Neuro-Symbolic Hierarchical Rule Induction

Claire Glanois † Xuening Feng † Zhaohui Jiang †
Paul Weng † Matthieu Zimmer † Dong Li ‡ Wulong Liu ‡

Abstract

We propose an efficient interpretable neuro-symbolic model to solve Inductive Logic Programming (ILP) problems. In this model, which is built from a set of meta-rules organised in a hierarchical structure, first-order rules are invented by learning embeddings to match facts and body predicates of a meta-rule. To instantiate it, we specifically design an expressive set of generic meta-rules, and demonstrate they generate a consequent fragment of Horn clauses. During training, we inject a controlled Gumbel noise to avoid local optima and employ interpretability-regularization term to further guide the convergence to interpretable rules. We empirically validate our model on various tasks (ILP, visual genome, reinforcement learning) against several state-of-the-art methods.

1 Introduction

Research on neuro-symbolic approaches [Santoro et al., 2017, Donadello et al., 2017, Manhaeve et al., 2018, Dai et al., 2019, d’Avila Garcez and Lamb, 2020] has become very active recently and has been fueled by the successes of deep learning [Krizhevsky et al., 2017] and the recognition of its limitations [Marcus, 2018]. At a high level, these approaches aim at combining the strengths of deep learning (e.g., high expressivity and differentiable learning) and symbolic methods (e.g., interpretability and generalizability) while addressing their respective limitations (e.g., brittleness for deep learning and scalability for symbolic methods).

In this paper, we specifically focus on inductive logic programming (ILP) tasks [Cropper and Dumančić, 2021]. The goal in ILP is to find a first-order logic (FOL) program that explains positive and negative examples given some background knowledge also expressed in FOL. In contrast to classic ILP methods, which are based on combinatorial search over the space of logic programs, neuro-symbolic methods (see Section 2) usually work on a continuous relaxation of this process.

For tackling ILP problems, we propose a new neuro-symbolic hierarchical model that is built upon a set of meta-rules, denoted HRI. For a more compact yet expressive representation, we introduce the notion of proto-rule, which encompasses multiple meta-rules. The valuations of predicates are computed via a soft unification between proto-rules and predicates using learnable embeddings. Like most ILP methods, our model can provide an interpretable solution and, being independent of the number of objects, manifests some combinatorial generalisation skills. In contrast to most other approaches, HRI is also independent of the number of predicates — e.g. this number may vary between training and testing. Our model can also allow recursive definitions, if needed. Since it is based on embeddings, it is able to generalize and is particularly suitable for multi-task ILP problems. Moreover, any semantic or visual priors on concepts can be leveraged, to initialize the predicate embeddings.

The design of proto-rules, crucially restricting the hypothesis space, embodies a well-known trade-off between efficiency and expressivity. Relying on minimal sets of proto-rules for rule induction models has been shown to improve both learning time and predictive accuracies [Cropper and Muggleton, 2014, Fonseca et al., 2004]. For our model to be both adaptive and efficient, we initially designed an expressive and minimal set \( R_0 \) of proto-rules. While most neuro-symbolic approaches do not formally discuss the expressivity of their models, we provide a the-

\[ \text{Trained on smaller instances, it can generalize to larger ones.} \]
oretical analysis to characterize the expressivity of $R_0$. Moreover, in contrast to ILP work based on combinatorial search [Cropper and Tourret, 2020], we found that a certain redundancy in the proto-rules may experimentally help learning in neuro-symbolic approaches. Therefore, as a replacement of $R_0$, we propose an extended set $R_\ast$ of proto-rules.

We validate our model using proto-rules in $R_\ast$ on classic ILP tasks. Despite using the same set of proto-rules, our model is competitive with other approaches that may require specifying different meta-rules for different tasks to work, which, unsatisfactorily, requires an a priori knowledge of the solution. In order to exploit the embeddings of our model and demonstrate its scalability, we distinctly evaluate our approach on a large domain, GQA [Hudson and Manning, 2019b] extracted from Visual Genome [Krishna et al., 2017]. Our model outperforms other methods such as NLIL [Yang and Song, 2020] on those tasks. We also empirically validate all our design choices.

Contributions Our contributions can be summarized as follows: (1) Hierarchical model for embedding-based rule induction (see Section 4), (2) Expressive set of generic proto-rules and theoretical analysis (see Section 4.2), (3) interpretability-oriented training method (see Section 5), (4) Empirical validation on various domains (see Section 6).

2 Related Work

Classic ILP methods Classic symbolic ILP methods [Quinlan, 1990, Muggleton, Cropper and Tourret, 2020] aim to learn logic rules from examples by a direct search in the space of logic programs. To scale to large knowledge graphs, recent work [Galárraga et al., 2013, Omran et al., 2018] learns predicate and entity embeddings to guide and prune the search. These methods typically rely on carefully hand-designed and task-specific templates in order to narrow down the hypothesis space. Their drawback is their difficulties with noisy data.

Differentiable ILP methods By framing an ILP task as a classification problem, these approaches based on a continuous relaxation of the logical reasoning process can learn via gradient descent. The learnable weights may be assigned to rules [Evans and Grefenstette, 2018]—although combinatorially less attractive—or to the membership of predicates in a clause [Payani and Fekri, 2019, Zimmer et al., 2021]. However, as ILP solvers, these models are hard to scale and have a constrained implementation, e.g., task-specific template-based variable assignments or limited number of predicates and objects. In another direction, NLM [Dong et al., 2019] learns rules as shallow MLPs; by doing so, it crucially loses in interpretability. In contrast to these methods, our model HRI ascribes embeddings to predicates, and the membership weights derive from a similarity score, which may be beneficial in rich and multi-task learning domains.

Multi-hop reasoning methods In multi-hop reasoning methods, developed around knowledge base (KB) completion tasks, predicate invention is understood as finding a relational path [Yang et al., 2017] or a combination of them [Yang and Song, 2020] on the underlying knowledge graph; this path amounts to a chain-like first-order rule. However, although computationally efficient, the restricted expressiveness of these methods limit their performances.

Embedding-based models In the context of KB completion or rule mining notably, many previous studies learn embeddings of both binary predicates and entities. Entities are typically attached to low-dimensional vectors, while relations are understood as bilinear or linear operators applied on entities possibly involving some non-linear operators [Bordes et al., 2013, Lin et al., 2015, Socher et al., Nickel et al., 2011, Guu et al., 2015]. These methods are either limited in their reasoning power or suffer from similar problems as multi-hop reasoning.

Our work is closely related to that of Campero et al. [2018] whose representation model is inspired by NTP [roc, 2017], attaching vector embeddings to predicates and rules. Their major drawback, like most classic or differentiable ILP methods, is the need for a carefully hand-designed template set for each ILP task. In contrast, we extend their model to a hierarchical structure with an expressive set of meta-rules, which can be used for various tasks. We further demonstrate our approach in the multi-
task setting.

3 Background

We first define first-order logic and inductive logic programming (ILP). Then, we recall some approaches for predicate invention, an essential aspect of ILP.

Notations Sets are denoted in calligraphic font (e.g., $C$). Constants (resp. variables) are denoted in lowercase (resp. uppercase). Predicates have the first letter of their names capitalized (e.g., $P$ or $Even$), their corresponding atoms are denoted sans serif (e.g., $P$), while predicate variables are denoted in roman font (e.g., $P$). Integer hyperparameters are denoted $n$ with a subscript (e.g., $n_L$).

3.1 Inductive Logic Programming

First-order logic (FOL) is a formal language that can be defined with a set of constants $C$, a set of predicates $P$, a set of functions, and a set of variables. Constants correspond to the objects of discourse, $n$-ary predicates can be seen as mappings from $C^n$ to the Boolean set $\mathbb{B} = \{\text{True}, \text{False}\}$, $n$-ary functions are mappings from $C^n$ to the set of constants $C$, and variables correspond to unspecified constants. As customary in most ILP work, we focus on a function-free fragment of FOL.

An atom is a predicate applied to a tuple of arguments, either constants or variables; it is called a ground atom if all its arguments are constants. Well-formed FOL formulas are defined recursively as combinations of atoms with logic connectives (e.g., negation, conjunction, disjunction, implication) and existential or universal quantifiers (e.g., $\exists, \forall$). Clauses are a special class of formulas, which can be written as a disjunction of literals, which are atoms or their negations. Horn clauses are clauses with at most one positive literal, while definite Horn clauses contain exactly one. In the context of ILP, definite clauses play an important role since they can be re-written as rules:

$$ H \leftarrow B_1 \land B_2 \land \cdots \land B_k $$

where $H$ is called the head atom and $B_i$’s the body atoms. The variables appearing in the head atom are instantiated with a universal quantifier, while the other variables in the body atoms are existentially quantified.

Inductive Logic Programming (ILP) aims at finding a set of rules such that all the positive examples and none of the negative ones of a target predicate are entailed by both these rules and some background knowledge expressed in FOL. The background knowledge may contain ground atoms, called facts, but also some rules.

Forward chaining can be used to chain rules together to deduce a target predicate valuation from some facts. Consider the Even-Succ task as an example. Given background facts \{Zero(0), Succ(0, 1), Succ(1, 2)\} and rules:

$$ Even(X) \leftarrow Zero(X) $$

$$ Even(X) \leftarrow Even(Y) \land Aux(Y, X) $$

$$ Aux(X, Y) \leftarrow Succ(X, Z) \land Succ(Z, Y), $$

we can easily deduce the facts \{Aux(0, 2), Even(2)\} by unifying the body atoms of the rules with the facts; repeating this process, we can, iteratively, infer all even numbers.

The predicates appearing in the background facts are called input predicates. Any other predicates, apart from the target predicate, are called auxiliary predicates. A predicate can be extensional, as defined by a set of ground atoms, or intensional, defined by a set of clauses.

3.2 Predicate Invention and Meta-Rules

One important part of solving an ILP problem is predicate invention, which consists in creating auxiliary predicates, intensionally defined, which would ultimately help define the target predicate. Without language bias, the set of possible rules to consider grows exponentially in the number of body atoms and in the maximum arity of the predicates allowed. In previous work, this bias is often enforced using meta-rules, also called rule templates, which restrict the hypothesis space by imposing syntactic constraints. The hypothesis space is generated by the successive applications of the meta-rules on the predicate symbols from the background knowledge.

\[ \text{This rule can be read as: if, for a certain grounding, all the body atoms are true, then the head atom is also true.} \]
A meta-rule corresponds to a second-order clause with predicate variables. For instance, the chain meta-rule is:

\[ H(X, Y) \leftarrow B_1(X, Z) \land B_2(Z, Y), \]  

(3)

where \( H, B_1, \) and \( B_2 \) correspond to predicate variables.

**LRI Model** We recall the differentiable approach to predicate invention proposed by [Campero et al., 2018]. Their model, Logical Rule Induction (LRI), learns an embedding for each predicate and each meta-rule atom; then, predicates and atoms of meta-rules are matched via a soft unification technique. For any predicate \( P \), let \( \theta_P \in \mathbb{R}^d \) denotes its \( d \)-dimensional embedding. The embeddings attached to a meta-rule like (3) with one head (H) and two body \((B_1, B_2)\) predicate variables are \((\theta_H, \theta_{B_1}, \theta_{B_2}) \in \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d\). The soft unification is based on cosine to measure the degree of similarity of embeddings of predicates and meta-rule atoms. For example, \( \alpha_{PH} = \cos(\theta_P, \theta_H) \in [0, 1] \) is the unification score between predicate \( P \) and head predicate variable \( H \); a higher unification score indicates a higher belief that \( P \) is the correct predicate for \( H \). Similarly, unification scores are computed for any pair of predicate and meta-rule atom.

In this model, a ground atom \( P = P(s, o) \) is represented as \((\theta_P, s, o, v_P)\), where \( \theta_P \) is the embedding of predicate \( P \), \( s \) and \( o \) are respectively the subject and object constants, and \( v_P \in [0, 1] \) is a valuation measuring the belief that \( P \) is true. In ILP tasks, the valuations are initialized to 1 for background facts and are otherwise set to 0, reflecting a closed-world assumption.

For better clarity, let us use meta-rule (3) as a running example to present how one inference step is performed. For an intensional predicate \( P \), the valuation of a corresponding ground atom \( P = P(x, y) \) with respect to a meta-rule \( \mathcal{R} \) of the form (3) and two ground atoms, \( P_1 = P_1(x, z) \) and \( P_2 = P_2(z, y) \), is computed as follows:

\[
v(P, \mathcal{R}, P_1, P_2) = (\alpha_{PH} \cdot \alpha_{P_1B_1} \cdot \alpha_{P_2B_2}) \cdot (v_{P_1} \cdot v_{P_2}).
\]  

(4)

The term in the first brackets measures how well the predicate tuple \((P, P_1, P_2)\) matches the meta-rule, while the term in the second brackets corresponds to a fuzzy AND. Note that (4), defined as a product of many terms in \([0, 1]\), leads to an underestimation issue. The valuation of atom \( P \) with respect to meta-rule \( \mathcal{R} \) is then computed as:

\[
v(P, \mathcal{R}) = \max_{P_1, P_2} v(P, \mathcal{R}, P_1, P_2).
\]  

(5)

Since \( P \) can be matched with the head of several meta-rules, the new valuation of \( P \) is, after one inference step:

\[
v(P) = \max_{\mathcal{R}_1} (v_{\text{old}}(P), \max_{\mathcal{R}_2} v(P, \mathcal{R}_2)).
\]  

(6)

where \( v_{\text{old}}(P) \) is the previous valuation of \( P \). The max over meta-rules allows an intensional predicate to be defined as a disjunction. After \( n \) steps of forward chaining, the model uses binary cross-entropy as loss function to measure the difference between inferred values and target values.

### 4 Hierarchical Rule Induction

Let us introduce our model HRI and our generic meta-rules set, alongside some theoretical results about its expressivity.

#### 4.1 Proposed Model

Building on LRI [Campero et al., 2018], our model includes several innovations, backed both by theoretical and experimental results: proto-rules, incremental prior, hierarchical prior, and improved model inference.

The priors reflect the hierarchical and progressive structure arguably present in the formation of human conceptual knowledge. HRI is illustrated in Figure 1. The boxes, spread in different layers, denote auxiliary predicates, which embody an asymmetrical disjunction of a conjunction of two atoms with a third body predicate. The arrows represent the soft unification between one body predicate and the heads of lower-level predicates, following (12).

**Proto-rules** We first define the notion of proto-rules, which implicitly correspond to sets of meta-rules. A \(^5\)In [Campero et al., 2018], \( d \) equals the number of predicates.
proto-rule is a meta-rule adaptive (see Appendix A for a formal definition). For instance, extended as a proto-rule could be written as:

\[ H(X, Y) \leftarrow \overline{B}_1(X, Z) \land \overline{B}_2(Z, Y), \]  

(7)

where the \( B_i \)'s correspond to predicate variables, of arity 2 or 1, where the second argument is optional. For instance, (7) can be instantiated as the following rule:

\[ P(X, Y) \leftarrow P_1(X) \land P_2(Z, Y) \]  

(8)

where \( P, P_2 \) are binary predicates, and \( P_1 \) is a unary predicate implicitly viewed as the binary predicate \( \overline{P}_1 \), where \( \forall z, \overline{P}_1(x, z) := P_1(x) \).

We propose a minimal and expressive set of proto-rules in Section 4.2. However, HRI can be instantiated with any set \( R \) of proto-rules. The choice of \( R \) defines the language bias used in the model.

**Incremental Prior** To reinforce the incremental aspect of predicate invention, in contrast to LRI, each auxiliary predicate is directly associated with a unique proto-rule defining it. This has two advantages. First, it partially addresses the underestimation issue of (4) since it amounts to setting \( \alpha_P H = 1 \). Second, it reduces computational costs since the soft unification for a rule can be computed only by matching the body atoms in order to preserve expressivity, we can re-introduce disjunctions in the model by considering proto-rules with disjunctions in their bodies (see Section 4.2): e.g., (7) could be extended to:

\[ H(X, Y) \leftarrow (\overline{B}_1(X, Z) \land \overline{B}_2(Z, Y)) \lor \overline{B}_3(X, Y). \]  

(9)

**Hierarchical Prior** A hierarchical architecture is enforced by organizing auxiliary predicates in successive layers, from layer 1 to the max layer \( n_L \), as illustrated in Figure 1. Layer 0 contains all input predicates. For each layer \( l = 1, \ldots, n_L \), each proto-rule in \( R \) generates one auxiliary predicate.

Lastly, the target predicate is matched with the auxiliary predicates in layer \( n_L \) only.

With this hierarchical architecture, an auxiliary predicate \( P \) at layer \( \ell \) can be only defined from a set \( P_{<\ell} \) of predicates at a layer lower or equal to \( \ell \). This set can be defined in several ways depending on whether recursivity is allowed. We consider three cases: no recursivity, iso-recursivity, and full recursivity. For the no recursivity (resp. full recursivity) case, \( P_{<\ell} \) is defined as the set \( P_{<\ell} \) (resp. \( P_{\leq\ell} \)) of predicates at layer strictly lower than (resp. lower or equal to) \( \ell \). For iso-recursivity, \( P_{<\ell} \) is defined as \( P_{<\ell} \cup \{ P \} \).

Enforcing this hierarchical prior has two benefits. First, it imposes a stronger language bias, which facilitates learning. Second, it also reduces computational costs since the soft unification does not need to consider all predicates.

---

- **It amounts to embed some lower-arity predicates as higher-arity predicates.**
- **The max over meta-rules in (6) is not needed anymore.**
- **Since identifying intensional predicates to proto-rules prevent them to have disjunctive definitions.**
- **Alternatively, several auxiliary predicates could be generated per proto-rules. For simplicity, we only generate one and control the model expressivity with only one hyperparameter \( n_L \).**
Improved Model Inference  We improve LRI’s inference technique with a soft unification computation that reduces the underestimation issue in (4), which is due to the product of many values in [0, 1]. For clarity’s sake, we illustrate the inference with proto-rule (9); although the equations can be straightforwardly adapted to any proto-rule. One inference step in our model is formulated as follows:[10]

\[
\begin{aligned}
v_{\text{and}} &= \text{POOL}_{p_1, p_2} (\alpha_{p_1 B_1} \cdot \alpha_{p_2 B_2} \cdot \text{AND}[v_{p_1}, v_{p_2}]) \\
v_{\text{or}} &= \text{OR}[v_{\text{and}}, \text{POOL}_{p_3} (\alpha_{p_3 B_3} \cdot v_{p_3})] \\
v &= \text{MERGE}(v_{\text{old}}, v_{\text{or}}),
\end{aligned}
\]

where \(v\) (resp. \(v_{\text{old}}\)) denotes the new (resp. old) valuation of a grounded auxiliary predicate \(P\). For an auxiliary predicate at layer \(\ell\), the POOL operation is performed over both predicates \(P_1, P_2, P_3 \in \mathcal{P}_\ell\) and the groundings that are compatible to \(P\). However, before pooling, in presence of an existential quantifier as in proto-rule (9), a max is applied to the corresponding dimension of the tensor. The valuation \(v'\) of the target predicate (with variable denoted \(P'\)) is computed as a \(\text{MERGE}\) of its previous valuation \(v'_{\text{old}}\) and a \(\text{POOL}\) over atoms at layer \(n_L\):

\[
v' = \text{MERGE}(v'_{\text{old}}, \text{POOL}_{p_1} (\alpha_{p_1 p'} \cdot v_{p_1})).
\]

The underestimation issue is alleviated in two ways. First, the \(\text{POOL}\) operation is implemented as a sum, \(\text{AND}\) as \(\min\), \(\text{OR}\) as \(\max\), and \(\text{MERGE}\) as \(\max\). Second, the unification scores \(\alpha_{p_1 B_1}^{s}\)’s based on cosine are renormalized with a softmax transformation:

\[
\alpha_{p_1 B_1} = \frac{\exp(\cos(\theta_{p_1}, \theta_{B_1})/\tau)}{\sum_{p_j} \exp(\cos(\theta_{p_j}, \theta_{B_1})/\tau)},
\]

where \(p_j \in \mathcal{P}_\ell\) and hyperparameter \(\tau\), called temperature, controls the renormalization. Score \(\alpha_{p_j p'}\) can be computed in a similar way with embedding \(\theta_{p'}\). These choices have been further experimentally validated (see Appendix B).

Figure 2: Three types of meta-rules: boxes represent variables, green arrows represent relations for head atoms and blue arrows represent relations for body atoms

4.2 Generic Set of Proto-Rules

Although HRI is generic, we design a small set of expressive proto-rules to instantiate our model. We introduce first the following set of proto-rules \(\mathcal{R}_0\) (see Figure 2):

\[
\begin{aligned}
\mathcal{A} : H(X) &\leftarrow B_1(X, Y) \land B_2(Y, X) \\
\mathcal{B} : H(X, Y) &\leftarrow B_1(X, Z) \land B_2(Z, Y) \\
\mathcal{C} : H(X, Y) &\leftarrow B_1(X, Y) \land B_2(Y, X)
\end{aligned}
\]

where the \(B_i\)’s correspond to predicate variables where the second argument is optional (see Section 4.1).

To support our design, we analyzed the expressivity of \(\mathcal{R}_0\), by investigating which fragment of first-order logic can be generated by \(\mathcal{R}_0\) from a set of predicates \(\mathcal{P}\). Because of space constraints, we only present here our main result, where we assume that \(\mathcal{P}\) contains the zero-ary predicate True and the equality symbol:

Theorem 4.1. The hypothesis space generated by \(\mathcal{R}_0\) from \(\mathcal{P}\) is exactly the set of function-free definite Horn clause fragment \(\mathcal{F}_{\mathcal{P}, \leq 2}\) composed of clauses with at most two body atoms involving unary and binary predicates in \(\mathcal{P}\).
We refer the reader to Appendix A for all formal definitions, proofs and further theoretical results. This result implies that our model with \( \mathcal{R}_0 \) can potentially solve perfectly any ILP task whose target predicate is expressible in \( \mathcal{F}_{\mathcal{P} \leq 2} \) with a large enough max layer \( n_L \).

However, to potentially reduce the max layer \( n_L \) and facilitate the definition of recursive predicates, we extend \( \mathcal{R}_0 \) to the set \( \mathcal{R}_0' \) by including a disjunction with a third atom in all the proto-rules:

\[
\begin{align*}
\mathcal{A}_* & : H(X) \leftarrow (\mathcal{B}_1(X, Y) \land \mathcal{B}_2(Y, X)) \lor \mathcal{B}_3(X, T) \\
\mathcal{B}_* & : H(X, Y) \leftarrow (\mathcal{B}_1(X, Z) \land \mathcal{B}_2(Z, Y)) \lor \mathcal{B}_3(X, Y) \\
\mathcal{C}_* & : H(Y, X) \leftarrow (\mathcal{B}_1(X, Y) \land \mathcal{B}_2(Y, X)) \lor \mathcal{B}_3(X, Y)
\end{align*}
\]

Moreover, as we have observed a certain redundancy to be beneficial to the learning, we incorporate the permutation rule \( \mathcal{I} \) in our experiments:

\[
\mathcal{R}_* = \mathcal{R}_0' \cup \{ \mathcal{I} : H(X, Y) \leftarrow \mathcal{F}(Y, X) \}
\]

This small set of protorules \( \mathcal{R}_* \), which has the same expressivity as \( \mathcal{R}_0 \) characterized in Theorem 4.1, is used to instantiate our model in all our experiments.

5 Proposed Training Method

Training We train our model with a fixed number \( n_T \) of training iterations. For each iteration, the model is trained using one training instance. During each iteration, we add a geometrically-decaying Gaussian-distributed noise to the embeddings for both predicates and rules to avoid local optima, similarly to LRI [Campero et al., 2018]. Then the improved inference procedure described in (10) is performed with these noisy embeddings for a number \( n_I \) of steps to get the valuations for all predicates. After these \( n_I \) inference steps, the binary cross entropy (BCE) loss of the ground-truth target predicate valuation and the learned target predicate valuation is calculated:

\[
\sum_{x, y} -G^t_{x,y} \log \left( v(P^t_{x,y}) \right) - (1-G^t_{x,y}) \log \left( 1-v(P^t_{x,y}) \right),
\]

where the sum is over all pairs of objects \( x, y \) in one training instance, \( G^t_{x,y} \in \{0, 1\} \) is the ground truth value of atom \( P^t_{x,y} = P^t(x, y) \). To encourage the unification scores to be closer to 0 or 1, an extra regularisation term is added to the BCE loss to update the embeddings:

\[
+ \lambda \sum_{P, Q} \alpha_{PQ} (1 - \alpha_{PQ}).
\]

where hyperparameter \( \lambda \) controls the regularization weight and the sum is over all predicates \( P \) that can be matched with a predicate variable \( Q \) appearing in some meta-rules.

In addition, to further help with convergence to an interpretable solution and avoid local optima, we replace during training the softmax transformation by a variant of Gumbel-softmax [Jang et al., 2017, Maddison et al., 2017]. Commonly, it amounts to injecting a Gumbel noise \( G_j \) in the softmax, i.e., in (12), we replace each \( \cos(\theta_j \cdot \theta_i) \) by \( \cos(\theta_j \cdot \theta_i) + G_j \) where Gumbel noise \( G_j = -\log(-\log(U_j)) \) is obtained by sampling \( U_j \) according to a uniform distribution. However, we rely on a variant of this noise: \( \tilde{G}_j = -\log(-\log(U_j)) \), which is related as \( \tilde{G}_j = \frac{\log(-\log(U_j))}{2} G_j \). This choice has been experimentally favored. We discuss it further in the Appendix. Gumbel scale \( g \in (0, +\infty) \) is linearly decreased during training. In contrast to the low-level parameter-noise applied directly to the embeddings, the Gumbel noise may be understood as a higher-level noise, enacting on the similarity coefficients themselves.

Convergence to an Interpretable Model The passage from our soft model to a symbolic model may be realized by taking the limit of the softmax temperature in the unification score to zero, or equivalently, switching to an argmax in the unification score; i.e., for each proto-rule, the final learned rules can be interpreted by assigning head and body atoms to the predicates that obtain the highest unification score.

For instance, at a layer \( \ell \), the extracted symbolic rule \( \mathcal{R} \) could be extracted from proto-rule \( \mathcal{R}_0 \) if \( P \) is the auxiliary predicate associated to that proto-rule at layer \( \ell \) and

\[
\begin{align*}
P_1 &= \arg \max_{P \in \mathcal{P}_1} \alpha_{PB_1} \\
P_2 &= \arg \max_{P \in \mathcal{P}_2} \alpha_{PB_2}
\end{align*}
\]

Then we can interpret the solution by successively unfolding the logical formula extracted for each involved predicate, starting from the target predicate.
Table 1: Percentage of successful runs among 10 runs. |I| is the smallest number of intensional predicates needed. Recursive means whether or not the solution needs to learn recursive rules.

### 6 Experimental Results

We evaluated our model on multiple domains and compared with relevant baselines. In the main paper, for space constraints, we only present two sets of experiments. First, on standard ILP tasks, we show that our model is competitive against other neuro-symbolic methods in terms of performance, but also computational/space costs. Second, on a large domain from Visual Genome [Krishna et al., 2017], we show that our model can scale to a large multi-task problem and is superior to NLIL [Yang and Song, 2020].

Our results are averaged over several runs with different random seeds. In each run, the model is trained over a fixed number of iterations. Hyper-parameter details are given in Appendix B and additional experiments (e.g., choice of operators, limitations of LRI, reinforcement learning) are discussed in Appendices B and C.

For simplicity and consistency, we instantiate our model with $R^*$, with full recursivity in all the tasks except Visual Genome (no recursivity).

Note the performances can be improved and/or learning can be accelerated if we customize $R_*$ to a task or restrict the recursivity. However, we intended to emphasize the genericity of our approach.

#### 6.1 ILP Tasks

We evaluated our model on the ILP tasks from [Evans and Grefenstette, 2018], which are detailed in Appendix B, where we also provide the interpretable solutions found by our model.

We compare our model with $\partial$ILP [Evans and Grefenstette, 2018] and LRI [Campero et al., 2018] using their reported results. In contrast to LRI or $\partial$ILP, which use a specific set of meta-rules tailored for each task, we use the same generic and more expressive template set $R_*$ to tackle all tasks; naturally, this generic approach impedes the learning for a few tasks.

A selection of the results on these ILP tasks are presented in Table 1 (see Appendix B). A run is counted as successful if the mean square error between inferred and given target valuations is less than 1e-4 for new evaluation datasets. Column train corresponds to the performance measured during training, while Column soft evaluation resp. symbolic evaluation to evaluation performance (no
noise), using\(^{11}\) resp. the interpretable solution (see end of Section\(^5\)).

We also ran our model on two other ILP tasks used by NLM [Dong et al., 2019] and DLM [Zimmer et al., 2021] to compare their performances and training times for a varying number of training constants. Those two ILP tasks are similar to those used in [Evans and Grefenstette, 2018], but with different background predicates. The results reported in Table 2 show that our model is one to two orders of magnitude faster and that it scales much better in terms of training constants, while achieving at least as good performances.

### 6.2 Visual Genome

Our model can be applied beyond classical ILP domains, and even benefit from richer environments and semantic structure. Furthermore, initial predicate embeddings can be bootstrapped by visual or semantic priors. We illustrate it by applying it to the larger dataset of Visual Genome [Krishna et al., 2017], or more precisely a pre-processed (less noisy) version known as the GQA [Hudson and Manning, 2019a]. Similarly to Yang and Song [2020], we filtered out the predicates with less than 1500 occurrences, which lead to a KB with 213 predicates; then, we solved ILP-like task to learn the explanations for the top 150 objects in the dataset. More details on task, results, baseline and metrics used are provided in Appendix\(^C\).

We ran a first set of experiments to validate that embeddings priors may be leveraged and compared three initialization methods: random initialization, pretrained embeddings from NLIL [Yang and Song, 2020], and pretrained embeddings from NLP model GPT2. We trained our model on a single ILP task, which consists in predicting predicate \textit{Car}. For this task, we further filtered the GQA dataset to keep 185 instances containing cars. Table 2 presents the accuracy, precision, and recall, obtained over 10 runs. Pretrained embeddings outperform the randomly initialized ones in all metrics. Embeddings from NLIL, trained specifically on this dataset, expectedly yield the best results. Although, to be fair, for the next experiments, we rely on the NLP embeddings.

In multitask setting, we trained 150 target models together with shared background predicate embeddings, where each model corresponds to one object classification. The training consists of \(N_r\) rounds. In each round, we trained each model sequentially, by sampling for each target \(n_p\) instances containing positive examples, and \(n_r\) with or without positive examples.

We compared our model with NLIL [Yang and Song, 2020] and two other supervised baselines (classifiers) mentioned in their paper: Freq, which predicts object class by checking the related relation that contains the target with the highest frequency; and MLP-RCNN, a MLP classifier trained with RCNN features extracted from object images in Visual Genome dataset. The latter is a strong baseline because it uses visual information for its predictions while the other methods only use relational information. Like NLIL, we use R@1 and R@5 to evaluate the trained models. Table 3 summarizes the results. We see that both our method and MLP+RCNN achieve best performance for R@1. For R@5, we outperform Freq and NLIL. Table 5 shows some extracted learned rules for the multitask model.

| Task            | #Training constants | % successful runs | Training time (secs) | Model        | Visual Genome |
|-----------------|---------------------|-------------------|----------------------|--------------|---------------|
| Adjacent to Red | 7                   | 100 90 100        | 163 920 62           | NLM          | R@1 0.53 R@5 0.81 |
|                 | 10                  | 90 90 100        | 334 6629 71          | DLM          |               |
|                 | 20                  | 100 100 100      | 1441 3331 89         | Ours         |               |

Table 2: Comparisons with NLM/DLM in terms of percentage of successful runs and average training times over 10 runs.

| Grandparent     | 9                   | 90 100 100        | 402 2047 79          | NLM          | R@1 0.51 R@5 0.52 |
|                 | 20                  | 100 100 100      | 1441 3331 89         | DLM          |               |
|                 | 20                  | 100 100 100      | 1441 3331 89         | Ours         |               |

Table 3: R@1 and R@5 for 150 objects classification on VG.
### Table 4: Performance of different embedding initializations for single Visual Genome task.

| Initialization | Accuracy | Precision | R@1 |
|----------------|----------|-----------|-----|
|                | soft evaluation | symbolic evaluation | soft evaluation | symbolic evaluation | soft evaluation | symbolic evaluation |
| Random         | 0.63     | 0.49      | 0.57 | 0.5       | 0.23 | 0.38 |
| NLIL           | 0.75     | 0.6       | 0.87 | 0.75      | 0.46 | 0.58 |
| GPT2           | 0.65     | 0.45      | 0.72 | 0.66      | 0.27 | 0.5  |

Table 5: Examples of extracted rules from multi-task model

| Target   | Rules                                                                 |
|----------|-----------------------------------------------------------------------|
| wrist    | wrist(X) ← (watch(Y) ∧ on(Y, X)) ∨ False                             |
| person   | person(X) ← (on(X, Y) ∧ bench(Y)) ∨ wearing(X, Y)                     |
| vase     | vase(X) ← (aux(X, Y) ∧ flowers(Y)) ∨ False                           |
|          | aux(X, Y) ← (True ∧ on(Y, X)) ∨ with(X, Y)                            |
| logo     | logo(X) ← (aux0(X) ∧ True) ∨ aux1(X)                                 |
|          | aux1(X) ← (on(X, Y) ∧ laptop(Y)) ∨ False                             |
|          | aux0(X) ← (on(X, Y) ∧ kite(Y)) ∨ False                               |

7 Ethical Considerations

Regarding our last experiment choice, let us point out that the use of data coming from annotated images, as well as the use of pretrained embeddings produced by the NLP model GPT2 should be regarded with circumspection. As it as been well documented [??], such data holds strong biases, which would raise ethical issues when deployed in the real-world. Our motive was merely to illustrate the performance of our model in more noisy domains typical of real-world scenarios (such as autonomous driving scenarios). We strongly advise against the use of such unbiased data for any real-world deployment.

8 Conclusion

We presented a new neuro-symbolic interpretable model performing hierarchical rule induction through soft unification with learned embeddings; it is initialised by a theoretically supported small-yet-expressive set of proto-rules, which is sufficient to tackle many classical ILP benchmark tasks, as it encompasses a consequent function-free definite Horn clause fragment. Our model has demonstrated its efficiency and performance in both ILP, reinforcement learning (RL) and richer domains against state-of-the-art baselines, where it is typically one to two orders of magnitude faster to train.

As future work, we plan to apply it to broader RL domains (e.g., inducing game rules from a game trace) but also to continual learning scenarios where an interpretable logic-oriented higher-level policy would be particularly pertinent (e.g., autonomous driving). Indeed, we postulate that our model may be suited for continual learning in semantically-richer domains, for multiple reasons detailed in Appendix E alongside current limitations of our approach, which could be further investigated upon.

References

End-to-end differentiable proving. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 3788–3800, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/b2ab001909a8a6f04b51920306046ce5-Abstract.html

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In NeurIPS, 2013.

A. Campero, A. Pareja, T. Klinger, J. Tenenbaum, and S. Riedel. Logical rule induction and theory learning using neural theorem proving. arXiv preprint arXiv:1809.02193, 2018.

A. Cropper and S. Dumančić. Inductive logic programming at 30: a new introduction. Machine Learning, 43, 2021. ISSN 1573-0565. doi: 10.1007/s10994-021-
variables. In ICLR, 2017. URL https://openreview.net/forum?id=S1jE5L5gL.

R. Manhaeve, S. Dumančić, A. Kimmig, T. Demeester, and L. De Raedt. Deepproblog: Neural probabilistic logic programming. In Neural Information Processing Systems, 2018.

G. Marcus. Deep learning: A critical appraisal. arXiv: 1801.00631, 2018.

S. Muggleton. Inverse entailment and progol. New Generation Computing, 13:245–286.

S. Muggleton. Inverse entailment and progol, 1995. URL https://doi.org/10.1007/BF03037227

M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In ICML, 2011.

P. Omran, K. Wang, and Z. Wang. Scalable rule learning via learning representation. pages 2149–2155, 07 2018. doi: 10.24963/ijcai.2018/297.

A. Payani and F. Fekri. Learning algorithms via neural logic networks. arXiv:1904.01554, 2019.

J. R. Quinlan. Learning logical definitions from relations. MACHINE LEARNING, 5:239–266, 1990.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multi-task learners. 2019.

J. A. Robinson. A machine-oriented logic based on the resolution principle. J. ACM, 12(1):23–41, Jan. 1965. ISSN 0004-5411. doi: 10.1145/321250.321253. URL https://doi.org/10.1145/321250.321253

A. Santoro, D. Raposo, D. G. T. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. 2017.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. arXiv preprint: 1707.06347, Aug. 2017.

R. Socher, D. Chen, C. D. Manning, and A. Y. Ng. Reasoning with neural tensor networks for knowledge base completion.

F. Yang, Z. Yang, and W. W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. In Neural Information Processing Systems, 2017.

Y. Yang and L. Song. Learn to explain efficiently via neural logic inductive learning. In International Conference on Learning Representations, 2020.

R. Zellers, M. Yatskar, S. Thomson, and Y. Choi. Neural motifs: Scene graph parsing with global context. In CVPR, 2018. URL http://arxiv.org/abs/1711.06640.

M. Zimmer, X. Feng, C. Glanois, Z. Jiang, J. Zhang, P. Weng, L. Dong, H. Jianye, and L. Wulong. Differentiable logic machines. arXiv preprint arXiv:2102.11529, 2021.
Appendix

A Expressivity Analysis

Many methods within inductive logic programming (ILP) define meta-rules, i.e., second-order Horn clauses which delineates the structure of learnable programs. The choice of these meta-rules, referred to as a language bias and restricting the hypothesis space, results in a well-known trade-off between efficiency and expressivity. In this section, we bring some theoretical justification for our designed set of second-order Horn Clause, by characterizing its expressivity. Relying on minimal sets of metarules for rule induction models has been shown to improve both learning time and predictive accuracies [Cropper and Muggleton, 2014; Fonseca et al., 2004]. For a model to be both adaptive and efficient, it seems pertinent to aim for a minimal set of meta-rules generating a sufficiently expressive subset of Horn clauses. The desired expressivity is ineluctably contingent to the tackled domain(s); a commonly-considered fragment is Datalog which has been proven expressive enough for many problems. Datalog is a syntactic subset of Prolog which, by losing the Turing-completeness of Prolog/Horn, provides the undeniable advantage that its computations always terminate. Below, we focus our attention to similar fragments, in concordance with previous literature [Galárraga et al., 2013; Cropper and Tourret, 2020].

To make the document self-contained, before stating our results, let us introduce both concepts and notations related to First and Second Order Logic. Recall $\mathcal{P}$ denote the set of all considered predicates.

Horn Logic We focus on Horn clause logic, since it is a widely adopted Turing-complete subset of FOL. A Horn clause is a clause with at most one positive literal. Horn clauses may be of the following types (where $P$, $Q$, $T$, and $U$ are atoms):

- **goal clauses** which have no positive literal; they are expressed as $\neg P \lor \neg Q \lor ... \lor \neg T$ or equivalently $\text{False} \leftarrow P \land Q \land ... \land T$,

- **definite clauses** which have exactly one positive literal; they are expressed as $\neg P \lor \neg Q \lor ... \lor \neg T \lor U$ or equivalently as a Horn rule $U \leftarrow P \land Q \land ... \land T$,

- **facts** which can be expressed as $U \leftarrow \text{True}$ where $U$ is grounded.

Meta-rules We follow the terminology from the Meta Interpretative Learning literature [Cropper and Tourret, 2020]:

**Definition 8.1.** A **meta-rule** is a second-order Horn clause of the form: $A_0 \leftarrow A_1 \land ... \land A_m$ where each $A_i$ is a literal of the form $P(T_1, ..., T_{n_i})$ where $P$ is a predicate symbol or a second-order variable that can be substituted by a predicate symbol, and each $T_i$ is either a constant symbol or a first-order variable that can be substituted by a constant symbol.

Here are some examples of intuitive meta-rules, which will be of later use:

$$(\sigma) \ P(A, B) \leftarrow Q(B, A) \quad \text{permute}$$

$$(\iota) \ P(A, B) \leftarrow Q(A) \quad \text{expand}$$

$$(\exists) \ P(A) \leftarrow \exists B, Q(A, B) \quad \text{existential contraction}$$

$$(\forall) \ P(A) \leftarrow \forall B, Q(A, B) \quad \text{universal contraction}$$

$$(\forall) \ P(A) \leftarrow \exists B, Q(A, B) \quad \text{diagonal extract}$$

$$(\Delta) \ P(A, A) \leftarrow Q(A) \quad \text{diagonal fill}$$

---

11 Pure Prolog programs are composed by definite clauses and any query in Prolog is a goal clause.
12 Horn clause logic and universal Turing machines are equivalent in terms of computational power.
13 Facts can be seen as a sub-case of definite clause assuming $\text{True} \in \mathcal{P}$. 

13
A common meta-rule set, which has been used and tested in the literature [Cropper and Tourret 2020], is the following:

\[
\mathcal{R}_{MIL} = \left\{ \begin{array}{c}
(\text{Indent}_1) & \mathcal{P}(A) \leftarrow \mathcal{Q}(A) \\
(\text{Indent}_1) & \mathcal{P}(A) \leftarrow \mathcal{Q}(A) \land \mathcal{R}(A) \\
(\text{Indent}_2) & \mathcal{P}(A, B) \leftarrow \mathcal{Q}(A, B) \\
(\text{Precon}) & \mathcal{P}(A, B) \leftarrow \mathcal{Q}(A) \land \mathcal{R}(A, B) \\
(\text{Postcon}) & \mathcal{P}(A, B) \leftarrow \mathcal{Q}(A, B) \land \mathcal{R}(B) \\
(\text{Curry}) & \mathcal{P}(A, B) \leftarrow \mathcal{Q}(A, B, R) \\
(\text{Chain}) & \mathcal{P}(A, B) \leftarrow \mathcal{Q}(A, C) \land \mathcal{R}(C, B)
\end{array} \right\}
\] (17)

where the letters \( \mathcal{P}, \mathcal{Q}, \mathcal{R} \) correspond to existentially-quantified second-order variables and the letters \( \mathcal{A}, \mathcal{B}, \mathcal{C} \) to universally-quantified first-order variables.

**Proto-rules** Our model relies on an extended notion of meta-rules, which we refer to as proto-rules. These templates have the specificity that they do not fix the arity of their body predicates; only the arity of the head predicate is determined. Formally, they can be defined as follows. For any \( n \in \mathbb{N} \), let \( \mathcal{P}^n \) (resp. \( \mathcal{P}^{\leq n} \)) be the predicates in \( \mathcal{P} \) that have arity equal to (resp. lower or equal to) \( n \).

Define the projection operator \( \nu^n \) which canonically embeds predicates of arity \( i \leq n \) in the space of predicates of arity \( n \):

\[
\nu^n : \mathcal{P}^{\leq n} \rightarrow \mathcal{P}^n \quad \text{s.t.} \quad \nu^n(P)(X_1, \ldots, X_n) = P(X_1, \ldots, X_i) \text{ for } P \in \mathcal{P}^i.
\] (18)

Note that the restriction of \( \nu^n \) on the space of predicates of arity \( n \) is identity: \( \nu^n \mid_{\mathcal{P}^n} = \text{id}_{\mathcal{P}^n} \). Moreover, this projection naturally extends to second-order variables. To ease the notations, we denote below the projection \( \nu^{n_i} \) with an overline (e.g., for a predicate \( P \), \( \nu^{n_i}(P) = \overline{P} \)), since there is no risk of confusion because \( n_i \) is specified by its arguments \( (T_1, \ldots, T_{n_i}) \).

**Definition 8.2.** A proto-rule is an extension of a second-order Horn clause of the form: \( A_0 \leftarrow A_1 \land \cdots \land A_m \), where \( A_0 \) (resp. each \( A_i, i > 0 \)) is a literal of the form \( P(T_1, \ldots, T_{n_i}) \) (resp. \( \nu^{n_i}(P)(T_1, \ldots, T_{n_i}) \)) in which \( P \), which is a predicate symbol or a second-order variable that can be substituted by a predicate symbol, is of arity lower or equal to \( n_i \), and each \( T_i \) is either a constant symbol or a first-order variable that can be substituted by a constant symbol.

Seeing second-order rules as functions over predicate spaces to first-order logic rules, we can provide another characterization of proto-rules: a meta-rule can be understood as a mapping \( \mathcal{P}^{n_1} \times \cdots \times \mathcal{P}^{n_m} \rightarrow \mathcal{H} \) for some indices \( n_i \), i.e., taking \( m \) predicates of specific arities and returning a first-order Horn clause; in contrast, a proto-rule is a mapping \( \mathcal{P}^{\leq n_1} \times \cdots \times \mathcal{P}^{\leq n_m} \rightarrow \mathcal{H} \), for some indices \( n_i \).

The first set of protorules we propose is the following:

\[
\mathcal{R}_0 := \left\{ \begin{array}{c}
\mathfrak{A} : & \mathcal{P}(A) \leftarrow \overline{\mathcal{Q}(A, B) \land \mathcal{R}(B, A)} \\
\mathfrak{B} : & \mathcal{P}(A, B) \leftarrow \overline{\mathcal{Q}(A, C) \land \mathcal{R}(C, B)} \\
\mathfrak{C} : & \mathcal{P}(A, B) \leftarrow \overline{\mathcal{Q}(A, B) \land \mathcal{R}(B, A)}
\end{array} \right\}
\] (19)

**Horn Fragments** A *Horn theory* \( \mathcal{T} \) is a set of Horn clauses.

Let us denote the set of second order Horn clause, the set of definite Horn clauses, and the set of function-free definite Horn clauses respectively by \( \mathcal{H}, \mathcal{S}, \mathcal{F} \). Within Horn clause logic, several restrictions have been proposed in the literature to narrow the hypothesis space, under the name of *fragment*. A fragment is a syntactically restricted subset of a theory. Following previous work (such as [Cropper and Tourret 2020]), we introduce below a list of classic fragments of \( \mathcal{F} \), in decreasing order:
• **connected** fragment, $C$: connected definite function-free clauses, i.e., clauses in $F$ whose literals can not be partitioned into two sets such that the variables attached to one set are disjoint from the variables attached to literals in the other set; e.g., $P(A, B) \leftarrow Q(A, C) \land R(A, B)$ is connected while the following clause is not $P(A, B) \leftarrow Q(A, C) \land R(B)$.

• **Datalog**$^{14}$ fragment, $D$: sub-fragment of $C$, composed of clauses such that every variable present in the head of the Horn rule also appears in its body.$^{15}$

• **two-connected** fragment, $K$: sub-fragment of $D$ composed of clauses such that the variables appearing in the body must appear at least twice.

• **exactly-two-connected** fragment, $E$: sub-fragment of $K$ such that the variables appearing in the body must appear exactly twice.

• **duplicate-free** fragment, $U$: sub-fragment of $K$ excluding clauses in which a literal contains multiple occurrences of the same variable, e.g., excluding $P(A, A) \leftarrow Q(A)$.

These fragments are related as follows:

$$ \mathcal{H} \supset \mathcal{S} \supset \mathcal{F} \supset \mathcal{C} \supset \mathcal{D} \supset \mathcal{K} \supset \mathcal{E} \supset \mathcal{U} \quad (20) $$

For $\mathcal{M}$ a set of meta-rules, we denote $\mathcal{M}^{\leq a}_{\leq m}$ the fragment of $\mathcal{M}$ where each literal has arity at most $a$ and each meta-rule has at most $m$ body literals. We extend the notation in order to allow both $a$ and $m$ to be a set of integers; e.g., $\mathcal{M}_{\{1,2\}}$ is the fragment of $\mathcal{M}$ where each literal corresponds to an unary or binary predicate and the body of each meta-rule contains exactly three literals.

**Binary Resolution**  As it would be of future use, let us introduce some notions around resolution. To gain intuition about the notion of resolved$^{16}$, let us look at an example before stating more formal definitions. Consider the following Horn clauses:

$$ C_1 : \quad P(X, Y) \lor \neg Q(X) \lor \neg R(X, Y) \quad \text{or, equivalently,} \quad P(X, Y) \leftarrow Q(X) \land R(X, Y) $$

$$ C_2 : \quad R(X, a) \lor \neg T(X, a) \lor \neg S(a) \quad \text{or, equivalently,} \quad R(X, a) \leftarrow S(a) \land T(X, a) $$

The atoms $R(X, Y)$ and $R(X, a)$ are unifiable, under the substitution $\sigma : \{a/Y\}$; under this substitution, we obtain the following clauses:

$$ \{P(X, a) \lor \neg Q(X) \lor \neg R(X, a), R(X, a) \lor \neg T(X, a) \lor \neg S(a)\} $$

Since either $R(X, a)$ is True, or $\neg R(X, a)$ is True, assuming $C_1$ and $C_2$ are True, we can deduce the following clause is True:

$$ (P(X, a) \lor \neg Q(X)) \lor (\neg T(X, a) \lor \neg S(a)) \quad \text{or, equivalently,} \quad P(X, a) \leftarrow Q(X) \land T(X, a) \land S(a) $$

$^{14}$Some wider Datalog fragment have been proposed, notably allowing negation in the body, under certain stratification constraints; here we consider only its intersection with function-free definite Horn clauses.

$^{15}$Since below each clause is assume to be a definite Horn clause, it can be equivalently represented as a Horn rule, with one head and several non-negated literals in the body; we below alternately switch between these representations.

$^{16}$Another way to gain intuition, is to think about propositional logic; there, a resolvent of two parent clauses containing complementary literals, such as $P$ and $\neg P$, is simply obtained by taking the disjunction of these clauses after removing these complementary literals. In the case of FOL or HOL, such resolvent may involve substitutions.
This resulting clause, is a binary resolvent of $C_1$ and $C_2$.

Now, let us state more formal definitions. We denote $\sigma = \{t_1/X_1, \cdots, t_n/X_n\}$, a substitution and $E \sigma$ the expression obtained from an expression $E$ by simultaneously replacing all occurrences of the variables $X_i$ by the terms $t_i$.

**Definition 8.3.** • A substitution $\sigma$ is called a unifier of a given set of expressions $\{E_1, \cdots, E_m\}$, $m \geq 2$ if $E_1 \sigma = \cdots = E_m \sigma$.

• A unifier $\theta$ of a unifiable set of expressions $E = \{E_1, \cdots, E_m\}$, $m \geq 2$, is said to be a most general unifier if for each unifier $\sigma$ of $E$ there exists a substitution $\lambda$ such that $\sigma = \theta \lambda$.

**Definition 8.4.** Consider two parent clauses $C_1$ and $C_2$ containing the literals $l_1$ and $l_2$ respectively. If $l_1$ and $\neg l_2$ have a most general unifier, then the clause $(C_1 \sigma \setminus l_2 \sigma) \lor (C_2 \sigma \setminus l_2 \sigma)$ is called a binary resolvent of $C_1$ and $C_2$.

**Derivation Reduction** Diverse reductions methods have been proposed in the literature, such as subsumption [Robinson 1965], entailment [Muggleton], and derivation [Cropper and Tourret 2018]. As previously pointed out (e.g., [Cropper and Tourret 2020]), common entailment methods may be too strong and remove useful clauses enabling to make predicates more specific. The relevant notion to ensure the reduction would not affect the span of our hypothesis space is the one of derivation-reduction ($D$-reduction) [Cropper and Tourret 2018], as defined in Definition 8.5.

Let us first define for the following operator for a Horn theory $T$:

$$
\begin{align*}
R_0(T) &= T \\
R_n(T) &= \{C \mid C \text{ is the binary resolvent of } C_1 \text{ and } C_2 \text{ with } C_1 \in R_{n-1}(T), C_2 \in T \}
\end{align*}
$$

**Definition 8.5.** The Horn closure of a Horn theory $T$ is: $R^*(T) = \cup_n R_n(T)$.

**Definition 8.6.** (i) A Horn clause $C$ is derivationally redundant in the Horn theory $T$ if $T \vdash C$, i.e., $C \in R^*(T)$; it is said to be $k$-derivable from $T$ if $C \in R^k(T)$.

(ii) A Horn theory $T$ is derivationally reduced ($D$-reduced) if and only if it does not contain any derivationally redundant clauses.

A clausal theory may have several D-reductions. For instance, we can easily see that

$$
\begin{align*}
C_1 & : P(A, B) \leftarrow Q(B, A) \\
C_2 & : P(A, B) \leftarrow Q(A, C), R(C, B) \\
C_3 & : P(A, B) \leftarrow Q(A, C), R(B, C)
\end{align*}
$$

may be D-reduced to either $\{C_1, C_2\}$ or $\{C_1, C_3\}$

**Hypothesis Space** Recall that $P$ (resp. $P_0$) denotes the set of predicates (resp. of initial predicates); and $B$ the background knowledge, may encompass both initial predicates and pre-given rules. As we are interested in characterizing the expressivity of our model, let us define the relevant notion of hypothesis space,

**Definition 8.7.** Given a predicate set $P$, and a meta-rule or proto-rule set $M$, let us define:

(i) the set $M_P$ as the set of all Horn clauses generated by all the possible substitutions of predicates variables in $M$ by predicates symbols from $P$.

(ii) the hypothesis space $M[P]$ generated by $M$ from $P$ as the Horn closure of $M_P$.

---

In this paper, we adopted the denomination of hypothesis space, which is unconventional in logic, because it refers to the space accessible to our learning algorithm. Similarly, by incorporating initial rules, we could also define the Horn theory generated by a set of meta-rules or proto-rules given a certain background knowledge $B$ and a set of predicates $P$. 

17In this paper, we adopted the denomination of hypothesis space, which is unconventional in logic, because it refers to the space accessible to our learning algorithm. Similarly, by incorporating initial rules, we could also define the Horn theory generated by a set of meta-rules or proto-rules given a certain background knowledge $B$ and a set of predicates $P$. 

16
Equality Assumption  The Equality relation is typically assumed part of FOL, so it is reasonable to assume it belongs to the background knowledge in our results below. In our approach, it amounts to integrating the predicate Equal to the set of initial predicates \( P_0 \); Equal, which corresponds to the identity, is intensionally defined on any domain \( D \) by:

\[
Equal : \{ D \times D \} \rightarrow \{ \text{True}, \text{False} \} = \begin{cases} 
\text{True} & \text{if } X = Y \\
\text{False} & \text{if } X \neq Y 
\end{cases}
\] (21)

Main Result  Here is our main result, concerning the investigation of the expressivity of the minimal set \( R_0 \):

**Theorem 8.8.**  (i) Assuming \( \text{True} \in P_0 \), the hypothesis space generated by \( R_0 \) from \( P \) encompasses the set of duplicate-free and function-free definite Horn clause composed of clauses with at most two body atoms involving unary and binary predicates in \( P \):

\[
\text{True} \in P_0 \implies R_0[P] \supset U_{\leq 2}^{(1,2)}[P]
\]

(ii) Assuming \( \text{True, Equal} \in P_0 \), the hypothesis space generated by \( R_0 \) from \( P \) corresponds to the set of function-free definite Horn clause composed of clauses with at most two body atoms involving unary and binary predicates in \( P \); i.e., in terms of the second order logic fragment:

\[
\text{True, Equal} \in P_0 \implies R_0[P] = F_{\leq 2}^{(1,2)}[P]
\]

We refer to the end of this section for the proof of this theorem.

**Corollary 8.9.**  Assuming the initial predicates \( P_0 \) contains only predicates of arity at most 2, the hypothesis space generated by \( R_0 \) given \( P \) encompasses the one generated by \( R_{MIL} \) as defined in (17).

\[
\forall P \in P_0, \quad \text{arity}(P) \leq 2 \implies R_0[P] \supset R_{MIL}[P]
\]

**Proof.**  Under our assumption, the meta-rule (Curry) in (17) may be disregarded. The remaining meta-rules present in \( R_{MIL} \) have already been examined in Theorem 8.8. \( R_0 \) has therefore at least the same expressivity than \( R_{MIL} \). In order to be able to conclude \( R_0 \) is strictly more expressive, we can mention the rules \( P(A) \leftarrow P(A, B) \) or \( P(A, B) \leftarrow P(B, A) \), which are reached by \( R_0 \) not \( R_{MIL} \).

Disjunction  In our model, to ensure an incremental learning of the target rule, we impose the restriction that each auxiliary predicate corresponds to one rule. Since such constraint risk to remove the possibility of disjunction, we redefine new proto-rules versions integrating disjunctions:

\[
R_0^\lor = \left\{ \begin{array}{l}
\forall \ : \quad P(A) \leftarrow (Q(A, B) \land R(B, A)) \lor S(A, B) \\
\exists \ : \quad P(A, B) \leftarrow (Q(A, C) \land R(C, B)) \lor S(A, B) \\
\in \ : \quad P(A, B) \leftarrow (Q(A, B) \land R(B, A)) \lor S(A, B) 
\end{array} \right\}
\] (22)

Adopting the minimal set \( R_0^\lor \), in our incremental setting does not affect the expressivity of the hypothesis space compared to when relying on the set \( R_0 \) in a non incremental setting. Let us remind an *incremental setting* imposes a predicate symbol to be the head of exactly one rule.
Lemma 8.10. The minimal proto-rule set $R_0^\cap$ in an incremental setting holds the same expressivity as $R_0$.

Proof. It is straightforward to see that $R_0^\cap$ is a minimal set. The hypothesis space generated by $R_0$ obviously encompasses the hypothesis space generated by $R_0^\cap$ in an incremental setting. Reciprocally, let us assume $n$ rules instantiated from $R_0$ are attached to one predicate symbol $P$. They can be expressed as:

$$
P(X,Y) \leftarrow f_1(X,Y)
$$
$$
P(X,Y) \leftarrow f_2(X,Y)
$$
$$\cdots \cdots \cdots
$$
$$
P(X,Y) \leftarrow f_n(X,Y)
$$

Recursively, we can define $P_i$ auxiliary predicates in the hypothesis space of $R_0^\cap$.

$$
P_1(X,Y) \leftarrow f_1(X,Y)
$$
$$
P_2(X,Y) \leftarrow P_1(X,Y) \lor f_2(X,Y)
$$
$$\cdots \cdots \cdots
$$
$$
P_n(X,Y) \leftarrow P_{n-1}(X,Y) \lor f_n(X,Y)
$$

Since $P_n$ is then equivalent to $P$, we can conclude.

Redundancy Since we have observed a certain redundancy to be beneficial to the learning, we incorporate the permutation rule $I$ in our experiments, and rely on the following set:

$$
R_* = R_0^\cap \cup \{ I : H(X,Y) \leftarrow F(Y,X) \}
$$

This small—yet non minimal—set of protorules $R_*$, has the same expressivity as $R_0$ (characterized in Theorem 8.8).

Lemma 8.11. The proto-rule set $R_*$ holds the same expressivity than $R_0$.

Proof. It stems from the fact, observed in the proof of Theorem 8.8 that the rule $\sigma$, equivalent to $I$ can be D-reduced from $R_0$.

Recursivity The development of Recursive Theory arose first around the Hilbert’s Program and Gödel’s proof of the Incompleteness Theorems (1931) and is tightly linked to questions of computability. While implementing recursive-friendly model, we should keep in mind considerations of computability and decidability. Enabling full recursivity within the templates is likely to substantially affect both the learning and the convergence, by letting room to numerous inconsistencies.

With this in mind, let us consider different stratifications in the rule space, and in our model. First, we introduce a hierarchical filtration of the rule space: initial predicates are set of layer 0, and auxiliary predicates in layer $\ell$ are defined from predicates from layers at most $\ell$. Within such stratified space, different restrictions may be considered:

- recursive-free hierarchical hypothesis space, where rules of layer $\ell$ are defined from rules of strictly lower layer. The proto-rules from $R_0^\cap$ are therefore restricted to the following form:

$$
P(\cdot) \leftarrow [Q(\cdot) \land R(\cdot)] \lor O(\cdot),
$$

with $P \in P_\ell, Q, R, O \in P_{<\ell}$. 
Proof of Theorem 8.8

Before we present the proof, let us introduce two additional notations:

- **iso-recursive hierarchical hypothesis space**, where the only recursive rules allowed from $\mathcal{R}_0$ have the form:

  $$P(\cdot) \leftarrow [Q(\cdot) \land R(\cdot)] \lor O(\cdot), \quad \text{with } P \in \mathcal{P}_t, Q, R \in \mathcal{P}_{\leq t} \cup \{P\}, O \in \mathcal{P}_{< t}.$$  

  Such space fits most recursive ILP tasks (e.g., Fizz, Buzz, Even, LessThan).

- **recursive hierarchical hypothesis space**, where the recursive rules allowed from $\mathcal{R}_0$ have the form:

  $$P(\cdot) \leftarrow [Q(\cdot) \land R(\cdot)] \lor O(\cdot) \quad \text{with } P \in \mathcal{P}_t, Q, R \in \mathcal{P}_{\leq t}, O \in \mathcal{P}_{\leq t}.$$  

  Such space is required for the recursiveness present in EvenOdd or Length. Such hierarchical notion is also present in the Logic Programming literature under the name of *stratified programs*.

**Proof of Theorem 8.8** Before we present the proof, let us introduce two additional notations:

- $\mathcal{R}_i \odot \mathcal{R}'$ refers to the resolvent of $\mathcal{R}$ with $\mathcal{R'}$, where the $i^{th}$ body member of $\mathcal{R}$ is resolved with the head of $\mathcal{R}'$.

  To illustrate it, consider the following meta-rules:

  $\mathcal{R}_i : \quad P(A) \leftarrow Q(A) \land R(A, B)$
  $\mathcal{R}'_i : \quad P(A) \leftarrow Q(A) \land R(B, A)$
  $\mathcal{R}''_i : \quad P(A, B) \leftarrow Q(B, A)$

  The resolvent $\mathcal{R}'_2 \odot \mathcal{R}''_2$ corresponds to resolve the second body atom of $\mathcal{R}'_2$ with $\mathcal{R}''_2$, which therefore leads to $\mathcal{R}_2$.

- We define the operation $\rho$ as the composition of a projection $\nu$, a permutation $(\sigma)$, and an existential contraction over the second variable $(\exists)$, which amounts to:

  $$\rho(R)(X) := \exists \sigma(\mathcal{R}(X, Y)) = \exists Y R(Y) \quad (23)$$

  Here, we identified the permutation clause $(\sigma)$ (as defined in (16)) with the operation it defines on predicates $\sigma$; similarly for the existential clause $(\exists)$ with the contraction operation $\exists$.

  Likewise, we can define the corresponding non-connected meta-rule attached to the operation $\rho$ by:

  $$(\rho) : \quad P(X) \leftarrow \exists Y R(Y) \quad (24)$$

**Proof.** The inclusion $\mathcal{R}_0[\mathcal{P}] \subset \mathcal{F}_{\leq 2}^{[1, 2]}[\mathcal{P}]$ is trivial since all the rules instantiated from $\mathcal{R}_0$ are definite Horn clause with two body predicates, and duplicate-free; similarly, if adding the duplicate-free constraint to $\mathcal{F}$ (and denoting it by $\mathcal{F}^D$), we can trivially state: $\mathcal{R}_0[\mathcal{P}] \subset \mathcal{F}^{D}_{\leq 2}^{[1, 2]}[\mathcal{P}]$.

  It remains to prove the other inclusion; e.g., for (ii), it amounts to $\mathcal{R}_0[\mathcal{P}] \supset \mathcal{F}_{\leq 2}^{[1, 2]}[\mathcal{P}]$, which boils down to $\mathcal{R}_0[\mathcal{P}] \supset \mathcal{F}^{[1, 2]}_{\leq 2}$ since $\mathcal{R}_0[\mathcal{P}]$ is closed under resolution. We will proceed by successively extending the sub-fragments $\mathcal{M}$ we are considering while proving $\mathcal{R}_0[\mathcal{P}] \supset \mathcal{M}_P$. For instance, under the assumption $\text{True}, \text{EqA1} \in \mathcal{P}_0$, we demonstrate that the hypothesis space generated by $\mathcal{R}_0$ encompasses the one generated by the increasingly larger fragments: (a) $\mathcal{U}$ (b) $\mathcal{K}$; (c) $\mathcal{D}$; (d) $\mathcal{C}$; (e) $\mathcal{F}$. For (i), under the simpler assumption $\text{True} \in \mathcal{P}_0$, we can follow almost identical steps, although the duplicate-free constraint can not be lifted, and therefore step (b) is skipped. Since the proof of (i) is more or less included in the proof of (ii), we will below focus only on proving (ii).

  For each fragment $\mathcal{M}$, to demonstrate such inclusion, we can simply show that every rule within the fragment $\mathcal{M}_P$ belongs to the hypothesis space generated by $\mathcal{R}_0$; more specifically, we will mostly work at the level of second order logic; there, we will show that any meta-rule in $\mathcal{M}$ can be reduced from meta-rules in $\mathcal{R}_0$, or generated by $\mathcal{R}_0$.  

19
However, to avoid listing all the meta-rules generating these fragments, and restrict our enumeration, we leverage several observations: first, since we assume the predicate set includes True, we can restrict our attention to clauses with exactly two body literals; secondly, the permutation meta-rule \((\sigma)\) (introduced in 16) can be derived from the proto-rule \(C\) upon matching the first body in \(C\) with True; therefore, we can examine rules only upon permutation \((\sigma)\) applied to their body or head predicates.

1. First, let us demonstrate the following inclusion:

\[
\text{Claim}_1: \quad \text{True} \in \mathcal{P}_0 \implies \mathcal{R}_0[\mathcal{P}] \supset U_{\leq 2}^{(1,2)}
\]  

(25)

By the previous observations, we can narrow down to examine a subset of the meta-rules generating \(U_{\leq 2}^{(1,2)}\), as listed and justified below. Our claim is that all these clauses are derivationally redundant from the Horn theory generated by \(\mathcal{R}_0\); for each of them, we therefore explicit the clauses used to reduce them\(^{18}\).

| Meta-Rules | Reduction | Comment |
|------------|-----------|---------|
| (i) \(P(A) \leftarrow Q(A) \land R(A)\) | \(\mathcal{A}\_2 \circ \sigma\) | \(\mathcal{A}(Q, \sigma(R))\) |
| (ii) \(P(A) \leftarrow Q(A) \land R(A, B)\) | \(\mathcal{A}\_2 \circ \sigma\) | \(\mathcal{A}(Q, \sigma(R))\) |
| (iii) \(P(A) \leftarrow Q(A, B) \land R(B)\) | \(\mathcal{A}\) | |
| (iv) \(P(A) \leftarrow Q(A, B) \land R(B, A)\) | \(\mathcal{A}\) | |
| (v) \(P(A) \leftarrow Q(A, B) \land R(B, C)\) | \(\mathcal{A}\) | |
| (vi) \(P(A, B) \leftarrow Q(A) \land R(A, B)\) | \(\mathcal{A}\_2 \circ \sigma\) | \(\mathcal{C}(Q, \sigma(R))\) |
| (vii) \(P(A, B) \leftarrow Q(A, C) \land R(C, B)\) | \(\mathcal{B}\) | |
| (viii) \(P(A, B) \leftarrow Q(A, B) \land R(B, A)\) | \(\mathcal{C}\) | |
| (ix) \(P(A, B) \leftarrow Q(A, B) \land R(B, C)\) | \(\mathcal{A}\_2 \circ \exists\) | \(\mathcal{A}(Q, \exists(R))\) |

Beside the notation of the resolvent \(\odot_H\) explained previously, we provided a more intuitive form for the resolution (under 'Comment'), in terms of composition between maps, where we identify a proto-rule (or meta-rule) with a map of the following form:

\[
\mathcal{P}\_\ast \times \cdots \times \mathcal{P}\_\ast \rightarrow \mathcal{P}\_\ast
\]

Let us justify why these clauses are the only clauses in \(U_{\leq 2}^{(1,2)}\) worth to examine in two steps:

- Let us consider clauses whose head predicate arity is 1. First, by the Datalog assumption, we can indeed restrict our attention to body clauses where \(A\) is appearing at least once. By symmetry, we can assume \(A\) appears in the first body. By the connected assumption, either \(A\) appears in the second body too, and/or an existential variable (say \(B\)) appears both in the first or second body. Upon permutation, it leads us to clauses (i) – (v).

- Let us consider clauses whose head predicate arity is 2. First, by the Datalog assumption, we can indeed restrict our attention to body clauses where \(A\) and \(B\) have to appear at least once. The only body clause of arity \((1, 1)\) satisfying this condition is not connected \((Q(A) \land R(B))\); upon permutation of \(A\) and \(B\), and of the body predicates \(Q\) and \(R\), and upon inversion \((\sigma)\) of the body predicates, the only body clause of arity \((1, 2)\) is of the form \(Q(A) \land R(A, B)\) (denoted \((vi)\) below, which reduces to \(C\) upon permutation); for arity \((2, 2)\), upon similar permutations, we can list three types of body clause, listed below as \((vii), (viii), (ix)\). While \((vii), (viii)\) are directly pointing to \(\mathcal{B}, C\), \((ix)\), can be obtained by reducing the second body of \(C\) through \((\exists)\) (as defined in 16). This justifies the remaining clauses, \((vi) \rightarrow (ix)\).

\(^{18}\)Note that we are allowed to use clauses \(\exists\) and \(\sigma\) in the reduction as we already explained why we can derive them from \(\mathcal{R}_0\) in the beginning of the proof of Theorem\(8.8\).
We can therefore conclude by Claim \(_1\).

2. Let us extend (25) by enabling duplicates in the clause:

\[
\text{Claim}_2: \text{True, Equal } \in \mathcal{P}_0 \implies \mathcal{R}_0[\mathcal{P}] \supseteq \mathcal{K}^{[1,2]}_{\leq 2}
\]  

(26)

To extend the result from \(\mathcal{U}^{[1,2]}_{\leq 2}\) to \(\mathcal{K}^{[1,2]}_{\leq 2}\), it is sufficient to show the meta-rules \((\nabla), (\Delta)\) from (16) can be derived from \(\mathcal{R}_0\), under the extra-assumption that \(\text{Equal}\) is included as background knowledge. This stems from the fact that these meta-rules can be written as follows:

| Meta-rule | Equivalent Form |
|-----------|-----------------|
| \((\nabla)\) | \(P(A, A) \leftarrow Q(A)\) \(P(A, B) \leftarrow Q(A) \land \text{Equal}(A, B)\) |
| \((\Delta)\) | \(P(A) \leftarrow Q(A, A)\) \(P(A) \leftarrow Q(A, B) \land \text{Equal}(A, B)\) |

Since the equivalent forms above are duplicate-free and two-connected, by (25), \((\nabla), (\Delta)\) can be derived from \(\mathcal{R}_0\), which implies \(\text{Claim}_2\).

3. Removing the assumption of two-connectedness from (26), we claim the following inclusion holds:

\[
\text{Claim}_3: \text{True, Equal } \in \mathcal{P}_0 \implies \mathcal{R}_0[\mathcal{P}] \supseteq \mathcal{D}^{[1,2]}_{\mathcal{P}, \leq 2}
\]  

(27)

The additional meta-rules we have to reduce can be narrowed down to:

\[
P(A) \leftarrow Q(A, A) \land R(A, C) \quad P(A, B) \leftarrow Q(A, A) \land R(A, B) \quad P(A, A) \leftarrow Q(A, A) \land R(A, B)
\]

By using the reduction for duplicated forms \(P(A, A), Q(A, A)\) stated previously in (b), these meta-rules may be derived from \(\mathcal{R}_0\), and \(\text{Claim}_3\) follows.

4. In the next step, we extend (27) to the connected Horn fragment:

\[
\text{Claim}_4: \text{True, Equal } \in \mathcal{P}_0 \implies \mathcal{R}_0[\mathcal{P}] \supseteq \mathcal{C}^{[1,2]}_{\mathcal{P}, \leq 2}
\]  

(28)

To get rid of the Datalog constraint that each head variable appears in the body, we are brought to examine the following meta-rules:

\[
P(A, B) \leftarrow Q(A, C)
\]

This meta-rule may be seen as a subcase of \(\Box\) once we have matched its second body with \(\text{True}\). It ensues that \(\mathcal{R}_0\) is able to generate the fragment \(\mathcal{C}^{[1,2]}_{2}\), as stated in \(\text{Claim}_4\).

5. Lastly, we can get rid of the assumption of "connected".

\[
\text{Claim}_5: \text{True, Equal } \in \mathcal{P}_0 \implies \mathcal{R}_0[\mathcal{P}] \supseteq \mathcal{F}^{[1,2]}_{\mathcal{P}, \leq 2}
\]  

(29)

Upon symmetries and permutations, we are brought to examine the following non-connected meta-rules:

| Meta-rule | Reduction | Comment |
|-----------|-----------|---------|
| (i) \(P(A) \leftarrow Q(A) \land R(B)\) | \(\Box\) | \(\Box(\exists(Q), R)\) |
| (ii) \(P(A) \leftarrow Q(A, B) \land R(C)\) | \(\Box_1 \circ \exists\) \(\Box(\exists(Q) \land \exists(R))\) |
| (iii) \(P(A) \leftarrow Q(A, B) \land R(C, D)\) | \(\Box_1 \circ \exists \circ \exists\) | |
| (iv) \(P(A, B) \leftarrow Q(A, B) \land R(C)\) | \(\Box_2 \circ \rho\) | \(\Box(Q, \rho(R))\) |
| (v) \(P(A, B) \leftarrow Q(A) \land R(B, C)\) | \(\Box_2 \circ \sigma\) | \(\Box(Q, \sigma(R))\) |
| (vi) \(P(A, B) \leftarrow Q(A, B) \land R(C, D)\) | \(\Box_2 \circ (\rho \circ \exists)\) | \(\Box(Q, \rho \circ \exists(R))\) |
| (vii) \(P(A, B) \leftarrow Q(A, C) \land R(B, D)\) | \(\Box_1 \circ \exists \circ \exists\) | \(\Box(\exists(Q) \land \sigma(R))\) |
Note that the clause/operation $\rho$, defined in \ref{23,24}, enables to express the clause $R(C)$ (resp. $R(C, D)$) appearing in the body of (iv) (resp. (vi)) as $(\rho)(R)(B)$. Since, by definition of $(\rho)$, it can be derived from $R_0$, we can thereupon conclude that all the above meta-rules are reducible to $R_0$, which concludes the proof.

We have therefore proven the following inclusions:

\[
\begin{align*}
\text{True, Equal} & \in P_0 \implies R_0[P] = F^{1,2}_{P, \leq 2} \supset C^{1,2}_{P, \leq 2} \supset D^{1,2}_{P, \leq 2} \supset K^{P,1,2}_{2} \supset U^{1,2}_{P, \leq 2}.
\end{align*}
\]

\[\Box\]

### B  ILP Experiment Results

**Extracted Interpretable Solutions**

We give a detailed description of the ILP tasks in our experiments, including the target, background knowledge, positive/negative examples, and the learned solution by our method for each task. Note that the solution for *Fizz* is missing since our current approach does not solve it with the generic proto-rules in $R_*$.

**Predecessor**  
The goal of this task is to learn the *predecessor*($X,Y$) relation from examples. The background knowledge is the set of facts defining predicate zero and the successor relation $\text{succ}$

\[
B = \{\text{True, False, zero(0), succ(0,1), succ(1,2),...}\}.
\]

The set of positive examples is

\[
P = \{\text{target}(1,0), \text{target}(2,1), \text{target}(3,2),...\}
\]

and the set of negative examples is

\[
N = \{\text{target}(X,Y)|(X,Y) \in \{0,1,...\}^2 \} - P.
\]

Among these examples, $\text{target}$ is the name of the target predicate to be learned. For example, in this task, $\text{target} = \text{predecessor}$. We use fixed training data for this task given the range of integers. One solution found by our method is:

\[
\text{target}(X,Y) \leftarrow \text{succ}(X,Y).
\]

**Undirected Edge**  
A graph is represented by a set of $\text{edge}(X,Y)$ atoms which define the existence of an edge from node $X$ to $Y$. The goal of this task is to learn the *undirected-edge*($X,Y$) relation from examples. This relation defines the existence of an edge between nodes $X$ and $Y$ regardless of the direction of the edge. An example set of background knowledge is

\[
B = \{\text{True, False, edge(a,b), edge(b,c)}\}.
\]

The corresponding set of positive examples is

\[
P = \{\text{target}(a,b), \text{target}(b,a), \text{target}(b,c), \text{target}(c,b)\}
\]

and the set of negative examples is

\[
P = \{\text{target}(a,c), \text{target}(c,a)\}.
\]
We use randomly generated training data for this task given the number of nodes in the graph. One solution found by our method is:

\[
\text{target}(X, Y) \leftarrow (\text{aux1}(X, Y) \land \text{edge}(Y, X)) \lor \text{edge}(X, Y),
\]

\[
\text{aux1}(X, Y) \leftarrow \text{edge}(X, Y),
\]

where \text{aux1} is an invented auxiliary predicate.

**Less Than** The goal of this task is to learn the \text{less-than}(X, Y) relation which is true if \(X\) is less than \(Y\). Here the background knowledge is the same as that in Predecessor task. The set of positive examples is

\[
\mathcal{P} = \{\text{target}(X, Y) | X < Y\}
\]

and the set of negative examples is

\[
\mathcal{N} = \{\text{target}(X, Y) | X \geq Y\}.
\]

We use fixed training data for this task given the range of integers. One solution found by our method is:

\[
\text{target}(X, Y) \leftarrow (\text{target}(X, Z) \land \text{target}(Z, Y)) \lor \text{succ}(X, Y).
\]

**Member** The goal of this task is to learn the \text{member}(X, Y) relation which is true if \(X\) is an element in list \(Y\). Elements in a list are encoded with two relations \text{cons} and \text{values}, where \text{cons}(X, Y) if \(Y\) is a node after \(X\) with null node 0 being the termination of lists and \text{value}(X, Y) if \(Y\) is the value of node \(X\).

Take the list \([3, 2, 1]\) as an example. The corresponding set of positive examples is

\[
\mathcal{P} = \{\text{target}(3, [3, 2, 1]), \text{target}(2, [3, 2, 1]), \text{target}(1, [3, 2, 1]),
\]

\[
\text{target}(2, [2, 1]), \text{target}(1, [2, 1]), \text{target}(3, [3, 2]),
\]

\[
\text{target}(2, [3, 2]), \text{target}(3, [3, 1]), \text{target}(1, [3, 1]),
\]

\[
\text{True, False}\}
\]

and the set of negative examples is

\[
\mathcal{P} = \{\text{target}(3, [2, 1]), \text{target}(1, [3, 2]), \text{target}(2, [3, 1])\}.
\]

We use randomly generated training data for this task given the length of the list. One solution found by our method is:

\[
\text{target}(X, Y) \leftarrow (\text{target}(X, Z) \land \text{cons}(Z, Y)) \lor \text{value}(X, Y).
\]

**Connectedness** The goal of this task is to learn the \text{connected}(X, Y) relation which is true if there is a sequence of edges connecting nodes \(X\) and \(Y\). An example set of background knowledge is

\[
\mathcal{B} = \{\text{True, False}, \text{edge}(a, b), \text{edge}(b, c), \text{edge}(c, d)\}.
\]

The corresponding set of positive examples is

\[
\mathcal{P} = \{\text{target}(a, b), \text{target}(b, c), \text{target}(c, d), \text{target}(a, c), \text{target}(a, d), \text{target}(b, d)\}
\]

and the set of negative examples is

\[
\mathcal{P} = \{\text{target}(b, a), \text{target}(c, b), \text{target}(d, c), \text{target}(d, a), \text{target}(d, b), \text{target}(c, a)\}.
\]
We use randomly generated training data for this task given the number of nodes in the graph. One solution found by our method is:

\[ \text{target}(X, Y) \leftarrow (\text{target}(X, Z) \land \text{target}(Z, Y)) \lor \text{edge}(X, Y). \]

**Son** The goal of this task is to learn the \( \text{son-of}(X, Y) \) relation which is true if \( X \) is the son of \( Y \). The background knowledge consists of various facts about a family tree containing the relations \( \text{father-of}, \text{brother-of} \) and \( \text{sister-of} \). An example set of background knowledge is

\[ B = \{ \text{True}, \text{False}, \text{father}(a, b), \text{father}(a, c), \text{father}(a, d), \text{brother}(b, c), \text{brother}(d, c), \text{sister}(c, b) \}. \]

The corresponding set of positive examples is

\[ P = \{ \text{target}(b, a), \text{target}(d, a) \} \]

and the set of negative examples \( N \) is a subset of all ground atoms involving the target predicates that are not in \( P \).

We use randomly generated training data for this task given the number of nodes in the family tree. One solution found by our method is:

\[ \text{target}(X, Y) \leftarrow (\text{aux1}(X, Y) \land \text{father}(Y, X)) \lor \text{False} \]

\[ \text{aux1}(X) \leftarrow (\text{father}(X, Z) \land \text{True}) \lor \text{brother}(X, T) \]

where \( \text{aux1} \) is an invented auxiliary predicate.

**Grandparent** The goal of this task is to learn the \( \text{grandparent}(X, Y) \) relation which is true if \( X \) is the grandparent of \( Y \). The background knowledge consists of various facts about a family tree containing the relations \( \text{father-of} \) and \( \text{mother-of} \). An example set of background knowledge is

\[ B = \{ \text{father}(c, b), \text{father}(b, a), \text{mother}(d, b), \text{mother}(e, a), \text{True}, \text{False} \}. \]

The corresponding set of positive examples is

\[ P = \{ \text{target}(c, a), \text{target}(d, a) \} \]

and the set of negative examples \( N \) is a subset of all ground atoms involving the target predicates that are not in \( P \).

We use randomly generated training data for this task given the number of nodes in the family tree. One solution found by our method is:

\[ \text{target}(X, Y) \leftarrow (\text{aux1}(X, Z) \land \text{aux1}(Z, Y)) \lor \text{False}, \]

\[ \text{aux1}(X, Y) \leftarrow (\text{mother}(X, Y) \land \text{True}) \lor \text{father}(X, Y), \]

where \( \text{aux1} \) is an invented auxiliary predicate.

**Adjacent to Red** In this task, nodes of the graph are colored either green or red. The goal of this task is to learn the \( \text{is-adjacent-to-a-red-node}(X) \) relation which is true if node \( X \) is adjacent to a red node. Other than the relation \( \text{edge} \), the background knowledge also consists of facts of relations \( \text{colour} \) and \( \text{red} \), where \( \text{colour}(X, C) \) if node \( X \) has colour \( C \) and \( \text{red}(C) \) if colour of \( C \) is red.

An example set of background knowledge is

\[ B = \{ \text{True}, \text{False}, \text{edge}(a, b), \text{edge}(b, a), \text{edge}(d, e), \text{edge}(d, f), \text{colour}(a, p), \text{red}(p), \text{colour}(d, q), \text{red}(q) \}. \]
The corresponding set of positive examples is

\[ P = \{ \text{target}(b), \text{target}(e), \text{target}(f) \} \]

and the set of negative examples \( N \) is a subset of all ground atoms involving the target predicates that are not in \( P \).

We use randomly generated training data for this task given the number of nodes in the graph. One solution found by our method is:

\[
\begin{align*}
\text{target}(X) & \leftarrow (\text{edge}(X, Z) \land \text{aux1}(Z, X)) \lor \text{False}, \\
\text{aux1}(X) & \leftarrow (\text{colour}(X, Z) \land \text{red}(Z, X)) \lor \text{False},
\end{align*}
\]

where \( \text{aux1} \) is an invented auxiliary predicate.

**Two Children** The goal of this task is to learn the \textit{has-at-least-two-children}(X) relation which is true if node \( X \) has at least two child nodes. Other than the relation \textit{edge}, the background knowledge also consists of facts of \textit{not-equals} relation \( \text{neq} \), where \( \text{neq}(X, Y) \) if node \( X \) does not equal to node \( Y \).

An example set of background knowledge is

\[ B = \{ \text{True}, \text{False}, \text{edge}(a, b), \text{edge}(a, c), \text{edge}(c, d), \text{neq}(a, b), \text{neq}(a, c), \text{neq}(a, d), \text{neq}(b, c), \text{neq}(b, d), \text{neq}(c, d) \} \].

The corresponding set of positive example(s) is

\[ P = \{ \text{target}(a) \} \]

and the set of negative examples \( N \) is a subset of all ground atoms involving the target predicates that are not in \( P \).

We use randomly generated training data for this task given the number of nodes in the graph. One solution found by our method is:

\[
\begin{align*}
\text{target}(X) & \leftarrow (\text{aux1}(X, Z) \land \text{edge}(Z, X)) \lor \text{False}, \\
\text{aux1}(X, Y) & \leftarrow (\text{edge}(X, Z), \land \text{neq}(Z, Y)) \lor \text{False},
\end{align*}
\]

where \( \text{aux1} \) is an invented auxiliary predicate.

**Relatedness** The goal of this task is to learn the \textit{related}(X, Y) relation, which is true if two nodes \( X \) and \( Y \) have any family relations. The background knowledge is \textit{parent}(X, Y) if \( X \) is \( Y \)'s parent.

An example set of background knowledge is

\[ B = \{ \text{True}, \text{False}, \text{parent}(a, b), \text{parent}(a, c), \text{parent}(c, e), \text{parent}(c, f), \text{parent}(d, c), \text{parent}(g, h) \} \].

The corresponding set of positive examples is

\[ P = \{ \text{target}(a, b), \text{target}(a, c), \text{target}(a, d), \text{target}(a, e), \text{target}(a, f), \text{target}(b, a), \text{target}(b, c), \text{target}(b, d), \text{target}(b, e), \text{target}(b, f), \text{target}(c, a), \text{target}(c, b), \text{target}(c, d), \text{target}(c, e), \text{target}(c, f), \text{target}(d, a), \text{target}(d, b), \text{target}(d, c), \text{target}(d, e), \text{target}(d, f), \text{target}(e, a), \text{target}(e, b), \text{target}(e, c), \text{target}(e, d), \text{target}(e, f), \text{target}(f, a), \text{target}(f, b), \text{target}(f, c), \text{target}(f, d), \text{target}(f, e) \text{target}(g, h), \text{target}(h, g) \} \]
and the set of negative examples $N$ is a subset of all ground atoms involving the target predicates that are not in $P$.

We use randomly generated training data for this task given the number of nodes in the family tree. One solution found by our method is:

\[
\begin{align*}
target(X, Y) & \leftarrow (target(X, Z) \land aux1(Z, Y)) \lor parent(X, Y), \\
aux1(X, Y) & \leftarrow (target(X, Y) \land target(Y, X)) \lor parent(X, Y),
\end{align*}
\]

where $aux1$ is an invented auxiliary predicate.

**Cyclic** The goal of this task is to learn the $is\text{-}cyclic(X)$ relation which is true if there is a path, i.e., a sequence of $edge$ connections, from node $X$ back to itself. An example set of background knowledge is

\[
B = \{\text{True}, False, edge(a, b), edge(b, a), edge(d, c), edge(d, b)\}.
\]

The corresponding set of positive examples is

\[
P = \{\text{target}(a), \text{target}(b)\}
\]

and the set of negative examples $N$ is a subset of all ground atoms involving the target predicates that are not in $P$.

We use randomly generated training data for this task given the number of nodes in the graph. One solution found by our method is:

\[
\begin{align*}
target(X) & \leftarrow (aux1(X, Z) \land aux1(Z, X)) \lor False, \\
aux1(X, Y) & \leftarrow (aux1(X, Z) \land edge(Z, Y)) \lor edge(X, Y).
\end{align*}
\]

where $aux1$ is an invented auxiliary predicate.

**Graph Coloring** The goal of this task is to learn the $adj\text{-}to\text{-}same(X, Y)$ relation which is true if nodes $X$ and $Y$ are of the same colour and there is an edge connection between them. The background knowledge consists of facts about a coloured graph containing the relations $edge$ and $colour$, which are similar to those in the task $Adjacent\text{ to Red}$. An example set of background knowledge is

\[
B = \{\text{True}, False, edge(a, b), edge(b, a), edge(b, c), edge(a, d), colour(a, p), colour(b, p), colour(c, q), colour(d, q)\}.
\]

The corresponding set of positive examples is

\[
P = \{\text{target}(a, b), \text{target}(b, a)\}
\]

and the set of negative examples $N$ is a subset of all ground atoms involving the target predicates that are not in $P$.

We use randomly generated training data for this task given the number of nodes in the graph. One solution found by our method is:

\[
\begin{align*}
target(X, Y) & \leftarrow (edge(X, Z) \land aux1(Z, X)) \lor False, \\
aux1(X, Y) & \leftarrow (colour(X, Z) \land colour(Z, Y)) \lor False,
\end{align*}
\]

where $aux1$ is an invented auxiliary predicate.

**Length** The goal of this task is to learn the $length(X, Y)$ relation which is true if the length of list $X$ is $Y$. Similar to the task $Member$, elements in a list are encoded with two relations $cons$ and $succ$, where $cons(X, Y)$ if $Y$ is a node after $X$ with null node 0 being the termination of lists and $succ(X, Y)$ if $Y$ is the next value of integer $X$. Moreover, this task adds $zero(X)$ as another background predicate, which is true if $X$ is 0.
Take the list $[3, 2, 1]$ as an example, suppose node 0 is the end of a list. The background knowledge is

$$B = \{ \text{True}, \text{False}, \text{zero}(0), \text{succ}(0, 1), \text{succ}(1, 2), \text{succ}(2, 3) \}.$$ 

The corresponding set of positive examples is

$$P = \{ \text{target}([3, 2, 1], 3), \text{target}([2, 1], 2), \text{target}([1], 1) \}.$$ 

We use randomly generated training data for this task given the number of nodes in a list.

**Even-Odd** The goal of this task is to learn the $\text{even}(X)$ relation which is true if value $X$ is an even number. The background knowledge includes two predicates, one is $\text{zero}(X)$, which is true if $X$ is 0, another one is $\text{succ}(X, Y)$, which is true if $Y$ is the next value of $X$. An example set of background knowledge is

$$B = \{ \text{True}, \text{False}, \text{zero}(0), \text{succ}(0, 1), \text{succ}(1, 2), \ldots \}.$$ 

The corresponding set of positive examples is

$$P = \{ \text{target}(0), \text{target}(2), \text{target}(4), \ldots \}.$$ 

and the set of negative examples is

$$N = \{ \text{target}(1), \text{target}(3), \text{target}(5), \ldots \}.$$ 

Once the number of constants is given, the dataset is deterministic. One solution found by our method is:

$$\text{target}(X) \leftarrow (\text{zero}(X) \land \text{aux1}(Z, X)) \lor \text{zero}(X),$$

$$\text{aux1}(X, Y) \leftarrow (\text{aux1}(X, Z) \land \text{aux1}(Z, Y)) \lor \text{aux2}(X, Y),$$

$$\text{aux2}(X, Y) \leftarrow (\text{succ}(X, Z) \land \text{succ}(Z, Y)) \lor \text{False},$$

where $\text{aux1}$ and $\text{aux2}$ are invented auxiliary predicates.

**Even-Succ2** Even-succ has the same backgrounds and target predicates as Even-Odd. In [Campero et al., 2018], they provide two different templates set tailored to Even-Succ respectively to Even-Odd. However, in our approach, this difference is not relevant anymore: since we provide a uniform template set, these two tasks become identical.

**Buzz** The goal of this task is to learn the $\text{buzz}(X)$ relation which is true if value $X$ is divisible by 5. The background knowledge consists of 4 predicates: $\text{zero}(X)$ is true if $X$ is 0, $\text{succ}(X, Y)$ is true if $Y$ is the next value of $X$, $\text{pred1}(X, Y)$ is true if $Y = X + 3$, $\text{pred2}(X, Y)$ is true if $Y = X + 2$. An example set of background knowledge is

$$B = \{ \text{True}, \text{False}, \text{zero}(0), \text{succ}(0, 1), \text{succ}(1, 2), \ldots, \text{pred1}(0, 3), \text{pred2}(2, 4), \ldots, \text{pred2}(0, 2), \text{pred2}(1, 3), \ldots \}.$$ 

The corresponding set of positive examples is

$$P = \{ \text{target}(0), \text{target}(5), \ldots \}.$$ 

and the set of negative examples is

$$N = \{ \text{target}(1), \text{target}(2), \text{target}(3), \text{target}(4), \text{target}(6), \text{target}(7), \ldots \}.$$ 

27
Once the number of constants is given, the dataset is deterministic. One solution found by our method is:

\[
\text{target}(X) \leftarrow (\text{aux}_1(X, Z) \land \text{pred}_2(Z, X)) \lor \text{zero}(X),
\]
\[
\text{aux}_1(X, Y) \leftarrow (\text{aux}_2(X, Z) \land \text{pred}_1(Z, Y)) \lor \text{False},
\]
\[
\text{aux}_2(X, Y) \leftarrow (\text{True} \land \text{aux}_3(Y)) \lor \text{zero}(X),
\]
\[
\text{aux}_3(X) \leftarrow (\text{aux}_1(X, Z) \land \text{pred}_2(Z, X)) \lor \text{zero}(X),
\]

where \text{aux}_1, \text{aux}_2 and \text{aux}_3 are invented auxiliary predicates.

**Fizz** The goal of this task is to learn the \text{fizz}(X) relation which is true if value \text{X} is divisible by 3. The background knowledge is the same with \text{Even} − \text{Odd} task. The corresponding set of positive examples is

\[ P = \{ \text{target}(0), \text{target}(5), \ldots \} \]

and the set of negative examples is

\[ N = \{ \text{target}(1), \text{target}(2), \text{target}(3), \text{target}(4), \text{target}(6), \text{target}(7), \ldots \}. \]

Once the number of constants is given, the dataset is deterministic.

**Further Experimental Results**

In Table 6, we provide the results for all the ILP tasks.

| Task          | | Recursive | \(\partial\)ILP | LRI | Ours |
|---------------|---------------|------------|-------------|-----|------|
|               | \(|I|\)       |            |             |     |      |
| Predecessor   | 1             | No         | 100         | 100 | 100  |
| Undirected Edge| 1             | No         | 100         | 100 | 100  |
| Less than     | 1             | Yes        | 100         | 100 | 100  |
| Member        | 1             | Yes        | 100         | 100 | 100  |
| Connectedness | 1             | Yes        | 100         | 100 | 100  |
| Son           | 2             | No         | 100         | 100 | 100  |
| Grandparent   | 2             | No         | 96.5        | 100 | 100  |
| Adjacent to Red| 2             | No         | 50.5        | 100 | 100  |
| Two Children  | 2             | No         | 95          | 100 | 100  |
| Relatedness   | 2             | Yes        | 100         | 100 | 100  |
| Cyclic        | 2             | Yes        | 100         | 100 | 100  |
| Graph Coloring| 2             | Yes        | 94.5        | 100 | 100  |
| Length        | 2             | Yes        | 92.5        | 100 | 20   |
| Even-Odd      | 2             | Yes        | 100         | 40  | 40   |
| Even-Succ2    | 2             | Yes        | 48.5        | 100 | 40   |
| Buzz          | 2             | Yes        | 35          | 70  | 100  |
| Fizz          | 3             | Yes        | 10          | 10  | 0    |

Table 6: Percentage of successful runs among 10 runs. \(|I|\) is the smallest number of intensional predicates needed. Recursive means whether or not the solution needs to learn recursive rules.
About Length Task: The deceiving performance in this table for task Length can be easily explained: in theory our model corresponds to rule-induction with function-free definite Horn clause in theory; yet, de facto, in the recursive-case, both the number of layers and the number of instantiated auxiliary predicate by proto-rule per layer define the actual expressivity of the model. The number of of instantiated auxiliary predicate by proto-rule per layer is by default set at 1 in all our experiments, as we intended to share the same hyperparameters set on all tasks; however, to have a chance to solve the task Length, we should increase this number to 2, to widen the hypothesis space.

About Even–Odd Task: As mentioned above, task Even–Odd corresponds, for our method, to the same task as the task Even–Succ2, as the target predicate is identical. This is not the case for other ILP approaches like ∂ILP or LRI, as they hand-engineer different template sets for each of these two tasks, corresponding to the desired auxiliary predicate.

Operation Choices

In our implementation, the default choice is sum for POOL, min for AND, max for OR. In we experimentally compare different choices for these operations. The first column shows the results with our default choices, while other columns show the results by using max for POOL, product for AND, and prodminus\(^{19}\) for OR.

| Task               | default | POOL-max | AND-product | OR-prodminus |
|--------------------|---------|----------|-------------|--------------|
| Adjacent To Red    | 100     | 20       | 100         | 100          |
| Member             | 100     | 40       | 100         | 90           |
| Cyclic             | 100     | 50       | 90          | 100          |
| Two Children       | 100     | 70       | 80          | 80           |

Table 7: Percentage of successful runs among 10 runs using soft evaluation, obtained by models trained with different implementations for POOL, MERGE, AND, OR.

Limitations of LRI \cite{Campero et al., 2018}

Using LRI, if gathering all the templates needed for all ILP tasks (in Table 6) in a template set \(\mathcal{R}_{LRI}\), we obtain 18 templates. Table 8 demonstrates how the evaluation success rate decrease if trained LRI with \(\mathcal{R}_{LRI}\). While LRI can obtain good results as indicated in Table 6 if provided with meta-rules customized for each task, the performance quickly degrades for hard tasks when provided a set of more generic meta-rules. This phenomenon is more accentuated for more complex tasks. For easy tasks, like Predecessor, the performance didn’t change.

\(^{19}\text{prodminus}(v_1, v_2) = v_1 + v_2 - v_1 \ast v_2\)
| Tasks               | LRI with specific templates | LRI with $R_{LRI}$ |
|---------------------|-----------------------------|-------------------|
| Predecessor         | 100                         | 100               |
| Undirected Edge     | 100                         | 80                |
| Less than           | 100                         | 100               |
| Member              | 100                         | 30                |
| Connectedness       | 100                         | 100               |
| Son                 | 100                         | 80                |
| Grandparent         | 100                         | 90                |
| Adjacent to Red     | 100                         | 0                 |
| Two Children        | 0                           | 0                 |
| Relatedness         | 100                         | 100               |
| Cyclic              | 100                         | 0                 |
| Graph Coloring      | 0                           | 0                 |
| Length              | 100                         | 50                |
| Even-Odd            | 100                         | 20                |
| Even-Succ2          | 100                         | 10                |
| Buzz                | 70                          | 0                 |
| Fizz                | 10                          | 0                 |

Table 8: LRI’s performance with increasing number of rule templates. Measured by percentage of successful runs among 10 runs using soft evaluation.

C Other Experimental Results

Visual Genome Experiments

For this experiment, we use a dataset called GQA [Hudson and Manning, 2019b] which is a preprocessed version of the Visual Genome dataset [Krishna et al., 2017], since the original is commonly considered to be too noisy [Zellers et al., 2018]. In GQA, the original scene graphs have been converted into a collection of KBs, leading to 1.9M facts, 1.4M constants, and 2100 predicates. Following [Yang and Song, 2020], we filter this dataset to remove the predicates that appear less than 1500 times and to focus on the top 150 objects.

We train 150 models to provide a logic explanation to those 150 objects. For the evaluation metrics, we use recall $R_{@1}$ and recall $R_{@5}$, which are computed on a held-out set. $R_{@k}$ measures the fraction of ground-truth atoms that appear among the top $k$ most confident predictions in an image. In our model, even though all the models are instantiated with the same max layer $n_L$, the trained model may have different layers. Indeed, since each auxiliary predicate at layer $\ell$ may be formed with predicates from any layer 0 to $\ell$ in $P^\downarrow_{\ell}$, the trained model may contain 1 to $n_L$ layers (without counting the target predicate). Given the soft unification computation in (10), a trained model with a larger number of layers has a tendency to output smaller values. Therefore, to make the output of all the models comparable, we use a simple $L_2$ normalization before comparing the outputs of those trained models in order to compute the evaluation metrics.

In the multi-task setting we initialized background embeddings from GPT2. We compare the semantic space underlying these embeddings after fine-tuning (i.e. training of our model). More precisely, we use cosine similarity to measure distances between all pairs of background embeddings, sort and select top 10 closest pairs. As Table 9 sug-
gests, the fine-tuned embeddings pairs have akin similarities; this initialization choice may therefore help with the performance to some extent.

| Top 10 pairs of similar embeddings |
|-----------------------------------|
| GPT2                             |
| (bag, backpack), (arrow, apple), (airplane, air), (backpack, airplane), (animal, airplane), (arrow, animal), (at, above), (apple, animal), (bag, airplane) |
| Fine-tuned                        |
| (bag, backpack), (bag, airplane), (arrow, apple), (airplane, air), (arm, air), (backpack, airplane), (air, above), (apple, air), (arrow, animal), (arm, airplane) |

Table 9: Top 10 closest embeddings from GPT2 and our trained multi-task model.

**RL Experiments**

We also tested our model in a sequential decision-making framework: on the tasks Stack, Unstack and On from the Blocksworld environment, as described in Jiang and Luo [2019]. In those tasks, an agent has 50 steps to build a stack (Stack task), place all the blocks directly on the floor (Unstack task), or move a specific block into another (On task).

From all the valuations of the target predicate Move(X,Y), we compute a softmax policy used during the exploration. During training, the supervised BCE loss [13] is replaced by a standard PPO loss [Schulman et al., 2017] on the softmax policy. To estimate the advantage function, we relied on the same critic architecture defined by Zimmer et al. [2021].

At the end of the training, the symbolic policy is extracted and evaluated on 5 testing scenarios not seen during training. We reported the performance in Table 10. Our model can be better than NLRL and as good as NLM and DLM.

| Task | Rewards | Training time |
|------|---------|---------------|
|      | NLRL   | NLM | DLM | Ours | NLRL | NLM | DLM | Ours |
| Unstack | 0.914 | 0.920 | 0.920 | 0.920 | hours | minutes | minutes | minutes |
| Stack   | 0.877 | 0.920 | 0.920 | 0.920 |
| On      | 0.885 | 0.896 | 0.896 | 0.896 |

Table 10: Comparisons with NLRL/NLM/DLM in terms of rewards on the testing scenarios and the order of magnitude of training times.

**Sensitivity to Hyperparameters**

To evaluate the sensitivity of our model to hyperparameter, we tested other hyperparameter choices on some ILP tasks. We refer to Table 13 for the results obtained with default hyperparameters, presented Table 12. As Table 11 suggests, both smaller inference steps, or smaller depth will affect performance; this performance decrease natural, since it narrows down the hypothesis space, which may not contain anymore the solution needed for the task. However,
we can appreciate the fact that larger inference step or depth usually will not affect much the performance (until a limit). We can also notice that cosine performs better than other similarity functions; naturally, restricting or excluding the recursivity would perform well compared with full recursivity, unless the Task require it.

| task                | default | inference-steps (train-steps, eval-steps) | similarity | recursivity | max-depth |
|---------------------|---------|------------------------------------------|------------|-------------|-----------|
| Adjacent to Red     | 100     | (s_t − 2, s_e − 2)                       | L1         |              |           |
|                     |         |                                         | L2         | full        | 1         |
|                     |         |                                         | scalar-product | min       | 2         |
|                     |         |                                         | none       | max         | 3         |
|                     |         |                                         | iso-recursive | 0          | 5         |
|                     |         |                                         | 1          |             |           |
|                     |         |                                         | 2          |             |           |
|                     |         |                                         | 3          |             |           |
|                     |         |                                         | 4          |             |           |
| Member              | 100     | (s_t − 1, s_e − 1)                       | L1         |              |           |
|                     |         |                                         | L2         | full        | 1         |
|                     |         |                                         | scalar-product | min       | 2         |
|                     |         |                                         | none       | max         | 3         |
|                     |         |                                         | iso-recursive | 0          | 5         |
|                     |         |                                         | 1          |             |           |
|                     |         |                                         | 2          |             |           |
|                     |         |                                         | 3          |             |           |
|                     |         |                                         | 4          |             |           |
| Cyclic              | 100     | (s_t + 1, s_e + 1)                       | L1         |              |           |
|                     |         |                                         | L2         | full        | 1         |
|                     |         |                                         | scalar-product | min       | 2         |
|                     |         |                                         | none       | max         | 3         |
|                     |         |                                         | iso-recursive | 0          | 5         |
|                     |         |                                         | 1          |             |           |
|                     |         |                                         | 2          |             |           |
|                     |         |                                         | 3          |             |           |
|                     |         |                                         | 4          |             |           |
| Two Children        | 100     | (s_t + 2, s_e + 2)                       | L1         |              |           |
|                     |         |                                         | L2         | full        | 1         |
|                     |         |                                         | scalar-product | min       | 2         |
|                     |         |                                         | none       | max         | 3         |
|                     |         |                                         | iso-recursive | 0          | 5         |
|                     |         |                                         | 1          |             |           |
|                     |         |                                         | 2          |             |           |
|                     |         |                                         | 3          |             |           |
|                     |         |                                         | 4          |             |           |

Table 11: Percentage of successful runs among 10 runs using soft evaluation, obtained by models trained with different hyper-parameter choices. Here, s_t and s_e are default inference steps used in training and evaluation, shown in Table 13.

D Hyperparameters

We list relevant generic and task-specific hyper-parameters used for our training method in Tables 12 and 13, respectively. In Table 13, the hyper-parameters train-num-constants and eval-num-constants represent the number of constants during training and evaluation, respectively. We keep the values of the two hyper-parameters for each task the same as those used in Campero et al. [2018]. Note that our model do not require the actual knowledge of the depth of the solution; the max-depth parameter simply could be an upper bound. Although, this parameter could be reduced for simpler tasks, we set max-depth= 4 for all tasks to make our training method more generic.

| Hyper-parameter | recursivity | fuzzy-and | fuzzy-or | similarity | lr | lr-rules |
|-----------------|-------------|-----------|----------|------------|----|----------|
| Value           | full        | min       | max      | cosine     | 0.01  | 0.03     |

| Hyper-parameter | temperature | Gumbel-noise | Gumbel-noise-decay-mode |
|-----------------|--------------|--------------|-------------------------|
| Value           | 0.1          | 0.3          | linear                  |

Table 12: Generic hyper-parameters for all tasks.
### Table 13: Specialized hyper-parameters for each ILP task.

| Task               | max-depth | train-steps | eval-steps | train-num-constants | eval-num-constants |
|--------------------|-----------|-------------|------------|---------------------|-------------------|
| Predecessor        | 4         | 2           | 4          | 10                  | 14                |
| Undirected Edge    | 4         | 2           | 2          | 4                   | 6                 |
| Less than          | 4         | 12          | 12         | 10                  | 12                |
| Member             | 4         | 12          | 12         | 5                   | 7                 |
| Connectedness      | 4         | 4           | 4          | 5                   | 5                 |
| Son                | 4         | 4           | 4          | 9                   | 10                |
| Grandparent        | 4         | 4           | 4          | 9                   | 11                |
| Adjacent to Red    | 4         | 4           | 4          | 7                   | 9                 |
| Two Children       | 4         | 4           | 5          | 5                   | 7                 |
| Relatedness        | 4         | 10          | 12         | 8                   | 10                |
| Cyclic             | 4         | 4           | 4          | 6                   | 7                 |
| Graph Coloring     | 4         | 4           | 4          | 8                   | 10                |
| Even-Odd           | 4         | 6           | 8          | 11                  | 15                |
| Even-Succ2         | 4         | 6           | 8          | 11                  | 15                |
| Buzz               | 4         | 8           | 10         | 11                  | 16                |

In multi-task GQA, we use used the same generic hyperparameters as given in Table 12 except that we disallow recursivity and decreased the learning rate (lr) to 0.001 for background embeddings and rule learning rate (lr-rules) to 0.01 for intensional embeddings. Other specific hyperparameters are given in Table 14.

### Table 14: Multi-task GQA hyperparameters.

| Hyperparameter          | Value |
|-------------------------|-------|
| Max depth               | 3     |
| Train-steps             | 4     |
| Embedding-dim           | 30    |
| Train-iterations $n_T$  | 3000  |
| Train-num-positive-instances $n_p$ | 5 |
| Train-num-random-instances $n_r$ | 5 |

In reinforcement learning, we used the same generic hyperparameters as given in Table 12 except that we don’t use recursivity and decreased the learning rate to 0.005. Additionally, hyperparameters are given in Table 15.
### Table 15: Reinforcement learning hyperparameters.

| Hyperparameter                  | Value |
|---------------------------------|-------|
| Max depth                       | 6     |
| Train-steps                     | 6     |
| Train-num-constants             | 5     |
| Temperature of softmax policy   | 0.01  |
| GAE $\lambda$                  | 0.9   |
| $\gamma$                       | 0.99  |
| Trajectory per update           | 5     |
| PPO $\epsilon$-clipping        | 0.2   |
| GRU hidden neurons (critic)     | 64    |

#### E Some promises and limitations of our Model

**Expressivity Limitations**  As made explicit in Theorem 8.8, our model corresponds to rule-induction with function-free definite Horn clause. Of course, the number of layers, affects the actual expressivity of each model; in the non-recursive case, this parameter is sufficient to reach all the function-free recursive-free definite Horn clause. In the recursive case, the number of instantiated rules from the proto-rule set (by default set at 1) also affects the expressivity; this may be seen in the task Length in ILP, where two rules should be initiated per template set to widen sufficient hypothesis space.

The expressivity of our model could be extended by different ways, more or less computationally-expensive, and therefore more or less judicious. The points below would be object of further investigations and experiments; for this reason, we only share in this paper some succinct comments on different tracks to gain expressivity:

+ Enabling negations.
+ Enabling functions.
+ Enabling zero-ary predicates: We could easily extend our result to include 0 ary predicates.

\[
\text{Claim: By adding the zero-ary predicate } Z \text{ to } R_0, \text{ the hypothesis space reaches the fragment } F_{P, \leq 2}^{\leq 2} \quad (30)
\]

where \( Z : P() \leftarrow Q(A, B) \).

+ Enabling more body atoms

\[
\text{Claim: Enabling 3 or 4 body atoms would result in the same hypothesis space.} \quad (31)
\]

This can be proven by reducing clauses with 3 resp. 4 body atoms to 2 resp. 3 body atoms. Instead of a formal proof, let us gives a visual illustration of these reduction\(^{20}\) in Figure 3 resp. Figure 5. Red arrows denote head predicates; full grey arrows indicates two arrows which can be reduced into the dotted grey arrow to lower the number of body atoms by introducing an auxiliary predicate.

\[^{20}\]For instance the language bias present in Galárraga et al. [2013] is narrowing the space to connected, two-connected and duplicate-free Horn clauses $U$. On the one hand, $U$ is a smaller fragment than $F$ which our model is reaching. However, they consider clauses with $N$ body atoms ($N$ hypothetically small in their experiments), which could enable greater expressivity.

\[^{21}\]We considered clauses upon permutation and symmetries, as usual; we omitted clauses having two body atoms with the same variables, as they can be trivially reduced.
In contrast, as illustrated Figure 5, some rules with 5 body clauses are not reducible to 2 body clauses, such as:

\[ P(A, B) \leftarrow Q(A, C), R(A, D), S(B, C), T(B, D), U(C, D). \]  

(32)

As mentioned in Cropper and Tourret [2020], \( F_{\leq 5}^{(1,2)} \) is therefore not D-reducible to \( F_{\leq 2}^{(1,2)} \). However, allowing higher number of clauses (such as 5) may drastically increase the computational cost.

+ Enabling higher-arity.

For instance, considering arity 3 predicates, while keeping two body clauses, could enable to reach \( \mathcal{C}_{\leq 5}^{(1,2)} \):

\[ \text{Claim} : \mathcal{F}_{\leq 2}^{(1,2,3)}(\mathcal{P}) \supset \mathcal{F}_{\leq 5}^{(1,2)} \]  

(33)

Further comparative experiments could be led in the future, to test different minimal or small meta-rules or proto-rules set, and investigate some judicious balance of minimalism/redundancy and expressivity/efficiency in diverse tasks.

Figure 3: Reduction of Meta-rules with 3 body atoms

Figure 4: Reduction of Meta-rules with 4 body atoms

Figure 5: Irreducible Meta-Rule with 5 body atoms
Promises In contrast to most of previous ILP or differentiable ILP works, we hypothesize that our model could be particularly suited for continual learning in semantically-richer domains; for instance, in RL scenarios where an interpretable logic-oriented higher level policy seems pertinent (e.g., autonomous driving). Here some arguments to support this belief:

+ In contrast to previous works, our model is independent of the number of predicates, which may vary between training and testing.
+ As our experiments in Visual Genome suggest, our model can be bootstrapped by semantic or visual priors, to initialise the predicates.
+ In semantically-rich domains, the complexity of our model would be more advantageous than other ILP works. Semantically rich domains are characterized by a high number of initial predicates, while enabling a much lower embedding dimension $d$ (as these predicates are highly interdependent), $d << |\mathcal{P}_0|$; such inequality implies a lower complexity for our model than common methods which have to learn one coefficient per predicate.

+ The embedding-based approach enables to reason by analogy, and possibly quickly generalise to new predicates. For instance, if the model has learned the rule $\text{OverTake}(\cdot) \leftarrow \neg (P(X) \land \text{OnNextLane}(X))$, and the embedding $\theta_P$ is learned close to $\theta_{\text{Car}}$, the rule would generalise to include Truck, assuming $\theta_{\text{Car}} \sim \theta_{\text{Truck}}$, despite having never seen the new predicate Truck during training.