

Our tree automata proof of decidability for

FOL(K)-separation over the signature

$\mathcal{I}_{\text{simple}} = \{R^2\}$ and $V = \{x, y\}$ is an

instance of a powerful technique which applies to many other symbolic learning problems.

In this exercise, you're to invent a new symbolic language L for which the same pattern of tree automata constructions can be used to prove that L has a decidable learning problem.

As a step toward that, you'll first get experience by working through the construction for $L = \text{regular expressions}$.

Part 1

Simplified Regular Expressions

Syntax

Let $\text{Reg}_\Sigma := L(\varphi)$

Alphabet $\Sigma = \{a, b\}$

$\varphi ::= x \in \Sigma \mid \varphi \cdot \varphi' \mid \varphi^*$

Semantics

$w \in \Sigma^* \models \varphi$ means " φ matches w "

$w_1 w_2 \dots w_n \models \varphi$ iff $w_1 w_2 \dots w_n, (1, n+1) \models \varphi$

$w, (l, l+1) \models x \in \Sigma$ iff $w_l = x$ and

$w, (l, r) \models \varphi \cdot \varphi'$ iff $\exists k \in [l, r]$
s.t. $w, (l, k) \models \varphi$ and
 $w, (k, r) \models \varphi'$

$w, (l, r) \models \varphi^*$ iff $\exists k \in [l+1, r]$
s.t. $w, (l, k) \models \varphi$ and $w, (k, r) \models \varphi^*$

So we've defined syntax and semantics of a simple regular expression language Reg_Σ .

Problem: Reg_Σ -separation Given finite sets

of positive words $P \subseteq \Sigma^+$ and negative words $N \subseteq \Sigma^+$, and given (tree) grammar G ,
output $\varphi \in L(G) \in \text{Reg}_\Sigma$ s.t.

$w \models \varphi$ for every $w \in P$ and

$w \not\models \varphi$ for every $w \in N$.

We follow exactly the same strategy as for $\text{FOL}(K)$ -separation to design a decision procedure for Reg_Σ -separation.

You will consider only the case of automata A_w for $w \in P$.

Fill in the missing parts of the definition for the automaton A_w , $w \in P$, s.t.

$$L(A_w) = \{ \varphi \in \text{Reg}\Sigma : w \models \varphi \}.$$

$$A_w = (Q, \Sigma', I, \delta), \text{ where}$$

$$Q = \{ (i, j) \in \mathbb{N}^2 : i \leq j \text{ and } i, j \in \{1, \dots, |w|+1\} \}.$$

$$\Sigma' = \{ a^0, b^0, \cdot^2, *^2 \}$$

$$I \subseteq Q = \text{Fill}$$

$$\delta((i, j), a) = \begin{cases} \text{true} & \text{if } w_i = a \text{ and } j = i+1 \\ \text{false} & \text{otherwise} \end{cases}$$

$$\delta((i, j), b) = \text{Fill}$$

$$\delta((i, j), \cdot) = \bigvee_{k \in [i, j]} \text{Fill}$$

$$\delta((i, j), *) = \text{Fill}$$

Describe a high-level proof strategy for proving:

$$L(A_w) = \{ \varphi \in \text{Reg}_Z : w \models \varphi \}.$$

Fill

Part 2

Define Syntax and semantics for a new language L , state a learning problem for L , and argue L has decidable learning by exhibiting an automaton construction $M \in \mathcal{M} \mapsto A_M$ s.t. A_M accepts $e \in E$ precisely when $M \models e$.

Define structures $\mathcal{M}$, expressions E , and semantic relation $M \models e$, for $M \in \mathcal{M}$, $e \in E$.

Notes: Your language does not need to be realistic or useful. Do not refer to my publications to find an example.

Hint on next page.

Hint: What do regular expressions and FOL(K) have in common that made the automata-based learning construction possible? Given any structure $M \in \mathcal{M}$, the number of states in the automaton A_M is finite, though $|Q|$ can grow with $|M|$.

Consider:

$$\text{FOL}(K) \text{ had } Q = [V \rightarrow D(M)] \times \{0,1\}$$

all partial assignments to variables $V = \{x, y\}$, quadratic in $|D(M)|$

$$\text{Reg}_\Sigma \text{ had } Q =$$

$$\{(i, j) \in \mathbb{N}^2 : i \leq j \text{ and } i, j \in \{1, \dots, |w|+1\}\}$$

quadratic in $|w|$

flag for handling negation

For any graph G or word w , these states Q are enough to accurately evaluate arbitrarily large expressions $\varphi \in \text{FOL}(2)$ or $\varphi \in \text{Reg}_\Sigma$.

The class of languages with this property is called **finite aspect checkable languages (FAC)**.

So your task is to invent a new FAC language.

Structures $\mathcal{M}$ **Fill**

Expressions E **Fill**

Semantics $\models$ **Fill**

Construction $M \in \mathcal{M} \mapsto A_M$

s.t. $L(A_M) = \{ e \in E : M \models e \}$

Fill