Avoiding Dead States in Query Learning of Regular Tree Languages

Frank Drewes

(**work done in collaboration with** **Johanna Högberg**)

Introduction - the subject

Algorithmic task Learn an initially unknown regular tree language U,
i.e., construct a finite tree automaton (fta) A such that $L(A) = U$.

Source of information about U A "teacher" who can answer

- membership queries – given a tree t, is it true that $t \in U$?
- equivalence queries – given an fta A, is it true that $L(A) = U$? Otherwise, return a tree which is a counterexample.

This is the popular MAT model for algorithmic learning (MAT='minimally adequate teacher') introduced by Angluin in 1987.

Introduction - the background

- In 1987, Angluin proposed a polynomial algorithm that learns a regular language U, returning the minimal finite automaton recognizing U.
- In 1990, Sakakibara extended this to regular tree languages. However,
  - the running time is polynomial only if the alphabet is fixed,
  - the size of counterexamples returned by the teacher affects the running time (too) heavily, and
  - as the fta constructed is total, it may be exponentially larger than the minimal partial fta recognizing U (unless the alphabet is fixed).

$\Rightarrow$ Sakakibara: Can we avoid dead states? It turns out that our solution to this problem is in fact a remedy for all three disadvantages.

Trees

- Trees are built over a ranked alphabet Σ (i.e., tree $=$ term).
- The notation $f^{(k)}$ indicates that $f \in \Sigma$ is of rank k.
- A tree with root $f^{(k)}$ and direct subtrees $t_1, \ldots, t_k$ is written $f[t_1, \ldots, t_k]$ (or simply f if $k = 0$).
- For a set T of trees, $\Sigma(T) = \{f[t_1, \ldots, t_k] \mid f^{(k)} \in \Sigma$ and $t_1, \ldots, t_k \in T\}$.
- A context is a tree c containing exactly one occurrence of $\Box^{(0)}$.
- If c is a context and t a tree, then $c[\![t]\!]$ is obtained by substituting t for $\Box$ in c.
- A tree language is a set of trees.

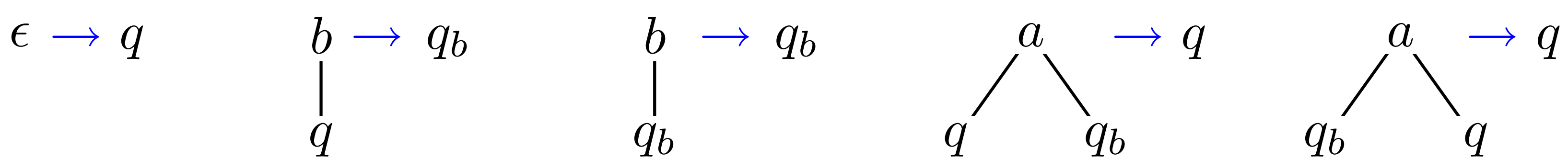
Finite tree automata (running example)

$U =$ all trees over $a^{(2)}, b^{(1)}, \epsilon^{(0)}$ in which precisely one child of each a is a b.

Fta: states q, q_b, q' (where q, q_b are accepting)

transitions

$\delta(\lambda, \epsilon) = q$ $\quad \delta(q, b) = q_b$ $\quad \delta(q_b, b) = q_b$ $\quad \delta(q q_b, a) = q$ $\quad \delta(q_b q, a) = q$



$\delta(\ldots, \ldots) = q'$ in all other cases (q' is a dead state)

Recall Myhill-Nerode: Trees $s = b[\epsilon]$ and $s' = b[b[\epsilon]]$ are equivalent in every context and need not be distinguished. This is why $\delta^*(s) = q_b = \delta^*(s')$.

The original idea

Angluin's idea is inspired by the Myhill-Nerode theorem:

- Trees s, s' are equivalent with respect to U
  iff $\delta^*(s) = \delta^*(s')$ in the minimal fta recognizing U
  iff $c[\![s]\!] \in U \iff c[\![s']\!] \in U$ for all contexts c.

- The algorithm collects

  (a) a set $S = \{s_1, \ldots, s_m\}$ of trees representing equivalence classes and

  (b) a set $C = \{c_1, \ldots, c_n\}$ of contexts distinguishing between the classes.

Intuitively, $S = \{s_1, \ldots, s_m\}$ and $C = \{c_1, \ldots, c_n\}$ yield an observation table:

	c_1	c_2	$\cdots$	c_n
s_1	$-$	$+$	$\cdots$	$+$
s_2	$+$	$+$	$\cdots$	$-$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
s_m	$-$	$-$	$\cdots$	$+$

set $C = \{c_1, \ldots, c_n\}$ of contexts

records the observation that $c_n[\![s_2]\!] \notin U$

observations $obs_C(s_m)$ made for s_m

Fta proposed to the teacher:

- Use the observations $obs_C(s_1), \ldots, obs_C(s_m)$ (the table rows) as states.
- Define $\delta(obs_C(s_{i_1}) \cdots obs_C(s_{i_k}), f) = obs_C(f[s_{i_1}, \ldots, s_{i_k}])$.
- Let $obs_C(s_i)$ be accepting iff $s_i \in U$.

Possible table in our running example (final stage):

	$\square$	$b[\square]$	$a[\epsilon, \square]$	
$a[b[b[b[\epsilon]]], \epsilon]$	$+$	$+$	$-$	correspond to q
ϵ	$+$	$+$	$-$	
$b[b[b[\epsilon]]]$	$+$	$+$	$+$	correspond to q_b
$b[b[\epsilon]]$	$+$	$+$	$+$	
$b[\epsilon]$	$+$	$+$	$+$	
$a[\epsilon, \epsilon]$	$-$	$-$	$-$	corresponds to q'

In the automaton constructed, we have, e.g., $\delta(\langle + + - \rangle \langle + + + \rangle, a) = \langle + + - \rangle$ because $obs_C(a[\epsilon, b[\epsilon]]) = \langle + + - \rangle$.

Some disadvantages

- If the teacher returns a counterexample s, each subtree of s is added to the table. This results in
  (a) a large table (equivalence classes are represented many times)
  (b) large trees (if the teacher returns large counterexamples)
- If $obs_C(s_i) = \langle - \cdots - \rangle$, then s_i may (!) represent a dead state.
- Note: These disadvantages are of little importance in Angluin's case.

The proposed approach

We maintain a third set $R \supseteq S$ of trees representing transitions.

- We always have $S \subseteq R \subseteq \Sigma(S)$ and $obs_C(R) \subseteq obs_C(S)$.
- As before, each $obs_C(s)$, $s \in S$, is a state.
- Each $r = f[s_1, \ldots, s_k] \in R$ yields the transition

$$\delta(obs_C(s_1) \cdots obs_C(s_k), f) = obs_C(r).$$

- Additional properties
  - Distince $s, s' \in S$ yield distinct states, i.e., $obs_C(s) \neq obs_C(s')$.
  - Distinct $r, r' \in R$ yield distinct transitions.
  - $|C| \leq |S|$.
  - No tree in s corresponds to a dead state.

The main procedure (the "learner") is a simple loop:

$T = (S, R, C)$ `:=` $(\emptyset, \emptyset, \emptyset)$;
`loop`
  A `:=` automaton resulting from T;
  t `:=` $\textsc{Counterexample}(A)$; *(ask teacher whether $L(A) = U$)*
  `if` $t = \bot$ `then return` A *(teacher said $L(A) = U$)*
  `else` T `:=` $\textsc{Extend}(T, t)$ — the tricky part
`end loop`

Extending the table (example)

Table after the first step (with $R = S = \{\epsilon\}$ and $C = \emptyset$):

$$\begin{array}{c||}
 \\ \hline\hline
\epsilon
\end{array}
\qquad
\begin{array}{l}
\Rightarrow \delta(\lambda, \epsilon) = \langle\rangle \\
\Rightarrow L(A) = \{\epsilon\} \text{ (}\langle\rangle\text{ is accepting since } \epsilon \in U\text{)} \\
\Rightarrow \text{a counterexample is, e.g., } t = b[a[b[b[\epsilon]], \epsilon]]
\end{array}$$

EXTEND chooses any subtree $r \in \Sigma(S) \setminus S$. Say, $t = c[\![r]\!] = b[a[b[b[\epsilon]], \epsilon]]$.

Case 1 If $r \notin R$, then r represents a missing transition and is added to R:

$$R \begin{cases} S\{ & \epsilon \\ & b[\epsilon] \end{cases}
\begin{array}{c||}
 \\ \hline\hline
\epsilon \\ \hline
b[\epsilon]
\end{array}
\qquad
\begin{array}{l}
\Rightarrow \delta(\langle\rangle, b) = \delta(\lambda, \epsilon) = \langle\rangle \\
\Rightarrow L(A) = \text{all trees of the form } b[\cdots b[\epsilon] \cdots] \\
\Rightarrow b[a[b[b[\epsilon]], \epsilon]] \text{ continues to be a counterexample}
\end{array}$$

R	S	ϵ
R		$b[\epsilon]$

Recall: the counterexample is still $b[a[b[b[\epsilon]], \epsilon]]$.

Case 2: Decomposition yields $t = c[\![r]\!] = b[a[b[b[\epsilon]], \epsilon]]$, again, but now $r \in R$!
EXTEND tries to make the counterexample smaller by replacing r with the unique $s \in S$ satisfying $obs_C(s) = obs_C(r)$ (i.e., with $s = \epsilon$).
Membership queries reveal that $c[\![s]\!] = b[a[b[\epsilon], \epsilon]]$ is indeed a counterexample.
$\Rightarrow$ set $t := c[\![s]\!]$ and continue with this counterexample.

Case 3: Now, decomposition yields $t = c[\![r]\!] = b[a[b[\epsilon], \epsilon]]$.
Again, $r \in R$, but now $c[\![s]\!] = b[a[\epsilon, \epsilon]]$ fails to be a counterexample.
$\Rightarrow$ the context $c = b[a[\square, \epsilon]]$ distinguishes s from r
$\Rightarrow$ c is added to C and r moved into S.

	$b[a[\Box, \epsilon]]$
ϵ	$-$
$b[\epsilon]$	$+$

$\Rightarrow$ $\delta(\lambda, \epsilon) = \langle - \rangle$ and $\delta(\langle - \rangle, b) = \langle + \rangle$

$\Rightarrow$ $L(A) = \{\epsilon, b[\epsilon]\}$

$\Rightarrow$ still, $b[a[b[\epsilon], \epsilon]]$ is a counterexample

$\Rightarrow$ $r = a[b[\epsilon], \epsilon]$ represents a missing transition

	$b[a[\Box, \epsilon]]$
ϵ	$-$
$b[\epsilon]$	$+$
$a[b[\epsilon], \epsilon]$	$-$

the new transition is $\delta(\langle + \rangle\langle - \rangle, a) = \langle - \rangle$

$\Rightarrow$ the counterexample $a[\epsilon, b[b[\epsilon]]]$ may be used twice

1. the decomposition $a[\epsilon, b[b[\epsilon]]]$ adds $b[b[\epsilon]]$ to R
2. $a[\epsilon, b[b[\epsilon]]] \rightsquigarrow a[\epsilon, b[\epsilon]]$ adds $a[\epsilon, b[\epsilon]]$ to R

	$b[a[\Box, \epsilon]]$
ϵ	$-$
$b[\epsilon]$	$+$
$a[b[\epsilon], \epsilon]$	$-$
$b[b[\epsilon]]$	$+$
$a[\epsilon, b[\epsilon]]$	$-$

The final table, which yields the correct fta.

The three cases of EXTEND (summary):

EXTEND decomposes t as $t = c[\![r]\!]$ with $r \in \Sigma(S) \setminus S$.

Case 1 $r \notin R$

$\Rightarrow$ add r to R (and to S if $obs_C(r) \notin obs_C(S)$)

Otherwise, there is a unique $s \in S$ with $obs_C(s) = obs_C(r)$.

Case 2 $c[\![s]\!]$ is also a counterexample (check by asking membership queries)

$\Rightarrow$ continue with $c[\![s]\!]$ as the counterexample.

Case 3 $c[\![s]\!]$ is not a counterexample

$\Rightarrow$ the context c proves that $c[\![r]\!]$ and $c[\![s]\!]$ are inequivalent

$\Rightarrow$ add c to C and move r into S.

This is also known as Shapiro's contradiction backtracking technique.

The running time of the learner

- $S \subseteq R \subseteq \Sigma(S)$ means that R can be represented as a dag with $|R|$ nodes.
- Basically, the recursion of EXTEND (repeatedly replacing $c[\![r]\!]$ with $c[\![s]\!]$) takes time linear in the size of the counterexample.
- Most of the time, new transitions are added. Then the dag representing R and the resulting fta can be updated without recomputing them from scratch.
- More time-consuming recomputations are only required in the seldom case where a new context is added (recall that $|C| \leq |S|$).

If (Σ, Q, δ, F) is the minimial partial fta recognizing U, r the maximum rank of symbols in Σ, and m the maximum size of counterexamples, then the overall running time of the learner is $O(r \cdot |Q| \cdot |\delta| \cdot (|Q| + m))$.
This does not include the time required by the teacher.

How many queries are asked?

Equivalence queries Each iteration of the main loop enlarges S or R

$\Rightarrow$ at most $|Q| \cdot |\delta|$ equivalence queries.

Optimization: reuse counterexamples as long as possible.

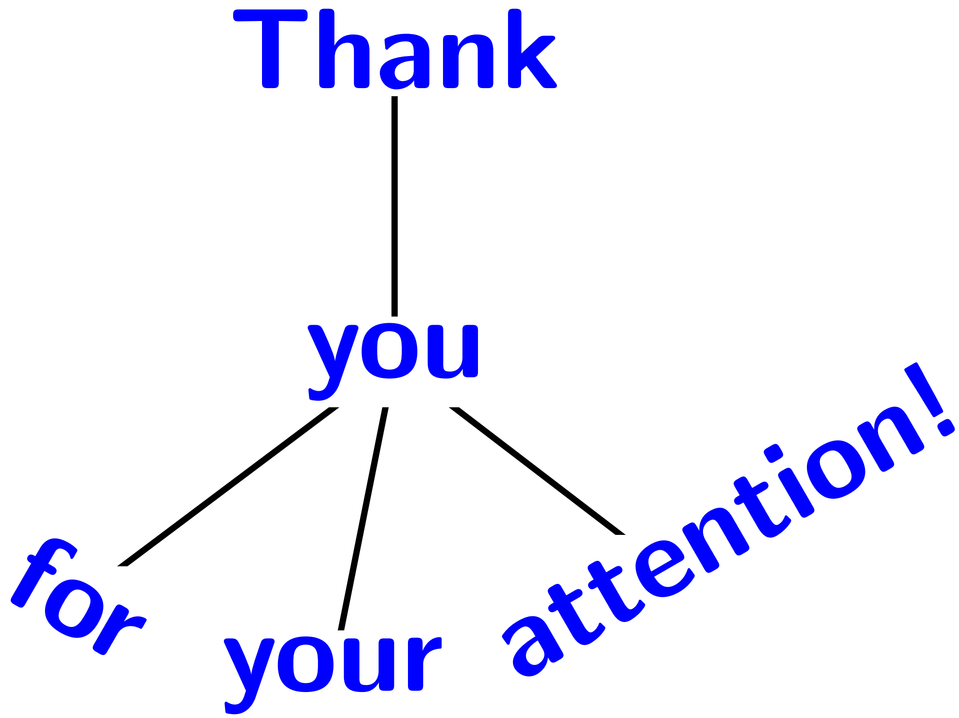
Membership queries

- $|Q| \cdot |\delta|$ entries of the observation table must be filled.
- $M =$ sum of sizes of counterexamples queries are needed to shrink the counterexamples.
- $|Q|$ queries are needed to determine whether states are accepting.

$\Rightarrow$ at most $M + |Q| \cdot (|\delta| + 1)$ membership queries.

Concluding remarks

- The gain regarding efficiency compared with Angluin/Sakakibara depends on U since the total fta recognizing U has about $|Q|^r$ transitions whereas the partial one sometimes has only $|Q|$ transitions.
- It also depends on the behaviour of the teacher since large counterexamples to not matter so much in our approach.
- Some open questions:
  - Identify language classes where the teacher can find counterexamples that reveal much about U ($=$ counterexamples that can be reused).
  - Can the approach be improved in such a way that fewer equivalence queries (e.g., only $O(|Q|)$) are used?
  - Learning of weighted tree automata???


Thank
you
for
your
attention!