq \mathrm{OT}[u]\}$;
$\mathbb{F}_\mathbb{A} \leftarrow \{q_u \in Q : \mathrm{OT}[u][\lambda] = 1\}$;
$\mathbb{F}_\mathbb{R} \leftarrow \{q_u \in Q : \mathrm{OT}[u][\lambda] = 0\}$;
for $q_u \in Q$ **do**
| **for** $a \in \Sigma$ **do** $\delta(q_u, a) \leftarrow q_w \in Q : \mathrm{OT}[ua] = \mathrm{OT}[w]$
end
return $\langle \Sigma, Q, q_\lambda, \mathbb{F}_\mathbb{A}, \mathbb{F}_\mathbb{R}, \delta \rangle$

Example

After applying construction from algorithm on given table, we obtain $Q = \{q_\lambda, q_a\}, \mathbb{F}_\mathbb{A} = q_a, \mathbb{F}\mathbb{R} = q_\lambda$ and δ given by the transition table

---

Algorithm 13.1: LSTAR-BUILDAUTOMATON.

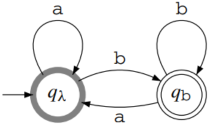
Input: a closed and complete observation table (STA, EXP, OT)
Output: DFA $\langle \Sigma, Q, q_\lambda, \mathbb{F}_\mathbb{A}, \mathbb{F}_\mathbb{R}, \delta \rangle$
$Q \leftarrow \{q_u : u \in \text{RED} \wedge \forall v < u \ \text{OT}[v] \neq \text{OT}[u]\}$;
$\mathbb{F}_\mathbb{A} \leftarrow \{q_u \in Q : \text{OT}[u][\lambda] = 1\}$;
$\mathbb{F}_\mathbb{R} \leftarrow \{q_u \in Q : \text{OT}[u][\lambda] = 0\}$;
for $q_u \in Q$ **do**
  for $a \in \Sigma$ **do** $\delta(q_u, a) \leftarrow q_w \in Q : \text{OT}[ua] = \text{OT}[w]$
end
return $\langle \Sigma, Q, q_\lambda, \mathbb{F}_\mathbb{A}, \mathbb{F}_\mathbb{R}, \delta \rangle$

	λ	a
λ	0	0
a	0	0
aa	0	0
aab	1	0
b	1	0
ab	1	0
aaba	0	0
aabb	1	0

(a) The observation table.

	a	b
q_λ	q_λ	q_b
q_b	q_λ	q_b

(b) The transition table.



(c) Automaton.

Consistency

Consistent table (with automaton)

Given an automaton $\mathcal{A}$ and an observation table $\langle \text{STA}, \text{EXP}, \text{OT} \rangle$, $\mathcal{A}$ is **consistent** with table when the following holds:

- $\text{OT}[r][c] = 1 \implies rc \in \mathbb{L}_{\mathbb{F}_{\mathbb{A}}}(\mathcal{A})$
- $\text{OT}[r][c] = 0 \implies rc \in \mathbb{L}_{\mathbb{F}_{\mathbb{R}}}(\mathcal{A})$

Theorem (Consistency)

Let $\langle STA, EXP, OT \rangle$be an observation table (closed and complete). If STA is prefix-closed and EXP is suffix-closed then LSTAR-BUILDAUTOMATON($\langle STA, EXP, OT \rangle$) is consistent with the data in $\langle STA, EXP, OT \rangle$

Proof:
LSTAR-BUILDAUTOMATON($\langle \text{STA}, \text{EXP}, \text{OT} \rangle$) is built from the data from $\langle \text{STA}, \text{EXP}, \text{OT} \rangle$ $\square$

Row consistency

Consistent table

A table is consistent if every equivalent pair of rows in *Red* remains equivalent in STA after appending any symbol
$\mathrm{OT}[r_1] = \mathrm{OT}[r_2] \implies \forall a \in \Sigma, \mathrm{OT}[r_1 a] = \mathrm{OT}[r_2 a]$

If table is inconsistent, then let $a \in \Sigma$ be the symbol for which the implication fails, and e the experiment for which the inconsistency has been found ($\mathrm{OT}[r_1 a][e] \neq \mathrm{OT}[r_2 a][e]$)
Then by adding experiment ae to the table, rows r_1 and r_2 are no longer equivalent ($\mathrm{OT}[r_1][ae] \neq \mathrm{OT}[r_2][ae]$)
Notice that experiments remain suffix-closed

Row consistency

Examples

Table (a) is inconsistent: rows a and ab look the same, but, upon experiment a, rows aa and aba are different. Column aa is added, resulting in table (b)
Table (c) is consistent, since we have not only $\mathrm{OT}[a] = \mathrm{OT}[ab]$, but also $\mathrm{OT}[aa] = \mathrm{OT}[aba]$ and $\mathrm{OT}[ab] = \mathrm{OT}[abb]$

	λ	a
λ	0	1
a	1	0
ab	1	0
b	1	0
aa	0	1
aba	0	0
abb	1	0

(a) An inconsistent table (because of a and ab).

	λ	a	aa
λ	0	1	0
a	1	0	1
ab	1	0	0
b	1	0	
aa	0	1	
aba	0	1	
abb	1	0	

(b) The table has become consistent.

	λ	a
λ	0	1
a	1	0
ab	1	0
b	1	0
aa	0	1
aba	0	1
abb	1	0

(c) A consistent table.

Fig. 13.5. Consistency.

Algorithm concept

Once the learner has built a complete, closed and consistent table, it can construct the DFA using Algorithm LSTAR-BUILDAUTOMATON and make an equivalence query

If the Oracle returns a counterexample u, then the learner should add all prefixes of u as *Red* states, and complete the *Blue* section, with all strings pa ($a \in \Sigma$, p is a prefix of u but pa is not)
In this way, at least one new *Red* line has been added

The Algorithm

Algorithm 13.2: LSTAR Learning Algorithm.

Input: –
Output: DFA $\mathcal{A}$
LSTAR-INITIALISE;
repeat
  while $\langle$STA, EXP, OT$\rangle$ *is not closed or not consistent* **do**
    if $\langle$STA, EXP, OT$\rangle$ *is not closed* **then**
    $\langle$STA, EXP, OT$\rangle \leftarrow$ LSTAR-CLOSE($\langle$STA, EXP, OT$\rangle$);
    if $\langle$STA, EXP, OT$\rangle$ *is not consistent* **then**
    $\langle$STA, EXP, OT$\rangle \leftarrow$ LSTAR-CONSISTENT($\langle$STA, EXP, OT$\rangle$)
  end
  Answer $\leftarrow$ EQ($\langle$STA, EXP, OT$\rangle$);
  if Answer $\neq$ YES **then**
  $\langle$STA, EXP, OT$\rangle \leftarrow$ LSTAR-USEEQ($\langle$STA, EXP, OT$\rangle$, Answer)
until Answer $=$ YES ;
return LSTAR-BUILDAUTOMATON($\langle$STA, EXP, OT$\rangle$)

The Algorithm

Algorithm 13.1: LSTAR-BUILDAUTOMATON.

Input: a closed and complete observation table (STA, EXP, OT)

Output: DFA $\langle \Sigma, Q, q_\lambda, \mathbb{F}_\mathbb{A}, \mathbb{F}_\mathbb{R}, \delta \rangle$

$Q \leftarrow \{q_u : u \in \textsc{Red} \wedge \forall v < u \; \mathrm{OT}[v] \neq \mathrm{OT}[u]\}$;

$\mathbb{F}_\mathbb{A} \leftarrow \{q_u \in Q : \mathrm{OT}[u][\lambda] = 1\}$;

$\mathbb{F}_\mathbb{R} \leftarrow \{q_u \in Q : \mathrm{OT}[u][\lambda] = 0\}$;

for $q_u \in Q$ **do**

  for $a \in \Sigma$ **do** $\delta(q_u, a) \leftarrow q_w \in Q : \mathrm{OT}[ua] = \mathrm{OT}[w]$

end

return $\langle \Sigma, Q, q_\lambda, \mathbb{F}_\mathbb{A}, \mathbb{F}_\mathbb{R}, \delta \rangle$

The Algorithm

Algorithm 13.3: LSTAR-INITIALISE.

Input: –
Output: table $\langle \text{STA}, \text{EXP}, \text{OT} \rangle$
$\textsc{Red} \leftarrow \{q_\lambda\}$;
$\textsc{Blue} \leftarrow \{q_a : a \in \Sigma\}$;
$\text{EXP} \leftarrow \{\lambda\}$;
$\text{OT}[\lambda][\lambda] \leftarrow \text{MQ}(\lambda)$;
for $a \in \Sigma$ **do** $\text{OT}[a][\lambda] \leftarrow \text{MQ}(a)$;
return $\langle \text{STA}, \text{EXP}, \text{OT} \rangle$

The Algorithm

Algorithm 13.4: LSTAR-CLOSE.

Input: a table $\langle \text{STA}, \text{EXP}, \text{OT} \rangle$

Output: table $\langle \text{STA}, \text{EXP}, \text{OT} \rangle$ updated

for $s \in \textsc{Blue}$ *such that* $\forall u \in \textsc{Red}\ \text{OT}[s] \neq \text{OT}[u]$ **do**

  $\textsc{Red} \leftarrow \textsc{Red} \cup \{s\}$;

  $\textsc{Blue} \leftarrow \textsc{Blue} \setminus \{s\}$;

  for $a \in \Sigma$ **do** $\textsc{Blue} \leftarrow \textsc{Blue} \cup \{s \cdot a\}$;

  for $u, e \in \Sigma^\star$ *such that* $\text{OT}[u][e]$ *is a hole* **do** $\text{OT}[u][e] \leftarrow \text{MQ}(ue)$

end

return $\langle \text{STA}, \text{EXP}, \text{OT} \rangle$

The Algorithm

Algorithm 13.5: LSTAR-CONSISTENT.

Input: a table $\langle \text{STA}, \text{EXP}, \text{OT} \rangle$

Output: table $\langle \text{STA}, \text{EXP}, \text{OT} \rangle$ updated

find $s_1, s_2 \in \textsc{Red}$, $a \in \Sigma$ and $e \in \text{EXP}$ such that $\text{OT}[s_1] = \text{OT}[s_2]$ and $\text{OT}[s_1 \cdot a][e] \neq \text{OT}[s_2 \cdot a][e]$;

$\text{EXP} \leftarrow \text{EXP} \cup \{a \cdot e\}$;

for $u, e \in \Sigma^\star$ *such that* $\text{OT}[u][e]$ *is a hole* **do** $\text{OT}[u][e] \leftarrow \text{MQ}(ue)$;

return $\langle \text{STA}, \text{EXP}, \text{OT} \rangle$

The Algorithm

Algorithm 13.6: LSTAR-USEEQ.

Input: a table $\langle \text{STA}, \text{EXP}, \text{OT} \rangle$, string Answer

Output: table $\langle \text{STA}, \text{EXP}, \text{OT} \rangle$ updated

for $p \in \text{PREF}(\text{Answer})$ **do**

  $\text{RED} \leftarrow \text{RED} \cup \{p\}$;

  for $a \in \Sigma : pa \notin \text{PREF}(\text{Answer})$ **do** $\text{BLUE} \leftarrow \text{BLUE} \cup \{pa\}$

end

for $u, e \in \Sigma^\star$ *such that* $\text{OT}[u][e]$ *is a hole* **do** $\text{OT}[u][e] \leftarrow \text{MQ}(ue)$;

return $\langle \text{STA}, \text{EXP}, \text{OT} \rangle$

Example

Algorithm 13.2: LSTAR Learning Algorithm.

Input: –
Output: DFA $\mathcal{A}$
LSTAR-INITIALISE;
repeat
  while $\langle$STA, EXP, OT$\rangle$ *is not closed or not consistent* **do**
    if $\langle$STA, EXP, OT$\rangle$ *is not closed* **then**
    $\langle$STA, EXP, OT$\rangle$ $\leftarrow$ LSTAR-CLOSE($\langle$STA, EXP, OT$\rangle$);
    if $\langle$STA, EXP, OT$\rangle$ *is not consistent* **then**
    $\langle$STA, EXP, OT$\rangle$ $\leftarrow$ LSTAR-CONSISTENT($\langle$STA, EXP, OT$\rangle$)
  end
  Answer $\leftarrow$ EQ($\langle$STA, EXP, OT$\rangle$);
  if Answer $\neq$ YES **then**
  $\langle$STA, EXP, OT$\rangle$ $\leftarrow$ LSTAR-USEEQ($\langle$STA, EXP, OT$\rangle$, Answer)
until Answer= YES ;
return LSTAR-BUILDAUTOMATON($\langle$STA, EXP, OT$\rangle$)

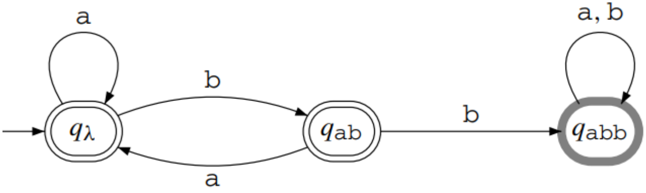


Fig. 13.8. Automaton after running LSTAR.

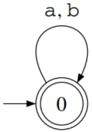
Example

Algorithm 13.2: LSTAR Learning Algorithm.

Input: –
Output: DFA $\mathcal{A}$
LSTAR-INITIALISE;
repeat
  while ⟨STA, EXP, OT⟩ *is not closed or not consistent* **do**
    if ⟨STA, EXP, OT⟩ *is not closed* **then**
    ⟨STA, EXP, OT⟩ ← LSTAR-CLOSE(⟨STA, EXP, OT⟩);
    if ⟨STA, EXP, OT⟩ *is not consistent* **then**
    ⟨STA, EXP, OT⟩ ← LSTAR-CONSISTENT(⟨STA, EXP, OT⟩)
  end
  Answer← EQ(⟨STA, EXP, OT⟩);
  if Answer≠ YES **then**
  ⟨STA, EXP, OT⟩ ← LSTAR-USEEQ(⟨STA, EXP, OT⟩, Answer)
until Answer= YES ;
return LSTAR-BUILDAUTOMATON(⟨STA, EXP, OT⟩)

	λ
λ	1
a	1
b	1

(a) A consistent table.



(b) The automaton corresponding to the Table 13.6(a).

Fig. 13.6. Consistency.

Example

Algorithm 13.2: LSTAR Learning Algorithm.

```
Input: –
Output: DFA 𝒜
LSTAR-INITIALISE;
repeat
    while ⟨STA, EXP, OT⟩ is not closed or not consistent do
        if ⟨STA, EXP, OT⟩ is not closed then
        ⟨STA, EXP, OT⟩ ← LSTAR-CLOSE(⟨STA, EXP, OT⟩);
        if ⟨STA, EXP, OT⟩ is not consistent then
        ⟨STA, EXP, OT⟩ ← LSTAR-CONSISTENT(⟨STA, EXP, OT⟩)
    end
    Answer ← EQ(⟨STA, EXP, OT⟩);
    if Answer ≠ YES then
    ⟨STA, EXP, OT⟩ ← LSTAR-USEEQ(⟨STA, EXP, OT⟩, Answer)
until Answer = YES ;
return LSTAR-BUILDAUTOMATON(⟨STA, EXP, OT⟩)
```

	λ
λ	1
a	1
ab	
abb	0
b	1
aa	
aba	
abba	
abbb	

(a) Table after equivalence query returned abb (as not in L).

	λ
λ	1
a	1
ab	1
abb	0
b	1
aa	1
aba	1
abba	0
abbb	0

(b) Membership queries are made: Table is not closed.

	λ	b
λ	1	1
a	1	1
ab	1	0
abb	0	0
b	1	
aa	1	
aba	1	
abba	0	
abbb	0	

(c) Adding a column to make the table closed.

	λ	b
λ	1	1
a	1	1
ab	1	0
abb	0	0
b	1	0
aa	1	1
aba	1	1
abba	0	0
abbb	0	0

(d) The table after filling the holes is closed and consistent.

Fig. 13.7. Running LSTAR.

Example

Algorithm 13.2: LSTAR Learning Algorithm.

Input: –
Output: DFA $\mathcal{A}$
LSTAR-INITIALISE;
repeat
    while ⟨STA, EXP, OT⟩ *is not closed or not consistent* **do**
        if ⟨STA, EXP, OT⟩ *is not closed* **then**
        ⟨STA, EXP, OT⟩ ← LSTAR-CLOSE(⟨STA, EXP, OT⟩);
        if ⟨STA, EXP, OT⟩ *is not consistent* **then**
        ⟨STA, EXP, OT⟩ ← LSTAR-CONSISTENT(⟨STA, EXP, OT⟩)
    end
    Answer← EQ(⟨STA, EXP, OT⟩);
    if Answer≠ YES **then**
    ⟨STA, EXP, OT⟩ ← LSTAR-USEEQ(⟨STA, EXP, OT⟩, Answer)
until Answer= YES ;
return LSTAR-BUILDAUTOMATON(⟨STA, EXP, OT⟩)

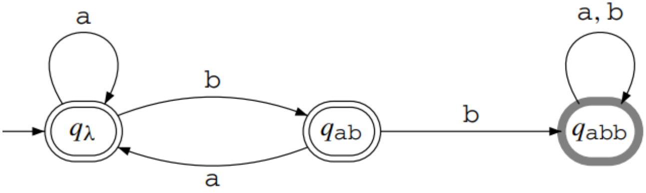


Fig. 13.8. Automaton after running LSTAR.

Proof

We have to show that algorithm terminates, and that it returns correct automaton

Every regular language admits a unique DFA, so we can assume (wlg) that this is our target, and it has n states
Since any DFA consistent with a table has at least as many states as different *Red* rows, and construction of consistent DFA is unique, the algorithm has to end when it has n rows

- each closure failure adds different row to *Red*
- each inconsistency failure adds one experiment
- each counterexample adds at least one different row to *Red*

Thus, each time a table is inconsistent or we get a counterexample, at least one different *Red* row is introduced
Number of steps between any of these operations is bounded, so the entire running time is bounded

Complexity

- each experiment introduces new different *Red* row, number of which is limited by n $\Rightarrow |EXP| \leq n$
- for same reason at most n equivalence queries are made
- maximal size of observation table is n columns and $nm|\Sigma|$ rows (m is max length of counterexample - thus it won't generate more than nm *Red* rows, thus there will be at most $nm|\Sigma|$ *Blue* rows)

Therefore we make at most $n^2m|\Sigma|$ MQ queries and n EQ queries

Implementation issues

There is an issue with handling redundancy. Instead of storing an entire observation table (which could be large and inefficient), we will store 3 association tables:

- MQ table
- Prefix (states) table
- Suffix (experiments) table

The actual observation table is simulated by a function $\mathrm{OT}(r, c) := \mathrm{MQ}[rc]$

Some more examples

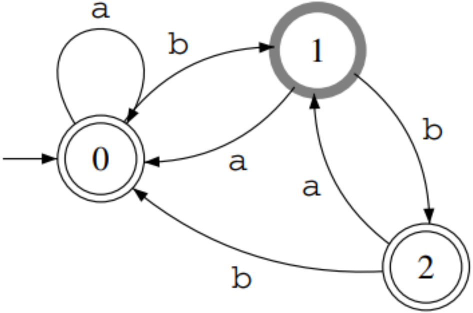


Fig. 13.9. A target automaton.

Some more examples

Algorithm 13.2: LSTAR Learning Algorithm.

Input: –
Output: DFA $\mathcal{A}$
LSTAR-INITIALISE;
repeat
    while $\langle$STA, EXP, OT$\rangle$ *is not closed or not consistent* **do**
        if $\langle$STA, EXP, OT$\rangle$ *is not closed* **then**
        $\langle$STA, EXP, OT$\rangle \leftarrow$ LSTAR-CLOSE($\langle$STA, EXP, OT$\rangle$);
        if $\langle$STA, EXP, OT$\rangle$ *is not consistent* **then**
        $\langle$STA, EXP, OT$\rangle \leftarrow$ LSTAR-CONSISTENT($\langle$STA, EXP, OT$\rangle$)
    end
    Answer$\leftarrow$ EQ($\langle$STA, EXP, OT$\rangle$);
    if Answer$\neq$ YES **then**
    $\langle$STA, EXP, OT$\rangle \leftarrow$ LSTAR-USEEQ($\langle$STA, EXP, OT$\rangle$, Answer)
until Answer$=$ YES ;
return LSTAR-BUILDAUTOMATON($\langle$STA, EXP, OT$\rangle$)

Algorithm 13.4: LSTAR-CLOSE.

Input: a table $\langle$STA, EXP, OT$\rangle$
Output: table $\langle$STA, EXP, OT$\rangle$ updated
for $s \in$ BLUE *such that* $\forall u \in$ RED OT$[s] \neq$ OT$[u]$ **do**
    RED $\leftarrow$ RED $\cup \{s\}$;
    BLUE $\leftarrow$ BLUE $\setminus \{s\}$;
    for $a \in \Sigma$ **do** BLUE $\leftarrow$ BLUE $\cup \{s \cdot a\}$;
    for $u, e \in \Sigma^\star$ *such that* OT$[u][e]$ *is a hole* **do** OT$[u][e] \leftarrow$ MQ(ue)
end
return $\langle$STA, EXP, OT$\rangle$

Algorithm 13.5: LSTAR-CONSISTENT.

Input: a table $\langle$STA, EXP, OT$\rangle$
Output: table $\langle$STA, EXP, OT$\rangle$ updated
find $s_1, s_2 \in$ RED, $a \in \Sigma$ and $e \in$ EXP such that OT$[s_1] =$ OT$[s_2]$ and OT$[s_1 \cdot a][e] \neq$ OT$[s_2 \cdot a][e]$;
EXP $\leftarrow$ EXP $\cup \{a \cdot e\}$;
for $u, e \in \Sigma^\star$ *such that* OT$[u][e]$ *is a hole* **do** OT$[u][e] \leftarrow$ MQ(ue);
return $\langle$STA, EXP, OT$\rangle$

Algorithm 13.1: LSTAR-BUILDAUTOMATON.

Input: a closed and complete observation table (STA, EXP, OT)
Output: DFA $\langle \Sigma, Q, q_\lambda, \mathbb{F}_\mathbb{A}, \mathbb{F}_\mathbb{R}, \delta \rangle$
$Q \leftarrow \{q_u : u \in \text{RED} \wedge \forall v < u \text{ OT}[v] \neq \text{OT}[u]\}$;
$\mathbb{F}_\mathbb{A} \leftarrow \{q_u \in Q : \text{OT}[u][\lambda] = 1\}$;
$\mathbb{F}_\mathbb{R} \leftarrow \{q_u \in Q : \text{OT}[u][\lambda] = 0\}$;
for $q_u \in Q$ **do**
    for $a \in \Sigma$ **do** $\delta(q_u, a) \leftarrow q_w \in Q : \text{OT}[ua] = \text{OT}[w]$
end
return $\langle \Sigma, Q, q_\lambda, \mathbb{F}_\mathbb{A}, \mathbb{F}_\mathbb{R}, \delta \rangle$

Some more examples