

S_ymbolic L_earning

K-variable FOL (FOL(K))

$$t ::= \dots \mid x \in V = \{x_1, \dots, x_k\} \mid \dots$$

Fix a signature τ .

Problem FOL(K)-separation : Given finite sets of τ -structures P and N , synthesize a τ -sentence $\varphi \in \text{FOL}(K)$ such that

$$M \in P \Rightarrow M \models \varphi \quad \text{and}$$

$$M \in N \Rightarrow M \not\models \varphi,$$

or if no such φ exists, halt with no output.

Example: Zendo

game involving
supervised and active
learning of logical concepts

$\tau_{\text{Zendo}} = (\text{Pyramid}^1, \text{Wedge}^1, \text{Block}^1,$
 $\text{Red}^1, \text{Yellow}^1, \text{Blue}^1,$
 $\text{Touching}^2, \text{Stacked}^2,$
 $\text{Flat}^1, \text{Upright}^1, \dots)$

$\varphi := \exists x. \text{Red}(x) \wedge \exists y. \text{Touching}(x, y)$

Consider: Suppose $K \gg |M|$ for $M \in P \cup N$.

How might we define some correct, but "undesirable" solutions to a learning instance which are too "specific"?

Fix ^{relational} τ -structure M with $D(M) = \{1, 2, \dots, l\}$.

Claim: $M' \models \varphi_M$ iff $M' \cong M$, where:

$$\varphi_M := \bigwedge_{R \in \mathcal{R}} \varphi_R \wedge \varphi_{D(M)} \wedge \bigwedge_{c \in C} x_c^M = c$$

$$\varphi_{D(M)} := \exists x_1, \dots, x_l \cdot \left(\bigwedge_{i \neq j \in [l]} x_i \neq x_j \right) \wedge \left(\forall y. \bigvee_{i \in [l]} y = x_i \right)$$

$$\varphi_R := \bigwedge_{(i,j) \in R^M} R(x_i, x_j) \wedge \bigwedge_{(i,j) \notin R^M} \neg R(x_i, x_j)$$

Fix learning instance with examples P and N .

$$\text{Let } \varphi := \bigvee_{m \in P} \varphi_m$$

Observe: φ solves (P, N) .

Take away: Expressive-enough logics can exactly define finite structures. In a learning setting this means they are capable of "memorizing" or "overfitting" to observed data.

One strategy to mitigate "memorization" is to favor syntactically simple, i.e. short, formulas over complex, i.e. long, ones.

Observe: φ above grows longer with the input.

Another strategy to mitigate "memorization" is through the problem specification rather than through the solution.

We can require that any proposed solution φ must be drawn from a fragment of the logic which is specified in the input alongside the examples.

Note: Similar to the input format in Syntax-guided Synthesis problems (SyGuS).

Revised Problem Definition ...

Problem FOL(K) - separation : Given grammar G and

given finite sets of τ -structures P and N ,
synthesize a τ -sentence $\varphi \in L(G)$ such that

$$M \in P \Rightarrow M \models \varphi \quad \text{and}$$

$$M \in N \Rightarrow M \not\models \varphi,$$

or if no such φ exists, halt with no output.

Example Grammar G specifies the existential
conjunctive fragment of FOL(K) over τ_{Graph} :

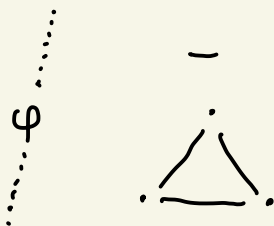
$$\varphi ::= R(\bar{t}) \mid \varphi \wedge \varphi' \mid t = t' \mid \exists x. \varphi$$

$$t ::= \dots$$

Positive graphs P



Negative graphs N



No solution!

How might we solve the FOL(k)-separation problem?

What would be a sound algorithm?

Generate all formulas in G up to size bound s and test each one against the examples P, N .

If any are consistent with the specification then output the smallest one. Otherwise halt with no output.

i.e. $\varphi ::= R(x,y) \mid Q(x,y) \mid \exists x. \varphi \mid \varphi \wedge \varphi'$

Enumerate:

$R(x,y), R(y,x), Q(x,y), Q(y,x), R(x,y) \wedge Q(x,y), \dots$

What would be a complete algorithm?

Similar to the sound algorithm except test the formulas as you generate them, and don't have any size bound s , only halting if a solution is found.

What would be a sound, complete, and terminating algorithm?

Enumeration won't work because, if there is no solution, how can we know when to stop enumerating and halt? The next formula in our enumeration might be a solution after all.

Wait, aren't many of the formulas logically equivalent? Such as $\varphi \vee \varphi'$ and $\neg(\neg\varphi \wedge \neg\varphi')$? Maybe the set of formulas is actually finite modulo logical equivalence?

No. Recall $\varphi_{0(n)} := \exists x_1, \exists x_2 \dots \exists x_n. \bigwedge_{i \neq j \in [n]} x_i \neq x_j. \dots$

$\{\varphi_{[i]}\}_{i \in \mathbb{N}}$ is an infinite family of formulas

which are pairwise logically inequivalent.

$$[i] \models \neg(\varphi_{[i]} \leftrightarrow \varphi_{[j]}) \text{ for any } i \neq j \in \mathbb{N}$$

But $\varphi_{[i]}$ uses i variables...

Maybe the fact that we're using finite-variable FOL leaves us with only finitely-many inequivalent formulas to consider?

No. Consider $\mathcal{T}_{\text{Graph}}$ and $K=3$ variables x, y, z .

$$\text{path}_1(x, y) := E(x, y)$$

$$\text{path}_{i+1}(x, y) := \exists z. E(x, z) \wedge \exists x. x = z \wedge \text{path}_i(x, y)$$

Claim: $G \models \exists x \exists y. \text{path}_i(x, y)$ iff G has a directed path of length i .

And all path_i use only 3 variables.

So even in finite-variable FOL there are infinitely-many logically-inequivalent formulas.

So, is there a sound, complete, terminating algorithm for $\text{FOL}(K)$ -separation?

Theorem. $FOL(K)$ -separation is decidable
in time $\exp(m n^K)$, where

$$m = |PUN|$$

$$n = \text{maximum size of encoding of} \\ \text{any } M \in PUN$$

$$K = \text{number of variables}$$

Overview of the algorithm (decision procedure):

1. Define tree automaton A_G s.t.

$$L(A_G) = \{ \text{ast}(\varphi) : \varphi \in L(G) \}.$$

2. Define tree automata A_M , for $M \in P$, s.t.

$$L(A_M) = \{ \text{ast}(\varphi) : M \models \varphi \}.$$

3. Define tree automata B_M , for $M \in N$, s.t.

$$L(B_M) = \{ \text{ast}(\varphi) : M \not\models \varphi \}.$$

4. Determine whether $L(A)$ is empty, where

$$A := A_G \times \bigtimes_{M \in P} A_M \times \bigtimes_{M \in N} B_M$$

Can construct $\varphi \in L(A)$ and determine non-emptiness in time $O(|A|)$.