Neuro-Symbolic Hierarchical Rule Induction

Claire Glanois [1] **Zhaohui Jiang** [2] **Xuening Feng** [2] **Paul Weng** [2] **Matthieu Zimmer** [2] **Dong Li** [3] **Wulong Liu** [3] **Jianye Hao** [3 4]

Abstract

We propose Neuro-Symbolic Hierarchical Rule Induction (HRI), an efficient interpretable neuro-symbolic model, to solve Inductive Logic Programming (ILP) problems. In this model, which is built from a pre-defined set of meta-rules organized in a hierarchical structure, first-order rules are invented by learning embeddings to match facts and body predicates of a meta-rule. To instantiate HRI, we specifically design an expressive set of generic meta-rules, and demonstrate they generate a consequent fragment of Horn clauses. As a differentiable model, HRI can be trained both via supervised learning and reinforcement learning. To converge to interpretable rules, we inject a controlled noise to avoid local optima and employ an interpretability-regularization term. We empirically validate our model on various tasks (ILP, visual genome, reinforcement learning) against relevant state-of-the-art methods, including traditional ILP methods and neuro-symbolic models.

1. Introduction

Research on neuro-symbolic approaches (Santoro et al., 2017; Manhaeve et al., 2018; Dai et al., 2019; d'Avila Garcez & Lamb, 2020; Amizadeh et al., 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, differentiable learning) and symbolic methods (e.g., interpretability, generalizability) while addressing their respective limitations (e.g., brittleness for deep learning, scalability for symbolic methods).

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

For tackling ILP problems, we propose a new neuro-symbolic hierarchical model, denoted HRI, which is built upon a set of meta-rules. 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 generalization* skills[1]. In contrast to other approaches, HRI is also independent of the number of predicates: this number may vary during training and testing [2]. 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, through the embeddings initialization.

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 meta-rules for rule induction models has been shown to improve both learning time and predictive accuracies (Cropper & Muggleton, 2014; Fonseca et al., 2004). For our model to be both adaptive and efficient, we initially designed an expressive and minimal set $\mathcal{R}_0$ of proto-rules. While most neuro-symbolic approaches do not formally discuss the expressivity of their models, we provide a theoretical analysis

[1]IT University of Copenhagen, Denmark [2]UM-SJTU Joint Institute, Shanghai Jiao Tong University, Shanghai, China [3]Huawei Noah's Ark Lab, China [4]School of Computing and Intelligence, Tianjin University. Correspondence to: <paul.weng@sjtu.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

[1]Trained on smaller instances, it can generalize to larger ones.

[2]In the case some input predicates only appear in the held-out dataset used for the final evaluation, our model can still predict thanks to the pre-trained embeddings, as mentioned in the Visual Genome Experiment in Appendix C.

to characterize the expressivity of $\mathcal{R}_0$. Moreover, in contrast to traditional ILP work based on combinatorial search (Cropper & 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 $\mathcal{R}_0$, we propose an extended set $\mathcal{R}_*$ of proto-rules.

We validate our model using proto-rules in $\mathcal{R}_*$ 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 & Manning, 2019b) extracted from Visual Genome (Krishna et al., 2017)). Our model outperforms other methods such as NLIL (Yang & 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, 1995; Law et al., 2018; Cropper & Morel, 2021). 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 drawbacks include scalability and potential struggle 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 & Grefenstette, 2018)—although combinatorially less attractive—or to the membership of predicates in a clause (Payani & Fekri, 2019; Zimmer et al., 2021). However, these models are still 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, Dong et al. (2019) learn rules as shallow multi-layer perceptrons (MLP), losing thus in interpretability. In contrast to these neuro-symbolic 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 for knowledge graph (KG) (Guu et al., 2015; Yang et al., 2017; Yang & Song, 2020; Ren et al., 2021), predicate invention is understood as finding a relational path (Yang et al., 2017) or a combination of paths (Yang & Song, 2020) in the KG; a path amounts to a chain-like first-order rule. However, although computationally efficient, their restricted expressiveness limits their performance and applicability.

Embedding-based models In the context of KG 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., 2013; 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 Neural Theorem Provers (Rocktäschel & Riedel, 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., $\mathcal{C}$). 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 $\mathcal{C}$, a set of predicates $\mathcal{P}$, 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 $\mathcal{C}^n$ to the Boolean set $\mathbb{B} = \{\texttt{True}, \texttt{False}\}$,

n-ary *functions* are mappings from $\mathcal{C}^n$ to the set of constants $\mathcal{C}$, and *variables* correspond to unspecified constants. As customary in most ILP work, we focus on a function-free fragment[3] 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 that 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*:

$$\mathsf{H} \leftarrow \mathsf{B}_1 \wedge \mathsf{B}_2 \wedge \cdots \wedge \mathsf{B}_k \tag{1}$$

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[4].

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 given 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:

$$\begin{aligned} Even(X) &\leftarrow Zero(X) \\ Even(X) &\leftarrow Even(Y) \wedge Aux(Y,X) \\ Aux(X,Y) &\leftarrow Succ(X,Z) \wedge Succ(Z,Y), \end{aligned} \tag{2}$$

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] A *fragment* of a logical language is a subset of this language obtained by imposing syntactical restrictions on it.

[4] This rule can be read as: if, for a certain grounding, all the body atoms are true, then the head atom is also true.

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 allowed arity of the predicates. 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.

A meta-rule corresponds to a second-order clause with predicate variables. For instance, the *chain meta-rule* is:

$$\mathrm{H}(X,Y) \leftarrow \mathrm{B}_1(X,Z) \wedge \mathrm{B}_2(Z,Y), \tag{3}$$

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 $\boldsymbol{\theta}_P \in \mathbb{R}^d$ denotes its d-dimensional[5] embedding. The embeddings attached to a meta-rule like (3) with one head (H) and two body ($\mathrm{B}_1, \mathrm{B}_2$) predicate variables are $(\boldsymbol{\theta}_{\mathrm{H}}, \boldsymbol{\theta}_{\mathrm{B}_1}, \boldsymbol{\theta}_{\mathrm{B}_2}) \in \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d$. The soft unification is based on cosine to measure the degree of similarity of embeddings of predicates and meta-rule atoms. For example, $\alpha_{P\mathrm{H}} = \cos(\boldsymbol{\theta}_P, \boldsymbol{\theta}_{\mathrm{H}}) \in [0,1]$ 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 $\mathsf{P} = P(s,o)$ is represented as $(\boldsymbol{\theta}_P, s, o, v_{\mathsf{P}})$, where $\boldsymbol{\theta}_P$ is the embedding of predicate P, s and o are respectively the subject and object constants, and $v_{\mathsf{P}} \in [0,1]$ 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 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 $\mathsf{P} = P(x,y)$ with respect to a meta-rule $\mathfrak{R}$ of the form (3) and two ground atoms, $\mathsf{P}_1 = P_1(x,z)$ and $\mathsf{P}_2 = P_2(z,y)$, is computed as follows:

$$v(\mathsf{P}, \mathfrak{R}, \mathsf{P}_1, \mathsf{P}_2) = (\alpha_{P\mathrm{H}} \cdot \alpha_{P_1\mathrm{B}_1} \cdot \alpha_{P_2\mathrm{B}_2}) \cdot (v_{\mathsf{P}_1} \cdot v_{\mathsf{P}_2}) \,. \tag{4}$$

[5] In Campero et al. (2018), d equals the number of predicates.

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 $\mathfrak{R}$ is then computed as[6]:

$$v(\mathrm{P}, \mathfrak{R}) = \max_{\mathrm{P}_1, \mathrm{P}_2} v(\mathrm{P}, \mathfrak{R}, \mathrm{P}_1, \mathrm{P}_2). \tag{5}$$

Since P is matched with several meta-rules's heads, the new valuation of P after one forward-chaining inference step is:

$$v(\mathrm{P}) = \max\left(v_{old}(\mathrm{P}), \max_{\mathfrak{R}} v(\mathrm{P}, \mathfrak{R})\right), \tag{6}$$

where $v_{old}(\mathrm{P})$ is the previous valuation of P. The $\max$ over meta-rules allows an intensional predicate to be defined as a disjunction. After n_I 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, using the recursive task Even. The boxes, appearing in different layers, denote auxiliary predicates, which embody an asymmetrical disjunction of a conjunction of two atoms with a third body predicate. An arrow represents the soft unification between one body predicate and the heads of lower-level predicates, following (13); same-level predicates are also considered when the recursivity allowed, as in this example. In this figure, the fully-exposed arrows are highlighting one solution found by our model (mentioned in Appendix B.1), which can be written as:

$$\begin{aligned} Even(X) &\leftarrow aux31(X) \\ aux31(X) &\leftarrow (Zero(X) \wedge aux21(Y, X)) \vee Zero(X) \\ aux21(X, Y) &\leftarrow (aux21(X, Z) \wedge aux21(Z, Y)) \vee aux11(X, Y) \\ aux11(X, Y) &\leftarrow (Succ(X, Z) \wedge Succ(Z, Y)) \vee False, \end{aligned} \tag{7}$$

[6]The max over ground atoms is taken both over possible existential variable (e.g., Z above), and over predicates $P_1, P_2 \in \mathcal{P}$.

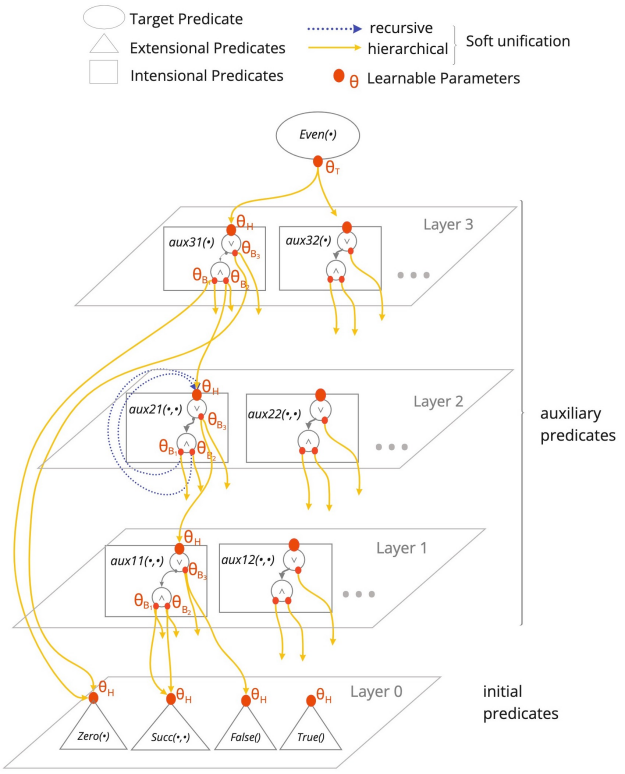


Figure 1. Hierarchical Model

Proto-rules We first define the notion of proto-rules, which implicitly correspond to sets of meta-rules. A *proto-rule* is an adaptive[7] meta-rule (see Appendix A for a formal definition). For instance, (3) extended as a proto-rule becomes:

$$\mathrm{H}(X, Y) \leftarrow \overline{\mathrm{B}}_1(X, Z) \wedge \overline{\mathrm{B}}_2(Z, Y), \tag{8}$$

where the $\overline{\mathrm{B}}_i$'s correspond to predicate variables, of arity 2 or 1, with an optional second argument. For instance, (8) can be instantiated as the following rule:

$$P(X, Y) \leftarrow P_1(X) \wedge P_2(Z, Y) \tag{9}$$

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

We propose a minimal and expressive set of proto-rules in Section 4.2. However, `HRI` can be instantiated with any set $\mathcal{R}$ of proto-rules. The choice of $\mathcal{R}$ 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

[7] It amounts to an embedding of lower-arity predicates into the space of higher-arity predicates.

addresses the underestimation issue of (4) since it amounts to setting $\alpha_{P\mathrm{H}} = 1$. Second, it partly reduces computational costs since the soft unification for a rule can be computed only by matching the body atoms[8]. In order to preserve expressivity[9], we can re-introduce disjunctions in the model by considering proto-rules with disjunctions in their bodies (see Section 4.2);[10] e.g., (8) could be extended to:

$$\mathrm{H}(X,Y) \leftarrow \left(\overline{\mathrm{B}}_1(X,Z) \wedge \overline{\mathrm{B}}_2(Z,Y)\right) \vee \overline{\mathrm{B}}_3(X,Y). \quad (10)$$

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 (extensional) predicates. For each layer $l = 1, \ldots, n_L$, each proto-rule[11] in $\mathcal{R}$ 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 ℓ can be only defined from a set $\mathcal{P}_\ell^{\downarrow}$ of predicates at a layer lower or equal to ℓ. 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, $\mathcal{P}_\ell^{\downarrow}$ is defined as the set $\mathcal{P}_{<\ell}$ (resp. $\mathcal{P}_{\leq\ell}$) of predicates at layer strictly lower than (resp. lower or equal to) ℓ. For iso-recursivity, $\mathcal{P}_\ell^{\downarrow}$ is defined as $\mathcal{P}_{<\ell} \cup \{P\}$.

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.

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 (10); although the equations can be straightforwardly adapted to any proto-rule. One inference step in our model is formulated as follows[12]:

$$\begin{aligned} v_{and} &= \textsc{pool}_{\mathsf{P}_1,\mathsf{P}_2}\left(\alpha_{P_1\mathrm{B}_1} \cdot \alpha_{P_2\mathrm{B}_2} \cdot \textsc{and}[v_{\mathsf{P}_1}, v_{\mathsf{P}_2}]\right) \\ v_{or} &= \textsc{or}\left[v_{and}, \textsc{pool}_{\mathsf{P}_3}\left(\alpha_{P_3\mathrm{B}_3} \cdot v_{\mathsf{P}_3}\right)\right] \\ v &= \textsc{merge}\left(v_{old}, v_{or}\right), \end{aligned} \quad (11)$$

where v (resp. v_{old}) denotes the new (resp. old) valuation of a grounded auxiliary predicate P. For an auxiliary predicate at layer ℓ, the POOL operation encompass a max over the groundings compatible to P followed by a pooling performed over both predicates $P_1, P_2, P_3 \in \mathcal{P}_\ell^{\downarrow}$. Indeed, in presence of an existential quantifier as in proto-rule (10), a max is applied to eliminate the corresponding existential variable before the actual pooling over predicates.

The valuation v^t of the target predicate (with variable denoted P^t) is computed as a MERGE of its previous valuation v_{old}^t and a POOL over atoms at layer n_L:

$$v^t = \textsc{merge}\left(v_{old}^t, \textsc{pool}_{\mathsf{P}_1}\left(\alpha_{P_1\mathrm{P}^t} \cdot v_{\mathsf{P}_1}\right)\right). \quad (12)$$

The underestimation issue is alleviated in two ways. First, the POOL operation is implemented as a sum, AND as $\min$, OR as $\max$, and MERGE as $\max$. Second, the unification scores $\alpha_{P_i\mathrm{B}_i}$'s based on cosine are renormalized with a softmax transformation:

$$\alpha_{P_i\mathrm{B}_i} = \frac{\exp(\cos(\boldsymbol{\theta}_{P_i}, \boldsymbol{\theta}_{\mathrm{B}_i})/\tau)}{\sum_{P_j} \exp(\cos(\boldsymbol{\theta}_{P_j}, \boldsymbol{\theta}_{\mathrm{B}_i})/\tau)}, \quad (13)$$

where $P_j \in \mathcal{P}_\ell^{\downarrow}$ and hyperparameter τ, called temperature, controls the renormalization. Score $\alpha_{P_1\mathrm{P}^t}$ can be computed in a similar way with embedding $\boldsymbol{\theta}_{\mathrm{P}^t}$. These choices have been further experimentally validated (see Appendix D).

Equation (11) implies that the computational complexity of one inference step at layer ℓ is $\mathcal{O}(|\mathcal{P}_\ell| \times |\mathcal{P}_\ell^{\downarrow}|^3 \times |\mathcal{C}|^2)$ where $\mathcal{P}_\ell$ is the set of predicates at layer ℓ and $\mathcal{P}_\ell^{\downarrow}$ is the set of predicates available for defining an auxiliary predicate at layer ℓ; parallelising the OR operation leads to a quadratic dependence in $|\mathcal{P}_\ell^{\downarrow}|$. Its spatial complexity is $\mathcal{O}(|\mathcal{P}| \times |\mathcal{C}|^2)$. The inference step described above is iterated n_I times, updating the valuations of auxiliary and target predicates. Note that when recursive definitions of predicates are allowed, it could be conceivable to iterate this inference step until convergence. A nice property of this inference procedure is that it is parallelizable, which our implementation on GPU takes advantage of.

4.2. Generic Set of Proto-Rules

Although `HRI` can be instantiated with any proto-rules, we design a small set of expressive proto-rules to instantiate it. We introduce first the following set of proto-rules $\mathcal{R}_0$:

$$\left\{\begin{array}{ll} \mathfrak{A}: \mathrm{H}(X) & \leftarrow \overline{\mathrm{B}}_1(X,Y) \wedge \overline{\mathrm{B}}_2(Y,X) \\ \mathfrak{B}: \mathrm{H}(X,Y) & \leftarrow \overline{\mathrm{B}}_1(X,Z) \wedge \overline{\mathrm{B}}_2(Z,Y) \\ \mathfrak{C}: \mathrm{H}(X,Y) & \leftarrow \overline{\mathrm{B}}_1(X,Y) \wedge \overline{\mathrm{B}}_2(Y,X) \end{array}\right\}$$

where the $\overline{\mathrm{B}}_i$'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

[8] The max over meta-rules in (6) is not needed anymore.

[9] Since identifying intensional predicates to proto-rules prevent them to have disjunctive definitions.

[10] Although such disjunction increases the computational cost again, it can be partly parallelised in the inference.

[11] Alternatively, several auxiliary predicates may be generated per proto-rules per layer. For simplicity, we only generate one and control the model expressivity with only one hyperparameter n_L.

[12] It generalizes (4)-(6), under the incremental prior.

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 $\mathcal{F}^{\{1,2\}}_{\mathcal{P},\leq 2}$ 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 results implies that our model with $\mathcal{R}_0$ can potentially solve perfectly any ILP task whose target predicate is expressible in $\mathcal{F}^{\{1,2\}}_{\mathcal{P},\leq 2}$ 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 $\mathcal{R}_0^\vee$ by including a disjunction with a third atom in all the proto-rules:

$$\left\{\begin{array}{ll} \mathfrak{A}_* : \mathrm{H}(X) & \leftarrow \big(\overline{\mathrm{B}}_1(X,Y) \wedge \overline{\mathrm{B}}_2(Y,X)\big) \vee \overline{\mathrm{B}}_3(X,T) \\ \mathfrak{B}_* : \mathrm{H}(X,Y) & \leftarrow \big(\overline{\mathrm{B}}_1(X,Z) \wedge \overline{\mathrm{B}}_2(Z,Y)\big) \vee \overline{\mathrm{B}}_3(X,Y) \\ \mathfrak{C}_* : \mathrm{H}(X,Y) & \leftarrow \big(\overline{\mathrm{B}}_1(X,Y) \wedge \overline{\mathrm{B}}_2(Y,X)\big) \vee \overline{\mathrm{B}}_3(X,Y) \end{array}\right\}$$

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

$$\mathcal{R}_* = \mathcal{R}_0^\vee \cup \{\mathfrak{I} : H(X,Y) \leftarrow \overline{F}(Y,X)\}$$

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

Supervised 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 (11) 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:

$$\sum_{x,y} -G^t_{x,y} \log\big(v(\mathsf{P}^t_{x,y})\big) - (1-G^t_{x,y}) \log\big(1-v(\mathsf{P}^t_{x,y})\big), \quad (14)$$

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

$$+\lambda \sum_{P,\mathrm{P}} \alpha_{P\mathrm{P}}\big(1-\alpha_{P\mathrm{P}}\big). \quad (15)$$

where hyperparameter λ controls the regularization weight and the sum is over all predicates P that can be matched with a predicate variable P 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 in (13), by a variant of Gumbel-softmax: we replace each $\cos(\boldsymbol{\theta}_{P_j}, \boldsymbol{\theta}_{B_i})$ by $\cos(\boldsymbol{\theta}_{P_j}, \boldsymbol{\theta}_{B_i}) + G_j$ where $G_j = -g_1 \log\left(-\log\left(g_2 U_j\right)\right)$ where $g_1 \in (0, +\infty)$ is the Gumbel scale, $g_2 \in (0, +\infty)$ is a novel hyperparameter, and $U_j \sim \mathcal{U}([0,1])$. With $g_2 = 1$, this leads to the usual Gumbel-softmax (Jang et al., 2017; Maddison et al., 2017). Introducing g_2 allows a better control of the noise range during training (see Appendix D for further discussion). In our experiments, g_2 is linearly-decreased during training, while g_1 is held fixed. In contrast to the low-level parameter-noise applied directly to the embeddings, G_j 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 ℓ, the extracted symbolic rule (9) could be extracted from proto-rule (8) if P is the auxiliary predicate associated to that proto-rule at layer ℓ and

$$\left\{\begin{array}{lcl} P_1 & = & \arg\max_{P \in \mathcal{P}^\downarrow_\ell} \alpha_{P\mathrm{B}_1} \\ P_2 & = & \arg\max_{P \in \mathcal{P}^\downarrow_\ell} \alpha_{P\mathrm{B}_2} \end{array}\right. \quad (16)$$

Then we can interpret the solution by successively unfolding the logical formula extracted for each involved predicate, starting from the target predicate.

6. Experimental Results

For the empirical validation of `HRI`[13], we carried out several series of experiments to answer the following questions: (1) how does `HRI` fare against other neuro-symbolic models? (2) when can `HRI` be superior to traditional ILP methods? (3) can `HRI` scale to tackle large domain? (4) how significant are the different choices in `HRI` ? For (1), we use both ILP and RL tasks. Since traditional ILP methods cannot be

[13]The source code of `HRI` and the scripts to reproduce the experimental results can be found at <https://github.com/claireaoi/hierarchical-rule-induction>.

applied to RL, we use ILP tasks for (2). For (3), we consider a large domain from Visual Genome (Krishna et al., 2017). For our ablation study (Table 11), we use again ILP tasks.

For space constraints, we only present a selection of the experimental results in the main paper. The remaining results, alongside with additional experiments (e.g., choice of operators, Gumbel Noise, limitations of LRI, sensitivity to hyperparameters) are discussed in Appendices B.2, B.4, C, D, and E. All of our results are averaged over several runs with different random seeds. In each run, a model is trained over a fixed number of iterations. Hyperparameter details are given in Appendix E.

For simplicity and consistency, we instantiate our model with $\mathcal{R}_*$, with full recursivity in all the ILP tasks. For the RL and Visual Genome tasks, we use no recursivity. In the latter domain, it is not expected to be helpful. Note the performances of `HRI` can be improved and/or learning can be accelerated if we customize $\mathcal{R}_*$ to a task or restrict its recursivity. However, we intended to emphasize the genericity of our approach.

6.1. Comparison with Neuro-Symbolic Models

ILP Experiments We first evaluate our model on the ILP tasks from Evans & Grefenstette (2018), which are detailed in Appendix B.1, where we also provide the interpretable solutions found by our model.

We compare our model with ∂ILP (Evans & Grefenstette, 2018) and LRI (Campero et al., 2018) using their reported results. A selection of the results on these ILP tasks are presented in Table 1 (see Appendix B.2 for full results). 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*) corresponds to the evaluation performance (no noise) using (12) (resp. the interpretable solution, as explained in the end of Section 5).

In contrast to both ∂ILP or LRI, which use a specific set of meta-rules tailored for each task, we use the same generic and more expressive template set $\mathcal{R}_*$ to tackle all tasks. Despite that, `HRI` can often match or outperform them. Naturally, on a few tasks, our generic approach based on $\mathcal{R}_*$ impedes learning, however our performance could be improved by enforcing more tailored meta-rules.

We also compare `HRI` to more generic neuro-symbolic models that do not require specific metarule templates: NLM (Dong et al., 2019) and DLM (Zimmer et al., 2021), which can scale better than methods like ∂ILP. For those two models, we run their respective open-source implementations that are also GPU-based, which provide a fair empirical comparison with `HRI`. To compare their performances and training times for a varying number of training constants, we evaluate them on two ILP tasks used by NLM: `Adjacent to Red and Grandparent`. Those two ILP tasks are similar to those used by Evans & Grefenstette (2018), but are defined 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 can scale much better in terms of training constants, while achieving at least as good performances.

RL Experiments We also tested our model in an RL setting on blocks manipulation tasks: `Stack`, `Unstack`, and `On` (Jiang & Luo, 2019). In those tasks, an agent has 50 steps to build a stack (`Stack`), place all the blocks directly on the floor (`Unstack`), or move a specific block on another (`On`). We compare `HRI` with NLRL (Jiang & Luo, 2019) (which extends ∂ILP to RL), NLM (Dong et al., 2019), and DLM (Zimmer et al., 2021).

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 (14) 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 test scenarios not seen during training. As shown in Table 4, our model outperforms NLRL and competes with NLM or DLM, both in terms of performance and training time.

6.2. Comparison with Traditional ILP Methods

To answer the second question, we selected two state-of-the-art classic ILP methods: ILASP3 (Law et al., 2018) and Popper (Cropper & Morel, 2021). Both are meta-level approaches, like most recent classic ILP approaches. While ILASP3 can handle noise, Popper can scale better.

In both the traditional ILP literature and the neuro-symbolic work, experimental comparisons of the two approaches are rare, with a few exceptions (Law et al., 2018). Our experiments in this section can be seen as a contribution to fill this gap and shed more light on which situations one approach may be preferred to another. We selected a sample of four diverse ILP tasks (e.g., easy vs harder tasks, need of recursion, etc.): `Connectedness`, `Grandparent`, `Member`, and `Undirected Edge`.

Our experimental evaluation indicates that ILASP3 and Popper are superior both in terms of performance and computational/space costs only when the dataset is small and noise-free, especially when a good language bias is available. However, when the dataset becomes larger, as shown in Figure 4 of Appendix B.4, `HRI` scales better because it

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.

Task	$\|I\|$	Recursive	∂ILP	LRI	Ours		
					train	soft evaluation	symbolic evaluation
Undirected Edge	1	No	100	100	100	100	100
Member	1	Yes	100	100	100	100	100
Connectedness	1	Yes	100	100	100	100	100
Grandparent	2	No	96.5	100	100	100	100
Adjacent to Red	2	No	50.5	100	100	100	100
Two Children	2	No	95	0	100	100	100
Graph Coloring	2	Yes	94.5	0	100	100	100
Even-Succ2	2	Yes	48.5	100	40	40	40
Buzz	2	Yes	35	70	100	40	40

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

Task	#Training constants	% successful runs			Training time (secs)		
		NLM	DLM	Ours	NLM	DLM	Ours
Adjacent to Red	7	100	90	100	163	920	62
	10	90	90	100	334	6629	71
Grandparent	9	90	100	100	402	2047	79
	20	100	100	100	1441	3331	89

Table 3. R@1 and R@5 for 150 objects classification on VG.

Model	Visual Genome	
	R@1	R@5
MLP+RCNN	0.53	0.81
Freq	0.40	0.44
NLIL	0.51	0.52
Ours	0.53	0.60

Table 4. Comparisons with NLRL/NLM/DLM in terms of rewards on the testing scenarios.

Task	Rewards			
	NLRL	NLM	DLM	Ours
Unstack	0.914	0.920	0.920	0.920
Stack	0.877	0.920	0.920	0.920
On	0.885	0.896	0.896	0.896

Table 5. Performance of different embedding initializations for the single Visual Genome task. Displaying both the soft and the symbolic evaluations (once the symbolic rule has been extracted).

Init.	Accuracy		Precision		R@1	
	soft	symb.	soft	symb.	soft	symb.
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

can be trained on mini-batches, which is not possible for traditional ILP methods.

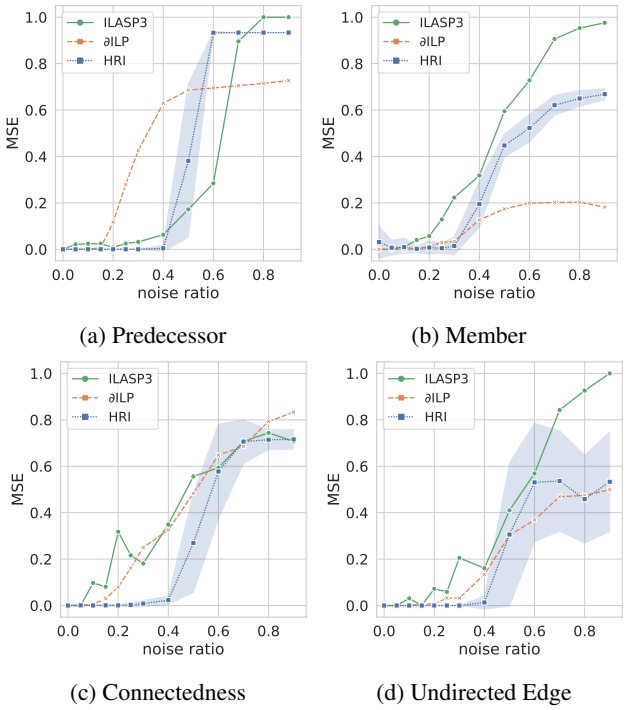
Noisy data Finally, we compared HRI with ILASP3 (Law et al., 2018) [14] and ∂ILP (Evans & Grefenstette, 2018) in a noisy data setting by ranging the ratio of mislabeled target training examples from 0% to 90%. We actually also evaluated ILASP4 (Law et al., 2020), which improves ILASP3 with a more efficient algorithm, using the experimental set-up described in (Law et al., 2018), however the results were worse than those of ILASP3. We discuss those experiments and results in Appendix B.4.

Our experiments suggest that HRI handles noise better than ILASP3 (and also ∂ILP) when the noise level is below about 40% (e.g., probability of incorrect positive/negative examples), as shown in Figure 2. This demonstrates the robustness of HRI since in practice noise levels would rarely be above 40%, otherwise a dataset would be hardly exploitable. Furthermore, note that ILASP3 adopts a very specific hypothesis bias and ∂ILP uses task-specific templates to obtain their results, while HRI is based on a much more general hypothesis space.

6.3. Visual Genome

Our model can be applied beyond classical ILP domains, and even benefit from richer environments and semantic structure. Furthermore, initial predicates embeddings can be bootstrapped by visual or semantic priors. We illustrate it by applying HRI to the larger dataset of Visual Genome

[14] The experimental results of ILASP3 and ∂ILP come from their own paper (Law et al., 2018; Evans & Grefenstette, 2018).



(a) Predecessor (b) Member

(c) Connectedness (d) Undirected Edge

Figure 2. Mean Squared Error (MSE) w.r.t. different noise ratios.

(Krishna et al., 2017), or more precisely a pre-processed (less noisy) version known as GQA (Hudson & Manning, 2019a). Similarly to Yang & Song (2020), we filtered out the predicates with less than 1500 occurrences, which lead to a KG with 213 predicates; then, we solved a multi-task ILP problem, which consists in learning the explanations for the top 150 objects in the dataset. More details on the multi-task setting, results, baseline, and metrics are provided in Appendix C. We also empirically confirm there that `HRI` can benefit from more informative pre-trained embeddings in a single-task setting (cf. Table 5). Based on those results, we used pretrained embeddings from the NLP model, GPT2 (Radford et al., 2019), in our other experiments on GQA.

In the 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 & 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, an 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.

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 & Song, 2020), and pretrained embeddings from NLP model GPT2 (Radford et al., 2019). We trained our model on a single ILP task, which consists in predicting predicate *Car*. For this task, we further filtered the GQA dataset to keep 185 instances containing cars. Table 5 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. Yet, to be fair, for the other experiments, we rely on the NLP embeddings.

7. Conclusion

We presented a new neuro-symbolic interpretable model performing hierarchical rule induction through soft unification with learned embeddings; it is initialized 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 ILP, RL, and richer domains against state-of-the-art baselines, where it is typically one to two orders of magnitude faster to train.

As discussed in Appendix F, future extensions of `HRI` could tackle further RL domains and continual learning scenarios benefiting from an interpretable logic-oriented policy, while extending our proto-rules set to broaden its expressivity.

Ethical Considerations Deploying our model trained with data from annotated images (e.g., Visual Genome), or embeddings produced by NLP models (e.g., GPT2) would raise ethical issues, because such data contain some biases as it has been well documented (Papakyriakopoulos et al., 2020; Tommasi et al., 2017).

References

Amizadeh, S., Palangi, H., Polozov, O., Huang, Y., and Koishida, K. Neuro-symbolic visual reasoning: Disentangling visual from reasoning. In *ICML*, 2020.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and

Yakhnenko, O. Translating embeddings for modeling multi-relational data. In *NeurIPS*, 2013.

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

Cropper, A. and Dumančić, S. Inductive logic programming at 30: a new introduction. *Machine Learning*, 43, 2021.

Cropper, A. and Morel, R. Learning programs by learning from failures. *Machine Learning*, 110(4):801856, 2021.

Cropper, A. and Muggleton, S. Logical minimisation of meta-rules within meta-interpretive learning. In *Inductive Logic Programming - 24th International Conference, ILP*, volume 9046, pp. 62–75. Springer, 2014.

Cropper, A. and Tourret, S. Derivation reduction of metarules in meta-interpretive learning. In *Inductive Logic Programming*, pp. 1–21, 01 2018.

Cropper, A. and Tourret, S. Logical reduction of metarules. *Machine Learning*, 109(7):1323–1369, 2020.

Dai, W.-Z., Xu, Q., Yu, Y., and Zhou, Z.-H. Bridging machine learning and logical reasoning by abductive learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

d'Avila Garcez, A. and Lamb, L. C. Neurosymbolic ai: The 3rd wave, 2020. URL https://arxiv.org/abs/2012.05876.

Dong, H., Mao, J., Lin, T., Wang, C., Li, L., and Zhou, D. Neural Logic Machines. In *International Conference on Learning Representations*. OpenReview.net, 2019.

Evans, R. and Grefenstette, E. Learning explanatory rules from noisy data. *Journal of AI Research*, 61:1–64, 2018.

Fonseca, N., Costa, V., Silva, F., and Camacho, R. On avoiding redundancy in inductive logic programming. In *Lecture Notes in Artificial Intelligence*, volume 3194, pp. 132–146, 09 2004.

Galárraga, L. A., Teflioudi, C., Hose, K., and Suchanek, F. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pp. 413–422. ACM Press, 2013.

Guu, K., Miller, J., and Liang, P. Traversing knowledge graphs in vector space. In *EMNLP*, 2015.

Hudson, D. A. and Manning, C. D. Gqa: A new dataset for real-world visual reasoning and compositional question answering. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6693–6702, 2019a.

Hudson, D. A. and Manning, C. D. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, 2019b.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Jiang, Z. and Luo, S. Neural Logic Reinforcement Learning. In *International Conference on Machine Learning*, April 2019.

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., Bernstein, M. S., and Fei-Fei, L. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *Int. J. Comput. Vision*, 123 (1):3273, May 2017.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):8490, May 2017.

Law, M., Russo, A., and Broda, K. Inductive learning of answer set programs from noisy examples. *Advances in Cognitive Systems*, 7:5776, 2018.

Law, M., Russo, A., and Broda, K. The ilasp system for inductive learning of answer set programs. *in The Association for Logic Programming Newsletter, 2020, Arxiv*, abs/2005.00904, 2020.

Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In *In Proceedings of AAAI15*, 2015.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017.

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

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

Muggleton, S. Inverse entailment and progol. *New Generation Computing*, 13:245–286, 1995.

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

Omran, P., Wang, K., and Wang, Z. Scalable rule learning via learning representation. In *IJCAI*, pp. 2149–2155, 07 2018.

Papakyriakopoulos, O., Hegelich, S., Serrano, J. C. M., and Marco, F. Bias in word embeddings. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, pp. 446457. Association for Computing Machinery, 2020.

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

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

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

Ren, H., Dai, H., Dai, B., Chen, X., Yasunaga, M., Sun, H., Schuurmans, D., Leskovec, J., and Zhou, D. Lego: Latent execution-guided reasoning for multi-hop question answering on knowledge graphs. In *ICML*, 2021.

Robinson, J. A. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):2341, January 1965.

Rocktäschel, T. and Riedel, S. End-to-end differentiable proving. In *NeurIPS*, volume 30, 2017.

Santoro, A., Raposo, D., Barrett, D. G. T., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. A simple neural network module for relational reasoning. *NeurIPS*, pp. 4967–4976, 2017.

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

Socher, R., Chen, D., Manning, C. D., and Ng, A. Y. Reasoning with neural tensor networks for knowledge base completion. 26, 2013.

Tommasi, T., Patricia, N., Caputo, B., and Tuytelaars, T. A deeper look at dataset bias. 2017.

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

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

Zellers, R., Yatskar, M., Thomson, S., and Choi, Y. Neural motifs: Scene graph parsing with global context. In *CVPR*, 2018.

Zimmer, M., Feng, X., Glanois, C., Jiang, Z., Zhang, J., Weng, P., Dong, L., Jianye, H., and Wulong, L. 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 & 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 $\mathcal{D}$ 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 & 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[15] *Turing-complete*[16] 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 $\neg\mathsf{P} \vee \neg\mathsf{Q} \vee ... \vee \neg\mathsf{T}$ or equivalently $\texttt{False} \leftarrow \mathsf{P} \wedge \mathsf{Q} \wedge ... \wedge \mathsf{T}$,
- *definite clauses* which have exactly one positive literal; they are expressed as $\neg\mathsf{P} \vee \neg\mathsf{Q} \vee ... \vee \neg\mathsf{T} \vee \mathsf{U}$ or equivalently as a *Horn rule* $\mathsf{U} \leftarrow \mathsf{P} \wedge \mathsf{Q} \wedge ... \wedge \mathsf{T}$,
- *facts* which can be expressed as $\mathsf{U} \leftarrow \texttt{True}$[17] where U is grounded.

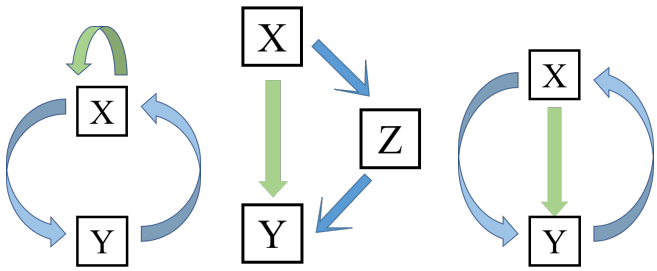
Meta-rules We follow the terminology from the Meta Interpretative Learning literature (Cropper & Tourret, 2020):

Definition 7.1. A *meta-rule* is a second-order Horn clause of the form: $A_0 \leftarrow A_1 \wedge ... \wedge A_m$ 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:

$$\begin{array}{llll} (\sigma) & \mathrm{P}(A,B) & \leftarrow & \mathrm{Q}(B,A) & \textit{permute} \\ (\iota) & \mathrm{P}(A,B) & \leftarrow & \mathrm{Q}(A) & \textit{expand} \\ (\exists) & \mathrm{P}(A) & \leftarrow & \exists B, \mathrm{Q}(A,B) & \textit{existential contraction} \\ (\forall) & \mathrm{P}(A) & \leftarrow & \forall B, \mathrm{Q}(A,B) & \textit{universal contraction} \\ (\nabla) & \mathrm{P}(A) & \leftarrow & \mathrm{Q}(A,A) & \textit{diagonal extract} \\ (\Delta) & \mathrm{P}(A,A) & \leftarrow & \mathrm{Q}(A) & \textit{diagonal fill} \end{array} \tag{17}$$

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



(a) Meta-rule $\mathfrak{A}$ (b) Meta-rule $\mathfrak{B}$ (c) Meta-rule $\mathfrak{C}$

Figure 3. Three types of meta-rules: boxes represent variables, green arrows represent relations for head atoms and blue arrows represent relations for body atoms

A common meta-rule set, which has been used and tested in the literature (Cropper & Tourret, 2020), is the following:

$$\mathcal{R}_{MIL} = \left\{ \begin{array}{llll} (\texttt{Indent}_1) & \mathrm{P}(A) & \leftarrow & \mathrm{Q}(A) \\ (\texttt{DIndent}_1) & \mathrm{P}(A) & \leftarrow & \mathrm{Q}(A) \wedge \mathrm{R}(A) \\ (\texttt{Indent}_2) & \mathrm{P}(A,B) & \leftarrow & \mathrm{Q}(A,B) \\ (\texttt{DIndent}_2) & \mathrm{P}(A,B) & \leftarrow & \mathrm{Q}(A,B) \wedge \mathrm{R}(A,B) \\ (\texttt{Precon}) & \mathrm{P}(A,B) & \leftarrow & \mathrm{Q}(A) \wedge \mathrm{R}(A,B) \\ (\texttt{Postcon}) & \mathrm{P}(A,B) & \leftarrow & \mathrm{Q}(A,B) \wedge \mathrm{R}(B) \\ (\texttt{Curry}) & \mathrm{P}(A,B) & \leftarrow & \mathrm{Q}(A,B,\mathrm{R}) \\ (\texttt{Chain}) & \mathrm{P}(A,B) & \leftarrow & \mathrm{Q}(A,C) \wedge \mathrm{R}(C,B) \end{array} \right\} \tag{18}$$

where the letters $\mathrm{P}, \mathrm{Q}, \mathrm{R}$ correspond to existentially-quantified second-order variables and the letters A, B, 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 ν^n which canonically embeds predicates of arity $i \leq n$ in the space of predicates of arity n:

$$\nu^n : \mathcal{P}^{\leq n} \longrightarrow \mathcal{P}^n \quad \text{s.t.} \quad \nu^n(P)(X_1, \cdots, X_n) = P(X_1, \cdots, X_i) \text{ for } P \in \mathcal{P}^i. \tag{19}$$

Note that the restriction of ν^n on the space of predicates of arity n is identity: $\nu^n \mid_{\mathcal{P}^n} = id_{\mathcal{P}^n}$. Moreover, this projection naturally extends to second-order variables. To ease the notations, we denote below the projection ν^{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 7.2. A *proto-rule* is an extension of a second-order Horn clause of the form: $A_0 \leftarrow A_1 \wedge \ldots \wedge A_m$, where A_0 (resp. each $A_i, i > 0$ is a literal of the form $P(T_1, ..., T_{n_i})$ (resp. $\nu^{n_i}\big(P\big)(T_1, ..., T_{n_i})$ 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 $\mathcal{P}^{n_1} \times \ldots \times \mathcal{P}^{n_m} \to \mathcal{H}$ 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 $\mathcal{P}^{\leq n_1} \times \ldots \times \mathcal{P}^{\leq n_m} \to \mathcal{H}$, for some indices n_i.

The first set of protorules we propose is the following (see Figure 3):

$$\mathcal{R}_0 := \left\{ \begin{array}{lllll} \mathfrak{A}: & P(A) & \leftarrow & \overline{Q}(A,B) \wedge \overline{R}(B,A) \\ \mathfrak{B}: & P(A,B) & \leftarrow & \overline{Q}(A,C) \wedge \overline{R}(C,B) \\ \mathfrak{C}: & P(A,B) & \leftarrow & \overline{Q}(A,B) \wedge \overline{R}(B,A) \end{array} \right\} \tag{20}$$

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 $\mathcal{H}$, $\mathcal{S}$, and $\mathcal{F}$. 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 & Tourret, 2020)), we introduce below a list of classic fragments of $\mathcal{F}$, in decreasing order:

- *connected* fragment, $\mathcal{C}$: connected definite function-free clauses, i.e., clauses in $\mathcal{F}$ 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., $P(A, B) \leftarrow Q(A, C) \wedge R(A, B)$ is connected while the following clause is not $P(A, B) \leftarrow Q(A, C) \wedge R(B)$.
- *Datalog*[18] fragment, $\mathcal{D}$: sub-fragment of $\mathcal{C}$, composed of clauses such that every variable present in the head of the Horn rule also appears in its body. [19]
- *two-connected* fragment, $\mathcal{K}$: sub-fragment of $\mathcal{D}$ composed of clauses such that the variables appearing in the body must appear at least twice.
- *exactly-two-connected* fragment, $\mathcal{E}$: sub-fragment of $\mathcal{K}$ such that the variables appearing in the body must appear exactly twice.
- *duplicate-free* fragment, $\mathcal{U}$: sub-fragment of $\mathcal{K}$ 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:

$$\mathcal{H} \supset \mathcal{S} \supset \mathcal{F} \supset \mathcal{C} \supset \mathcal{D} \supset \mathcal{K} \begin{array}{l} \supset \mathcal{E} \\ \supset \mathcal{U} \end{array} \tag{21}$$

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., $\mathcal{M}^{\{1,2\}}_{\{3\}}$ 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 *resolvent*[20], let us look at an example before stating more formal definitions. Consider the following Horn clauses:

$$\begin{array}{lllll} C_1: & P(X,Y) \vee \neg Q(X) \vee \neg R(X,Y) & \text{or, equivalently,} & P(X,Y) \leftarrow Q(X) \wedge R(X,Y) \\ C_2: & R(X,a) \vee \neg T(X,a) \vee \neg S(a) & \text{or, equivalently,} & R(X,a) \leftarrow S(a) \wedge T(X,a) \end{array}$$

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:

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

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:

$$(P(X,a) \vee \neg Q(X)) \vee (\neg T(X,a) \vee \neg S(a)) \text{ or, equivalently, } P(X,a) \leftarrow Q(X) \wedge T(X,a) \wedge S(a)$$

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.

[18] 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.

[19] 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.

[20] 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.

Definition 7.3.
- A substitution σ 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 θ 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 σ of E there exists a substitution λ such that $\sigma = \theta\lambda$.

Definition 7.4. 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 σ, then the clause $(C_1\sigma \setminus l_2\sigma) \vee (C_2\sigma \setminus l_2\sigma)$ 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, 1995), and derivation (Cropper & Tourret, 2018). As previously pointed out (e.g., (Cropper & 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 & Tourret, 2018), as defined in Definition 7.6.

Let us first define for the following operator for a Horn theory $\mathcal{T}$:

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

Definition 7.5. The *Horn closure* of a Horn theory $\mathcal{T}$ is: $R^*(\mathcal{T}) = \cup_n R_n(\mathcal{T})$.

Definition 7.6. (i) A Horn clause C is *derivationally redundant* in the Horn theory $\mathcal{T}$ if $\mathcal{T} \vdash C$, i.e., $C \in R^*(\mathcal{T})$; it is said to be k-*derivable* from $\mathcal{T}$ if $C \in R^k(\mathcal{T})$.

(ii) A Horn theory $\mathcal{T}$ 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

$$\left\{ \begin{array}{lrcl} C_1: & P(A,B) & \leftarrow & Q(B,A) \\ C_2: & P(A,B) & \leftarrow & Q(A,C), R(C,B) \\ C_3: & P(A,B) & \leftarrow & Q(A,C), R(B,C) \end{array} \right\} \text{ may be D-reduced to either } \{C_1, C_2\} \text{ or } \{C_1, C_3\}$$

Hypothesis Space Recall that $\mathcal{P}$ (resp. $\mathcal{P}_0$) denotes the set of predicates (resp. of initial predicates); and $\mathcal{B}$ 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[21], which could apply to any rule-induction model attached to at meta-rule or proto-rule set:

Definition 7.7. Given a predicate set $\mathcal{P}$, and a meta-rule or proto-rule set $\mathcal{M}$, let us define:

(i) the set $\mathcal{M}_\mathcal{P}$ as the set of all Horn clauses generated by all the possible substitutions of predicates variables in $\mathcal{M}$ by predicates symbols from $\mathcal{P}$.

(ii) the *hypothesis space* $\mathcal{M}[\mathcal{P}]$ generated by $\mathcal{M}$ from $\mathcal{P}$ as the Horn closure of $\mathcal{M}_\mathcal{P}$.

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 $\mathcal{P}_0$; `Equal`, which corresponds to the identity, is intensionally defined on any domain $\mathbb{D}$ by:

$$\texttt{Equal}: \left\{ \begin{array}{lcl} \mathbb{D} \times \mathbb{D} & \rightarrow & \mathbb{B} = \{\texttt{True}, \texttt{False}\} \\ (X, Y) & \mapsto & \left\{ \begin{array}{ll} \texttt{True} & \text{if } X = Y \\ \texttt{False} & \text{if } X \neq Y \end{array} \right. \end{array} \right. \tag{22}$$

[21] 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 $\mathcal{B}$ and a set of predicates $\mathcal{P}$.

Main Result Here is our main result, concerning the investigation of the expressivity of the minimal set $\mathcal{R}_0$:

Theorem 7.8. (i) *Assuming* $\texttt{True} \in \mathcal{P}_0$*, the hypothesis space generated by* $\mathcal{R}_0$ *from* $\mathcal{P}$ *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* $\mathcal{P}$*:*

$$\texttt{True} \in \mathcal{P}_0 \implies \mathcal{R}_0[\mathcal{P}] \supset \mathcal{U}_{\leq 2}^{\{1,2\}}[\mathcal{P}]$$

(ii) *Assuming* $\texttt{True}, \texttt{Equal} \in \mathcal{P}_0$*, the hypothesis space generated by* $\mathcal{R}_0$ *from* $\mathcal{P}$ *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* $\mathcal{P}$*; i.e., in terms of the second order logic fragment:*

$$\texttt{True}, \texttt{Equal} \in \mathcal{P}_0 \implies \mathcal{R}_0[\mathcal{P}] = \mathcal{F}_{\leq 2}^{\{1,2\}}[\mathcal{P}]$$

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

Theorem 7.8 allows us to conclude that $\mathcal{R}_0$ is more expressive than the set commonly found in the literature $\mathcal{R}_{MIL}$, assuming we are working with predicates of arity at most 2:

Corollary 7.9. *Assuming the initial predicates* $\mathcal{P}_0$ *contains only predicates of arity at most* 2*, the hypothesis space generated by* $\mathcal{R}_0$ *given* $\mathcal{P}$ *encompasses the one generated by* $\mathcal{R}_{MIL}$ *as defined in (18).*

$$\forall P \in \mathcal{P}_0, \quad \text{arity}(P) \leq 2 \implies \mathcal{R}_0[\mathcal{P}] \supset \mathcal{R}_{MIL}[\mathcal{P}]$$

Proof. Under our assumption, the meta-rule $(\texttt{Curry})$ in (18) may be disregarded. The remaining meta-rules present in $\mathcal{R}_{MIL}$ have already been examined in Theorem 7.8. $\mathcal{R}_0$ has therefore at least the same expressivity than $\mathcal{R}_{MIL}$. In order to be able to conclude $\mathcal{R}_0$ 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 $\mathcal{R}_0$ not $\mathcal{R}_M IL$. □

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:

$$\mathcal{R}_0^\vee = \left\{ \begin{array}{lllll} \mathfrak{A}: & P(A) & \leftarrow & \big(\overline{Q}(A,B) \wedge \overline{R}(B,A)\big) \vee \overline{S}(A,B) \\ \mathfrak{B}: & P(A,B) & \leftarrow & \big(\overline{Q}(A,C) \wedge \overline{R}(C,B)\big) \vee \overline{S}(A,B) \\ \mathfrak{C}: & P(A,B) & \leftarrow & \big(\overline{Q}(A,B) \wedge \overline{R}(B,A)\big) \vee \overline{S}(A,B) \end{array} \right\} \tag{23}$$

Adopting the minimal set $\mathcal{R}_0^\vee$, in our incremental setting does not affect the expressivity of the hypothesis space compared to when relying on the set $\mathcal{R}_0$ in a non incremental setting. Let us remind an *incremental setting* imposes a predicate symbol to be the head of exactly one rule.

Lemma 7.10. *The minimal proto-rule set* $\mathcal{R}_0^\vee$ *in an* incremental setting *holds the same expressivity as* $\mathcal{R}_0$*.*

Proof. It is straightforward to see that $\mathcal{R}_0^\vee$ is a minimal set.
The hypothesis space generated by $\mathcal{R}_0$ obviously encompasses the hypothesis space generated by $\mathcal{R}_0^\vee$ in an incremental setting. Reciprocally, let us assume n rules instantiated from $\mathcal{R}_0$ are attached to one predicate symbol P. They can be expressed as:

$$\begin{array}{lll} P(X,Y) & \leftarrow & f_1(X,Y) \\ P(X,Y) & \leftarrow & f_2(X,Y) \\ \cdots & \cdots & \cdots \\ P(X,Y) & \leftarrow & f_n(X,Y) \end{array}$$

Recursively, we can define P_i auxiliary predicates in the hypothesis space of $\mathcal{R}_0^\vee$.

$$\begin{array}{lll} P_1(X,Y) & \leftarrow & f_1(X,Y) \\ P_2(X,Y) & \leftarrow & P_1(X,Y) \vee f_2(X,Y) \\ \cdots & \cdots & \cdots \\ P_n(X,Y) & \leftarrow & P_{n-1}(X,Y) \vee f_n(X,Y) \end{array}$$

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 $\mathfrak{I}$ in our experiments, and rely on the following set:

$$\mathcal{R}_* = \mathcal{R}_0^\vee \cup \{\mathfrak{I} : H(X,Y) \leftarrow \overline{F}(Y,X)\}$$

This small—yet non minimal—set of protorules $\mathcal{R}_*$, has the same expressivity as $\mathcal{R}_0$ (characterized in Theorem 7.8)

Lemma 7.11. *The proto-rule set $\mathcal{R}_*$ holds the same expressivity than $\mathcal{R}_0$.*

Proof. It stems from the fact, observed in the proof of Theorem 7.8 that the rule (σ), equivalent to $(\mathfrak{I})$ can be D-reduced from $\mathcal{R}_0$. □

Recursivity The development of Recursive Theory arose first around the Hilberts Program and Gödels 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 ℓ are defined from predicates from layers at most ℓ. Within such stratified space, different restrictions may be considered:

- *recursive-free hierarchical hypothesis space*:, where rules of layer ℓ are defined from rules of strictly lower layer. The proto-rules from $\mathcal{R}_0^\vee$ are therefore restricted to the following form:

  $$P(\cdot) \leftarrow [Q(\cdot) \wedge R(\cdot)] \vee O(\cdot),$$

  with $P \in \mathcal{P}_\ell, Q, R, O \in \mathcal{P}_{<\ell}$.

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

  $$P(\cdot) \leftarrow [Q(\cdot) \wedge R(\cdot)] \vee O(\cdot), \quad \text{with } P \in \mathcal{P}_\ell, Q, R \in \mathcal{P}_{<\ell} \cup \{P\}, O \in \mathcal{P}_{<\ell}$$

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

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

  $$P(\cdot) \leftarrow [Q(\cdot) \wedge R(\cdot)] \vee O(\cdot) \quad \text{with } P \in \mathcal{P}_\ell, Q, R \in \mathcal{P}_{\leq\ell}, O \in \mathcal{P}_{\leq\ell}.$$

  Such space is required for the recursivity 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 7.8 Before we present the proof, let us introduce two additional notations:

- $\mathfrak{R}\ {}_i\odot\mathfrak{R}'$ refers to the resolvent of $\mathfrak{R}$ with $\mathfrak{R}'$, where the i^{th} body member of $\mathfrak{R}$ is resolved with the head of $\mathfrak{R}'$. To illustrate it, consider the following meta-rules:

  $$\begin{array}{ll} \mathfrak{R}: & P(A) \leftarrow Q(A) \wedge R(A,B) \\ \mathfrak{R}': & P(A) \leftarrow Q(A) \wedge R(B,A) \\ \mathfrak{R}'': & P(A,B) \leftarrow Q(B,A) \end{array}$$

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

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

  $$\rho(R)(X) := \exists \circ \sigma(\overline{R}(X,Y)) = \exists Y R(Y) \tag{24}$$

  Here, we identified the permutation clause (σ) (as defined in (17)) with the operation it defines on predicates σ; 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 ρ by:

  $$(\rho): \ P(X) \leftarrow \exists Y R(Y) \tag{25}$$

Proof. The inclusion $\mathcal{R}_0[\mathcal{P}] \subset \mathcal{F}^{\{1,2\}}_{\leq 2}[\mathcal{P}]$ 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: $\mathcal{R}_0[\mathcal{P}] \subset \mathcal{F}^{D,\{1,2\}}_{\leq 2}[\mathcal{P}]$.

It remains to prove the other inclusion; e.g., for (ii), it amounts to $\mathcal{R}_0[\mathcal{P}] \supset \mathcal{F}^{\{1,2\}}_{\leq 2}[\mathcal{P}]$, which boils down to $\mathcal{R}_0[\mathcal{P}] \supset \mathcal{F}^{\{1,2\}}_{\mathcal{P},\leq 2}$ 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 $\mathcal{R}_0[\mathcal{P}] \supset \mathcal{M}_{\mathcal{P}}$. For instance, under the assumption $\texttt{True}, \texttt{Equal} \in \mathcal{P}_0$, 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 $\texttt{True} \in \mathcal{P}_0$, 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 $\mathcal{M}_{\mathcal{P}}$ 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 $\mathcal{R}_0$.

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 (σ) (introduced in 17) can be derived from the proto-rule $\mathfrak{C}$ upon matching the first body in $\mathfrak{C}$ with `True`; therefore, we can examine rules only upon permutation (σ) applied to their body or head predicates.

1. First, let us demonstrate the following inclusion:

$$\texttt{Claim}_1: \quad \texttt{True} \in \mathcal{P}_0 \implies \mathcal{R}_0[\mathcal{P}] \supset \mathcal{U}^{\{1,2\}}_{\mathcal{P},\leq 2} \tag{26}$$

   By the previous observations, we can narrow down to examine a subset of the meta-rules generating $\mathcal{U}^{\{1,2\}}_{\leq 2}$, 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[22].

	Meta-Rules	Reduction	Comment
(i)	$P(A) \leftarrow Q(A) \wedge R(A)$	$\mathfrak{A}_2 \odot \sigma$	$\mathfrak{A}(Q, \sigma(R))$
(ii)	$P(A) \leftarrow Q(A) \wedge R(A,B)$	$\mathfrak{A}_2 \odot \sigma$	$\mathfrak{A}(Q, \sigma(R))$
(iii)	$P(A) \leftarrow Q(A,B) \wedge R(B)$	$\mathfrak{A}$	
(iv)	$P(A) \leftarrow Q(A,B) \wedge R(B,A)$	$\mathfrak{A}$	
(v)	$P(A) \leftarrow Q(A,B) \wedge R(B,C)$	$\mathfrak{A}$	
(vi)	$P(A,B) \leftarrow Q(A) \wedge R(A,B)$	$\mathfrak{A}_2 \odot \sigma$	$\mathfrak{C}(Q, \sigma(R))$
(vii)	$P(A,B) \leftarrow Q(A,C) \wedge R(C,B)$	$\mathfrak{B}$	
$(viii)$	$P(A,B) \leftarrow Q(A,B) \wedge R(B,A)$	$\mathfrak{C}$	
(ix)	$P(A,B) \leftarrow Q(A,B) \wedge R(B,C)$	$\mathfrak{A}_2 \odot \exists$	$\mathfrak{A}(Q, \exists(R))$

   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:

$$\mathcal{P}^\bullet \times \cdots \times \mathcal{P}^\bullet \to \mathcal{P}^\bullet$$

   Let us justify why these clauses are the only clauses in $\mathcal{U}^{\{1,2\}}_{\leq 2}$ 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 ($Q(A) \wedge R(B)$); upon permutation of A and B, and of the body

[22] Note that we are allowed to use clauses $\exists$ and σ in the reduction as we already explained why we can derive them from $\mathcal{R}_0$ in the beginning of the proof of Theorem 7.8.

predicates Q and R, and upon inversion (σ) of the body predicates, the only body clause of arity $(1,2)$ is of the form $Q(A) \wedge R(A,B)$ (denoted (vi) below, which reduces to $\mathfrak{C}$ 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 $\mathfrak{B}, \mathfrak{C}$, (ix), can be obtained by reducing the second body of $\mathfrak{C}$ through $(\exists)$ (as defined in 17). This justifies the remaining clauses, $(vi) - (ix)$.

We can therefore conclude by $\texttt{Claim}_1$.

2. Let us extend (26) by enabling duplicates in the clause:

$$\texttt{Claim}_2: \quad \texttt{True}, \texttt{Equal} \in \mathcal{P}_0 \implies \mathcal{R}_0[\mathcal{P}] \supset \mathcal{K}^{\{1,2\}}_{\mathcal{P},\leq 2} \tag{27}$$

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 (17) can be derived from $\mathcal{R}_0$, under the extra-assumption that $\texttt{Equal}$ is included as background knowledge. This stems from the fact that these meta-rules can be written as follows:

	Meta-rule	Equivalent Form
(∇)	$P(A,A) \leftarrow Q(A)$	$P(A,B) \leftarrow Q(A) \wedge \texttt{Equal}(A,B)$
(Δ)	$P(A) \leftarrow Q(A,A)$	$P(A) \leftarrow Q(A,B) \wedge \texttt{Equal}(A,B)$

Since the equivalent forms above are duplicate-free and two-connected, by (26), $(\nabla), (\Delta)$ can be derived from $\mathcal{R}_0$, which implies $\texttt{Claim}_2$.

3. Removing the assumption of two-connectedness from (27), we claim the following inclusion holds:

$$\texttt{Claim}_3: \quad \texttt{True}, \texttt{Equal} \in \mathcal{P}_0 \implies \mathcal{R}_0[\mathcal{P}] \supset \mathcal{D}^{\{1,2\}}_{\mathcal{P},\leq 2} \tag{28}$$

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

$$P(A) \leftarrow Q(A,A) \wedge R(A,C) \quad ; \quad P(A,B) \leftarrow Q(A,A) \wedge R(A,B) \quad ; \quad P(A,A) \leftarrow Q(A,A) \wedge R(A,B)$$

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 $\texttt{Claim}_3$ follows.

4. In the next step, we extend (28) to the connected Horn fragment;

$$\texttt{Claim}_4: \quad \texttt{True}, \texttt{Equal} \in \mathcal{P}_0 \implies \mathcal{R}_0[\mathcal{P}] \supset \mathcal{C}^{\{1,2\}}_{\mathcal{P},\leq 2} \tag{29}$$

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 $\mathfrak{B}$ once we have matched its second body with $\texttt{True}$. It ensues that $\mathcal{R}_0$ is able to generate the fragment $\mathcal{C}^{\{1,2\}}_2$, as stated in $\texttt{Claim}_4$.

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

$$\texttt{Claim}_5: \quad \texttt{True}, \texttt{Equal} \in \mathcal{P}_0 \implies \mathcal{R}_0[\mathcal{P}] \supset \mathcal{F}^{\{1,2\}}_{\mathcal{P},\leq 2} \tag{30}$$

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

	Meta-rule	Reduction	Comment
(i)	$P(A) \leftarrow Q(A) \wedge R(B)$	$\mathfrak{A}$	
(ii)	$P(A) \leftarrow Q(A,B) \wedge R(C)$	$\mathfrak{A}_1 \odot \exists$	$\mathfrak{A}(\exists(Q), R)$
(iii)	$P(A) \leftarrow Q(A,B) \wedge R(C,D)$	$(\mathfrak{A}_1 \odot \exists)_2 \odot \exists$	$\mathfrak{A}(\exists(Q) \wedge \exists(R))$
(iv)	$P(A,B) \leftarrow Q(A,B) \wedge R(C)$	$\mathfrak{C}_2 \odot \rho$	$\mathfrak{C}(Q, \rho(R))$
(v)	$P(A,B) \leftarrow Q(A) \wedge R(B,C)$	$\mathfrak{B}_2 \odot \sigma$	$\mathfrak{B}(Q, \sigma(R))$
(vi)	$P(A,B) \leftarrow Q(A,B) \wedge R(C,D)$	$\mathfrak{C}_2 \odot (\rho \circ \exists)$	$\mathfrak{C}(Q, \rho \circ \exists(R)))$
(vii)	$P(A,B) \leftarrow Q(A,C) \wedge R(B,D)$	$(\mathfrak{C}_1 \odot \exists)_2 \odot \sigma$	$\mathfrak{C}(\exists(Q) \wedge \sigma(R)))$

Note that the clause/operation ρ, defined in (24,25), 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 (ρ), it can be derived from $\mathcal{R}_0$, we can thereupon

conclude that all the above meta-rules are reducible to $\mathcal{R}_0$, which concludes the proof.

We have therefore proven the following inclusions:

$$\texttt{True}, \texttt{Equal} \in \mathcal{P}_0 \implies \mathcal{R}_0[\mathcal{P}] = \mathcal{F}^{\{1,2\}}_{\mathcal{P},\leq 2} \supset \mathcal{C}^{\{1,2\}}_{\mathcal{P},\leq 2} \supset \mathcal{D}^{\{1,2\}}_{\mathcal{P},\leq 2} \supset \mathcal{K}_2^{\mathcal{P}\{1,2\}} \supset \mathcal{U}^{\{1,2\}}_{\mathcal{P},\leq 2}$$

□

B. ILP Experiment Results

B.1. 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 $\mathcal{R}_*$.

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 *succ*

$$\mathcal{B} = \{\texttt{True}, \texttt{False}, zero(0), succ(0, 1), succ(1, 2), \dots\}.$$

The set of positive examples is

$$\mathcal{P} = \{target(1, 0), target(2, 1), target(3, 2), \dots\}$$

and the set of negative examples is

$$\mathcal{N} = \{target(X, Y) | (X, Y) \in \{0, 1, \dots\}^2\} - \mathcal{P}.$$

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

$$target(X, Y) \leftarrow succ(X, Y).$$

Undirected Edge A graph is represented by a set of *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 $undirected\text{-}edge(X, Y)$ 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

$$\mathcal{B} = \{True, False, edge(a, b), edge(b, c)\}.$$

The corresponding set of positive examples is

$$\mathcal{P} = \{target(a, b), target(b, a), target(b, c), target(c, b)\}$$

and the set of negative examples is

$$\mathcal{P} = \{target(a, c), target(c, a)\}.$$

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

$$target(X, Y) \leftarrow (aux1(X, Y) \wedge edge(Y, X)) \vee edge(X, Y),$$
$$aux1(X, Y) \leftarrow edge(X, Y),$$

where $aux1$ is an invented auxiliary predicate.

Less Than The goal of this task is to learn the $less\text{-}than(X, Y)$ 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

$$\mathcal{P} = \{target(X, Y) | X < Y\}$$

and the set of negative examples is

$$\mathcal{N} = \{target(X, Y) | X \geq Y\}.$$

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

$$target(X, Y) \leftarrow (target(X, Z) \wedge target(Z, Y)) \vee succ(X, Y).$$

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

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

$$\begin{aligned}\mathcal{P} = \{&target(3, [3, 2, 1]), \quad target(2, [3, 2, 1]), \quad target(1, [3, 2, 1]),\\ &target(2, [2, 1]), \quad target(1, [2, 1], \quad target(3, [3, 2]),\\ &target(2, [3, 2]), \quad target(3, [3, 1]), \quad target(1, [3, 1]),\\ &True, False\}\end{aligned}$$

and the set of negative examples is

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

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

$$\begin{aligned}target(X, Y) &\leftarrow (target(X, Z) \wedge aux1(Z, Y)) \vee value(X, Y),\\ aux1(X, Y) &\leftarrow (cons(Y, X) \wedge True) \vee False.\end{aligned}$$

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

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

The corresponding set of positive examples is

$$\mathcal{P} = \{target(a, b), target(b, c), target(c, d), target(a, c), target(a, d), target(b, d)\}$$

and the set of negative examples is

$$\mathcal{P} = \{target(b, a), target(c, b), target(d, c), target(d, a), target(d, b), target(c, a)\}.$$

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

$$target(X, Y) \leftarrow (target(X, Z) \wedge target(Z, Y)) \vee edge(X, Y).$$

Son The goal of this task is to learn the $son\text{-}of(X, Y)$ 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 $father\text{-}of, bother\text{-}of$ and $sister\text{-}of$. An example set of background knowledge is

$$\mathcal{B} = \{True, False, father(a, b), father(a, c), father(a, d), bother(b, c), bother(d, c), sister(c, b)\}.$$

The corresponding set of positive examples is

$$\mathcal{P} = \{target(b, a), target(d, a)\}$$

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

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:

$$\begin{aligned}target(X, Y) &\leftarrow (aux1(X, Y) \wedge father(Y, X)) \vee False\\ aux1(X) &\leftarrow (father(X, Z) \wedge True) \vee brother(X, T)\end{aligned}$$

where $aux1$ is an invented auxiliary predicate.

Grandparent The goal of this task is to learn the $grandparent(X, Y)$ 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 $father\text{-}of$ and $mother\text{-}of$. An example set of background knowledge is

$$\mathcal{B} = \{father(c,b), father(b,a), mother(d,b), mother(e,a), True, False\}.$$

The corresponding set of positive examples is

$$\mathcal{P} = \{target(c,a), target(d,a)\}$$

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

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:

$$target(X,Y) \leftarrow (aux1(X,Z) \wedge aux1(Z,Y)) \vee False,$$
$$aux1(X,Y) \leftarrow (mother(X,Y) \wedge True) \vee father(X,Y),$$

where $aux1$ 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 $is\text{-}adjacent\text{-}to\text{-}a\text{-}red\text{-}node(X)$ relation which is true if node X is adjacent to a red node. Other than the relation $edge$, the background knowledge also consists of facts of relations $colour$ and red, where $colour(X,C)$ if node X has colour C and $red(C)$ if colour of C is red.

An example set of background knowledge is

$$\mathcal{B} = \{True, False, edge(a,b), edge(b,a), edge(d,e), edge(d,f), colour(a,p), red(p), colour(d,q), red(q)\}.$$

The corresponding set of positive examples is

$$\mathcal{P} = \{target(b), target(e), target(f)\}$$

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

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

$$target(X) \leftarrow (edge(X,Z) \wedge aux1(Z,X)) \vee False,$$
$$aux1(X) \leftarrow (colour(X,Z) \wedge red(Z,X)) \vee False,$$

where $aux1$ is an invented auxiliary predicate.

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

An example set of background knowledge is

$$\mathcal{B} = \{True, False, edge(a,b), edge(a,c), edge(c,d), neq(a,b), neq(a,c), neq(a,d), neq(b,c), neq(b,d), neq(c,d)\}.$$

The corresponding set of positive example(s) is

$$\mathcal{P} = \{target(a)\}$$

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

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

$$target(X) \leftarrow (aux1(X,Z) \wedge edge(Z,X)) \vee False,$$
$$aux1(X,Y) \leftarrow (edge(X,Z), \wedge neq(Z,Y)) \vee False,$$

where $aux1$ is an invented auxiliary predicate.

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

An example set of background knowledge is

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

The corresponding set of positive examples is

$$\begin{aligned}\mathcal{P} = \{&target(a,b), target(a,c), target(a,d), target(a,e), target(a,f),\\ &target(b,a), target(b,c), target(b,d), target(b,e), target(b,f),\\ &target(c,a), target(c,b), target(c,d), target(c,e), target(c,f),\\ &target(d,a), target(d,b), target(d,c), target(d,e), target(d,f),\\ &target(e,a), target(e,b), target(e,c), target(e,d), target(e,f),\\ &target(f,a), target(f,b), target(f,c), target(f,d), target(f,e)\\ &target(g,h), target(h,g)\}\end{aligned}$$

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

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:

$$\begin{aligned}target(X,Y) &\leftarrow (target(X,Z) \wedge aux1(Z,Y)) \vee parent(X,Y),\\ aux1(X,Y) &\leftarrow (target(X,Y) \wedge target(Y,X)) \vee parent(X,Y),\end{aligned}$$

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

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

The corresponding set of positive examples is

$$\mathcal{P} = \{target(a), target(b)\}$$

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

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

$$\begin{aligned}target(X) &\leftarrow (aux1(X,Z) \wedge aux1(Z,X)) \vee False,\\ aux1(X,Y) &\leftarrow (aux1(X,Z) \wedge edge(Z,Y)) \vee edge(X,Y).\end{aligned}$$

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 to Red`. An example set of background knowledge is

$$\mathcal{B} = \{True, False, edge(a,b), edge(b,a), edge(b,c), edge(a,d), colour(a,p), colour(b,p), colour(c,q), colour(d,q)\}.$$

The corresponding set of positive examples is

$$\mathcal{P} = \{target(a,b), target(b,a)\}$$

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

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

$$\begin{aligned}target(X,Y) &\leftarrow (edge(X,Z) \wedge aux1(Z,X)) \vee False,\\ aux1(X,Y) &\leftarrow (colour(X,Z) \wedge colour(Z,Y)) \vee False,\end{aligned}$$

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

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

The corresponding set of positive examples is

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

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 $even(X)$ relation which is true if value X is an even number. The background knowledge includes two predicates, one is $zero(X)$, which is true if X is 0, another one is $succ(X, Y)$, which is true if Y is the next value of X. An example set of background knowledge is

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

The corresponding set of positive examples is

$$\mathcal{P} = \{target(0), target(2), target(4), \dots\}$$

and the set of negative examples is

$$\mathcal{N} = \{target(1), target(3), target(5), \dots\}.$$

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

$$\begin{aligned} target(X) &\leftarrow (zero(X) \wedge aux1(Z, X)) \vee zero(X), \\ aux1(X, Y) &\leftarrow (aux1(X, Z) \wedge aux1(Z, Y)) \vee aux2(X, Y), \\ aux2(X, Y) &\leftarrow (succ(X, Z) \wedge succ(Z, Y)) \vee False, \end{aligned}$$

where $aux1$ and $aux2$ 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 $buzz(X)$ relation which is true if value X is divisible by 5. The background knowledge consists of 4 predicates: $zero(X)$ is true if X is 0, $succ(X, Y)$ is true if Y is the next value of X, $pred1(X, Y)$ is true if $Y = X + 3$, $pred2(X, Y)$ is true if $Y = X + 2$. An example set of background knowledge is

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

The corresponding set of positive examples is

$$\mathcal{P} = \{target(0), target(5), \dots\}$$

and the set of negative examples is

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

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

$$\begin{aligned} target(X) &\leftarrow (aux1(X, Z) \wedge pred2(Z, X)) \vee zero(X), \\ aux1(X, Y) &\leftarrow (aux2(X, Z) \wedge pred1(Z, Y)) \vee False, \\ aux2(X, Y) &\leftarrow (True \wedge aux3(Y)) \vee zero(X), \\ aux3(X) &\leftarrow (aux1(X, Z) \wedge pred2(Z, X)) \vee zero(X), \end{aligned}$$

where $aux1$, $aux2$ and $aux3$ are invented auxiliary predicates.

Fizz The goal of this task is to learn the $fizz(X)$ relation which is true if value X is divisible by 3. The background

knowledge is the same with $Even - Odd$ task. The corresponding set of positive examples is

$$\mathcal{P} = \{target(0), target(5), \dots\}$$

and the set of negative examples is

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

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

B.2. Full ILP Experimental Results

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

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.

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

About task `Length`: 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; 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 instantiated auxiliary predicates by proto-rule per layer is by default set to 1 in all our experiments, as we intended to share the same hyperparameter 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 task `Even-Odd`: As mentioned above, task `Even-Odd` corresponds, for our method, to the same task as 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.

B.3. Limitations of LRI (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 7 demonstrates how the evaluation success rate decreases if LRI is trained with 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 with a set of more generic meta-rules. This phenomenon is more accentuated for more complex tasks. For easy tasks, like `Predecessor`, the performance did not change.

Table 7. LRI's performance with an increased number of rule templates. Measured in percentage of successful runs over 10 runs using soft evaluation.

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

B.4. Comparison with traditional ILP methods

Large dataset We test if traditional ILP methods can handle large datasets with a large number of objects. As far as we know, traditional ILP methods can not process input knowledge by mini-batches: users must provide all background knowledge and examples to the solver before it starts running. Moreover, they do not utilize GPUs to speed up solving. In contrast, `HRI` can sample a small subgraph from the large dataset in each training iteration, and can make use of GPU to accelerate training easily. Thus, it can scale better to large datasets. To verify this, we compare `HRI` with Popper and ILASP3. To have a fair comparison, we try to give the two baseline methods a similar hypothesis space as `HRI`, including the hierarchical bias and meta-rule templates.

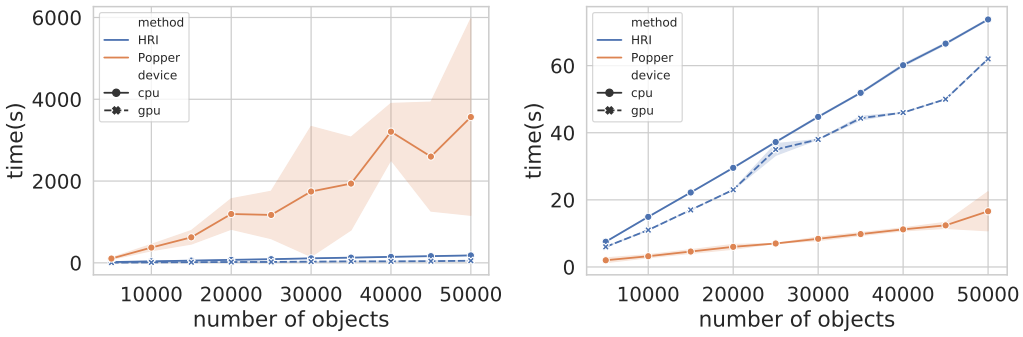
However, once we give a similar hypothesis space as `HRI`, ILASP3 can not find a correct solution within 30 minutes even for a graph with 10 objects in the `Grandparent` task, therefore we did not test it further in larger dataset with such a general bias.

As shown in Figure 4, Popper costs much more time than `HRI` for the `Grandparent` task, which needs predicate invention. Although Popper performs better for the `Undirected Edge` task, note that this is a very simple task, which does not require predicate invention, and Popper can verify the solution straightforwardly once the corresponding meta-rules are given. Besides, for `Member` and `Connectedness`, in the case that the number of objects is 10,000, Popper can not find correct solutions within 30 minutes while `HRI` can be successful within 5 minutes.

Noisy Data Finally, as mentioned in the main paper, we compared `HRI` with ILASP3 (Law et al., 2018) [23] and ∂ILP (Evans & Grefenstette, 2018) in a noisy data setting by ranging the ratio of mislabeled target training examples from 0% to 90%, as shown in Figure 2.

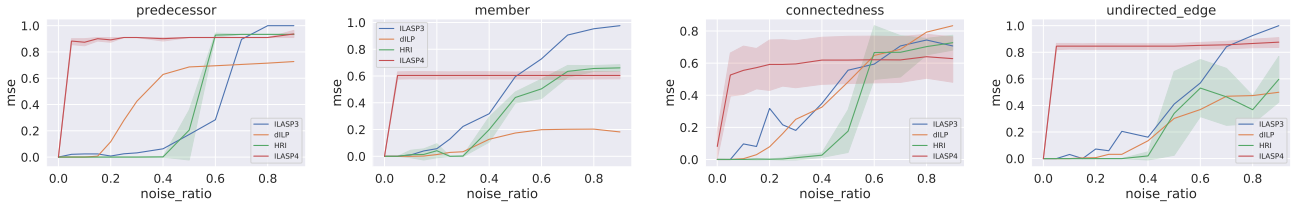
We also ran additional comparison with ILASP4 (Law et al., 2020), which is a successor of ILASP3 computationally less expensive, since it only considers necessary (and not sufficient) constraints. Since we did not find any reference for hyperparameters and biases chosen for ILASP4, we used similar ones than for the ones the authors indicate for ILASP3 (Law et al., 2018). We used a time limit of 300s and added 2 to their 'maxbl' parameter, to increase the hypothesis space, although similar results were obtained with identical maxbl. Based on these parameters, the preliminary results as displayed in Figure Figure 5 (averaged on 5 runs) suggest that ILASP4 is quite sensitive to noise. This should be further investigated, with different parameters and language bias since we may missed the best settings parameters.

[23] Experimental results of ILASP3 and ∂ILP come from their own paper (Law et al., 2018; Evans & Grefenstette, 2018).



(a) Grandparent (b) Undirected Edge

Figure 4. Time(s) to find the correct solution with respect to the number of objects in dataset



(a) Predecessor (b) Member (c) Connectedness (d) Undirected Edge

Figure 5. Mean Squared Error (MSE) w.r.t. different noise ratios.

C. Visual Genome Experiments

For these experiments, we use a dataset called GQA (Hudson & 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 KGs, leading to 1.9M facts, 1.4M constants, and 2100 predicates. Following (Yang & Song, 2020), we filter this dataset to remove the predicates that appear less than 1500 times and to focus on the top 150 objects.

In the multi-task setting, we train 150 models to provide a logic explanation to the top 150 objects. For the evaluation metrics, we use recall @1 (R@1) and recall @5 (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 ℓ may be formed with predicates from any layer 0 to ℓ in $\mathcal{P}_\ell^\downarrow$), the trained model may contain 1 to n_L layers (without counting the target predicate). Given the soft unification computation in (11), 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.

After training `HRI`, we compare the semantic space underlying these embeddings. More precisely, we use cosine similarity to measure distances between all pairs of background embeddings, sort, and select the top 10 closest pairs. As Table 8 suggests, the fine-tuned embeddings pairs have akin similarities; this initialization choice may therefore help with the performance to some extent.

Table 8. Top 10 closest embeddings from GPT2 and our trained multi-task model.

	Top 10 pairs of similar embeddings
GPT2	(bag, backpack), (arrow, apple), (airplane, air), (backpack, airplane), (apple, 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 shows some extracted learned rules for the multitask model.

Table 9. Examples of extracted rules from multi-task model

Target	Rules
wrist	$wrist(X) \leftarrow (watch(Y) \wedge on(Y, X)) \vee False$
person	$person(X) \leftarrow (on(X, Y) \wedge bench(Y)) \vee wearing(X, T)$
vase	$vase(X) \leftarrow (aux(X, Y) \wedge flowers(Y)) \vee False$ $aux(X, Y) \leftarrow (True \wedge on(Y, X)) \vee with(X, Y)$
logo	$logo(X) \leftarrow (aux_0(X) \wedge True) \vee aux_1(X)$ $aux_1(X) \leftarrow (on(X, Y) \wedge laptop(Y)) \vee False$ $aux_0(X) \leftarrow (on(X, Y) \wedge kite(Y)) \vee False$

D. Design Choices

Operation Choices In our implementation, the default choice is sum for POOL, $\min$ for AND, $\max$ for OR. In Table 10, 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 probabilistic sum (probsum)[24] for OR.

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

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

Ablation Study Table 11 shows some ablation study about hierarchical and incremental bias on a few ILP tasks, and with or without the adoption of proto-rules. In the case of no-protorules, we use the set of all templates used in LRI Campero et al. (2018); the first line correspond to LRI model with a unified set of templates. As displayed in the table, there is a clear improvement of the performance upon adoption of these different biases.

Extended Gumbel Noise We adopt an extended version of Gumbel noise defined as follows:

$$G = -g_1 \log\left(-\log\left(g_2 U\right)\right),$$

[24] $probsum(v_1, v_2) = v_1 + v_2 - v_1 * v_2$

Table 11. Percentage of successful runs among 10 runs. Ablation study of incremental and hierarchical biases.

protorules	incremental	hierarchical	Adjacent To Red	Member	Cyclic	Two Children
✗	✗	✗	0	30	0	0
✓	✓	✗	60	10	70	80
✓	✓	✓	100	100	100	100

Table 12. Percentage of successful runs among 20 runs for some ILP tasks, with different settings about the extended Gumbel noise.

g_1	g_2	decay mode	Adjacent to Red	Member	Cyclic	Two Children
1	0.3	linear	100	100	100	100
	0.5		100	95	100	95
	0.3	exponential	100	85	95	80
	0.5		60	95	95	40
0.3	1	linear	95	85	85	65
0.5			65	75	35	25
0.3		exponential	30	80	50	0
0.5			0	0	0	0
0	\	\	60	20	75	30

where U is sampled according to a uniform distribution $\mathcal{U}([0,1])$, and with a *Gumbel scale* $g_1 \in (0, +\infty)$ and a *Gumbel factor* $g_2 \in (0, +\infty)$.

As illustrated in Table 12, using a linearly-decaying Gumbel factor and fixed Gumbel scale (instead of a decaying Gumbel scale and fixed Gumbel factor as commonly adopted) was experimentally beneficial for the tasks we considered. The last row in Table 12 shows the significance of adding a Gumbel noise for hard tasks like `Member` or `Two Children`.

In our model, this noise term is added to the cosine-similarity scores in (13), before applying a softmax, in order to lead to the unification scores. More precisely:

$$\alpha_{P_i \mathrm{B}_i} = \frac{\exp\left(\left(\cos(\boldsymbol{\theta}_{P_i}, \boldsymbol{\theta}_{\mathrm{B}_i}) + G_{(i,i)}\right)/\tau\right)}{\sum_{P_j} \exp\left(\left(\cos(\boldsymbol{\theta}_{P_j}, \boldsymbol{\theta}_{\mathrm{B}_i}) + G_{(j,i)}\right)/\tau\right)}, \tag{31}$$

Let us examine the behavior of this decay to justify our approach. On the one hand, making g_2 tends towards 0 would make the Gumbel term G tends towards $-\infty$, which seems unreasonable. However, if we keep g_2 above a moderately small value (as for instance 10^{-4} after 3000 iterations steps in our experiments, with a linear decay), reducing g_2 amounts to reduce the range and variance of the applied noise. In our context, note that because of the softmax, only this range matters for the unification scores to be affected. Let us point out that, at the extreme, adding a constant noise term to each of the similarity scores (as in the evaluation task), would not affect the unification scores (equivalent to having G being 0).

For instance, with g_2 going from 0.3 to $0.3 * (1 - \frac{3999}{4000})$, the range of the Gumbel noise G is roughly going from (for U between 0.0001 to 1) $[-2.543, -0.186]$ to $[-3.045, -2.251]$. In that sense, our choice would encourage the similarity scores to be different enough, and with g_2 small enough (> 0), would not affect the unification scores, similarly as a noise going to 0.

E. Hyperparameters

We list relevant generic and task-specific hyperparameters used for our training method in Tables 13 and 14, respectively. In Table 14, the hyperparameters *train-num-constants* and *eval-num-constants* represent the number of constants during training and evaluation, respectively. We keep the values of the two hyperparameters for each task the same as those used in Campero et al. (2018). Note that our model does 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.

Table 13. Generic hyperparameters for all tasks.

Hyperparameter	recursivity	fuzzy-and	fuzzy-or	similarity	lr	lr-rules
Value	full	min	max	cosine	0.01	0.03
Hyperparameter	temperature	Gumbel-Scale	Gumbel-Factor	Gumbel-factor-decay-mode		
Value	0.1	1.0	0.3	linear		

Table 14. Specialized hyperparameters 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 used the same generic hyperparameters as given in Table 13 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 hyperparametes are given in Table 15.

Table 15. 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 13 except that we did not use recursivity and decreased the learning rate to 0.005. Additional hyperparameters are given in Table 16.

Table 16. Reinforcement learning hyperparameters.

Hyperparameter	Value
Max depth	6
Train-steps	6
Train-num-constants	5
Temperature of softmax policy	0.01
GAE λ	0.9
γ	0.99
Trajectory per update	5
PPO ϵ-clipping	0.2
GRU hidden neurons (critic)	64

Sensitivity to Hyperparameters To evaluate the sensitivity of our model to hyperparameters, we tested other hyperparameter choices on some ILP tasks. We refer to Table 14 for the results obtained with default hyperparameters, presented in Table 13. As Table 17 suggests, both smaller inference steps, or smaller depth will affect performance; this performance decreases naturally, 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 requires it.

Table 17. Percentage of successful runs among 10 runs using soft evaluation, obtained by models trained with different hyperparameter choices. Here, s_t and s_e are default inference steps used in training and evaluation, shown in Table 14.

Task	default	Inference-steps (train-steps, eval-steps)			
		(s_t-2, s_e-2)	(s_t-1, s_e-1)	(s_t+1, s_e+1)	(s_t+2, s_e+2)
Adjacent to Red	100	90	100	100	100
Member	100	100	100	100	100
Cyclic	100	90	90	90	100
Two Children	100	80	100	100	100

Task	Similarity			Recursivity		Max-depth			
	L1	L2	scalar-product	none	iso-recursive	1	2	3	5
Adjacent to Red	10	80	40	100	100	0	90	100	100
Member	60	100	80	0	100	50	100	100	100
Cyclic	60	80	50	90	100	10	60	80	90
Two Children	0	60	10	90	100	0	70	70	100

F. Limitations and Potentials of `HRI`

Expressivity Limitations As made explicit in Theorem 7.8, our model corresponds to rule-induction with function-free definite Horn clauses. 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 clauses. In the recursive case, the number of instantiated rules from the proto-rule set (by default set to 1) also affects the expressivity; this may be seen in task `Length` in ILP, where two rules should be initiated per proto-rule to sufficiently widen the hypothesis space.

The expressivity of our model could be extended in 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.

$$\texttt{Claim}: \text{By adding the zero-ary predicate} \mathfrak{Z} \text{ to } \mathcal{R}_0, \text{the hypothesis space reaches the fragment} \mathcal{F}^{\leq 2}_{\mathcal{P},\leq 2} \tag{32}$$

where $\mathfrak{Z}: \ \mathrm{P}() \leftarrow \overline{\mathrm{Q}}(A, B)$.

+ Enabling more body atoms:[25]

$$\texttt{Claim}: \text{Enabling 3 or 4 body atoms would result in the same hypothesis space.} \tag{33}$$

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 give a visual illustration of these reductions[26], in Figure 6 (resp. Figure 7). Red arrows denote head predicates; full grey arrows depict two arrows that can be reduced into the dotted grey arrow to lower the number of body atoms by introducing an auxiliary predicate.

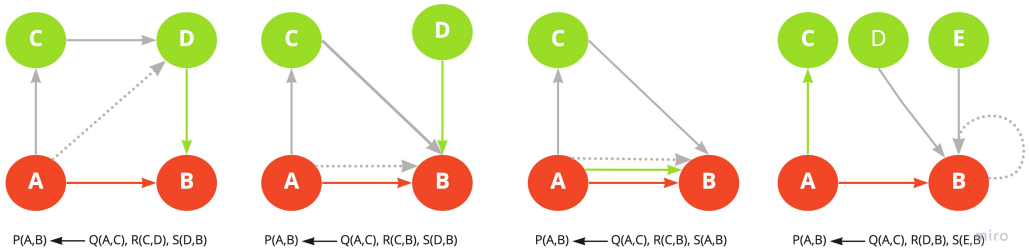


Figure 6. Reduction of meta-rules with 3 body atoms

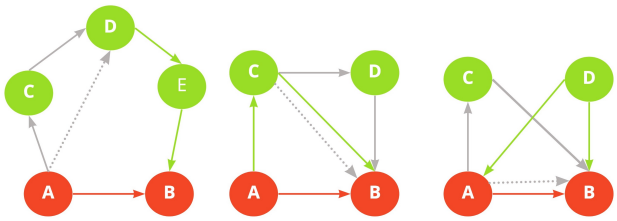


Figure 7. Reduction of meta-rules with 4 body atoms

In contrast, as illustrated in Figure 8 some rules with 5 body clauses are not reducible to 2 body clauses, such as:

$$P(A, B) \leftarrow Q(A, C), R(A, D), S(B, C), T(B, D), U(C, D). \tag{34}$$

As mentioned in Cropper & Tourret (2020), $\mathcal{F}^{\{1,2\}}_{\leq 5}$ is therefore not D-reducible to $\mathcal{F}^{\{1,2\}}_{\leq 2}$. However, allowing a 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 $\mathcal{C}^{\{1,2\}}_{\leq 5}$:

$$\texttt{Claim}: \mathcal{F}^{\{1,2,3\}}_{\leq 2}[\mathcal{P}] \supset \mathcal{F}^{\{1,2\}}_{\leq 5} \tag{35}$$

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.

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

[26] 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.

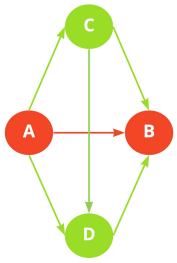


Figure 8. Irreducible Meta-Rule with 5 body atoms

Potentials In contrast to most previous classic ILP or differentiable ILP works, we hypothesize that `HRI` could be particularly suited for continual learning in semantically-richer domains, such as in RL scenarios where an interpretable logic-oriented higher-level policy seems pertinent (e.g., autonomous driving). Here are some arguments to support this belief:

+ In contrast to some previous works, `HRI` is independent of the number of predicates, which may vary between training and testing.
+ As our experiments in Visual Genome suggest, `HRI` can be bootstrapped by semantic or visual priors, to initialize the predicates.
+ In semantically-rich domains, the complexity of `HRI` 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), $d << \mid \mathcal{P}_0 \mid$; such inequality implies a lower complexity for `HRI` than common methods which have to learn one coefficient per predicate.
+ The embedding-based approach enables to reason by *analogy*, and possibly quickly generalize to new predicates. For instance, if the model has learned the rule $OverTake() \leftarrow \neg(P(X) \wedge OnNextLane(X))$, and the embedding θ_P is learned to be close to θ_{Car}, the rule would generalize to include Truck, assuming $\theta_{Car} \sim \theta_{Truck}$, despite having never seen the new predicate Truck during training.