GEOILP: A SYNTHETIC DATASET TO GUIDE LARGE-SCALE RULE INDUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Inductive logic programming (ILP) is a machine learning approach aiming to learn explanatory rules from data. While existing ILP systems can successfully solve small-scale tasks, large-scale applications with various language biases are rarely explored. Besides, it is crucial for a large majority of current ILP systems to require expert-defined language bias, which hampers the development of ILP towards broader utilizations. In this paper, we introduce GeoILP, a large-scale synthetic dataset of diverse ILP tasks involving numerous aspects of language bias. The ILP tasks are built from geometry problems, at the level from textbook exercise to regional International Mathematical Olympiad (IMO), with the help of a deduction engine. These problems are elaborately selected to cover all challenging language biases, such as recursion, predicate invention, and high arity. Experimental results show that no existing method can solve GeoILP tasks. In addition, along with classic symbolic-form data, we provide image-form data to boost the development of the joint learning of neural perception and symbolic rule induction.

1 INTRODUCTION

Inductive logic programming (ILP), at the intersection of machine learning (ML) and symbolic artificial intelligence, learns hypotheses from background knowledge and examples (Muggleton & De Raedt, 1994; Muggleton et al., 2012; Cropper et al., 2020a; Cropper & Dumančić, 2022; Zhang et al., 2023). ILP adopts logical formulae to represent knowledge, examples, and hypotheses uniformly. The most fascinating merit of ILP, differing from other ML approaches, is the ability to learn highly interpretable hypotheses, which reveals a potential way toward human-comprehensible, controllable, and trust-worthy artificial intelligence.

Classic symbolic ILP are based on discrete search, suffering from the combinatorially growing search space and thus restricting to small-scale scenes. To alleviate this obstacle, symbolic methods require user-defined language bias to limit searching, which is markedly crucial for efficiency (Cropper & Dumančić, 2022). Such hand-crafted work is more or less the same as the feature engineering (Khalid et al., 2014) in other ML tasks at the early stage, requiring certain expert knowledge and considerably many troublesome trial and error. In the modern ML community, feature engineering is often superseded by automatic feature extractors, such as various neural networks, which achieve amazing success in large-scale applications (e.g., GPT-4 (Achiam et al., 2023)). Accordingly, thoroughly turning hand-crafted determination of language bias into automatic language bias discovery is a promising direction toward broader applications of ILP.

Modern neural-symbolic ILP relaxes the hypotheses space into a continuous space and leverage gradient-based optimization techniques to induce solutions, from which interpretable rules can be extracted. Despite not requiring an elaborated language bias, existing neural-symbolic methods are limited to a relatively small hypotheses space, presuming low-arity predicates, function-free clauses, and few rule's body atoms (Glanois et al., 2022). Scaling up to large-scale scenarios is also the major challenge for this line of work.

However, large-scale ILP datasets are lacking in evaluating more powerful methods and guiding enhancement. Existing datasets are either small or relatively large but lack reference hypotheses. Our goal is to construct a large-scale ILP dataset, providing reference hypotheses to guide the resolution of present limitations in ILP, which would lead ILP to an expert-free learning paradigm (like modern neural learning) and exceedingly broader utilization. Furthermore, we aim to evaluate ILP systems without much expert priors (like other ML tasks), i.e., training & testing without excessive user-defined bias.

Therefore, we construct GeoILP, a large-scale dataset synthesized from *plane geometry* rules that help generate reference hypotheses involving various language biases. We first adopt a symbolic deduction engine to obtain target examples from the rules and determine the background knowledge and hypotheses by traceback from the examples. We also consider the noisy and multi-task settings, which are closer to real-world applications and actively studied in other ML tasks.

In summary, GeoILP contains 836 single-tasks and 207 multi-tasks. The predicate arity is up to 8, the number of variables in a rule is up to 12, and the number of body atoms is up to 9. Overall, 85% single-tasks and 50% multi-tasks leverage hypothesis with the number of rules ranging from (at least) 10 to 100 (refer to section 5.3 for details). Besides, the language biases also involve argument symmetry, constraints, different types of recursion, and predicate invention.

We conduct experiments on existing applicable methods, which show that GeoILP is completely unreachable.

In addition, GeoILP provides image-form background knowledge, which requires jointly training a perception network, transforming the raw sensory input (image) into symbolic knowledge, and an ILP system inducing the hypothesis. The breakthrough for such joint learning, which remains less explored, would be a breakthrough for the whole artificial intelligence community (Cropper & Dumančić, 2022).

2 RELATED WORK

2.1 ILP METHODS

Symbolic methods search in the hypotheses space defined by language biases. Among these, notable methods include FOIL (Quinlan, 1990), Progol (Muggleton, 1995; Muggleton & Bryant, 2000), TILED (Blockeel & De Raedt, 1997), ALEPH (SRINIVASAN, 2001), Metagol (Muggleton et al., 2015), ILASP (Law et al., 2018; 2020). These methods suffer from combinatorially growing hypotheses, noisy data, and inefficient predicate invention. Popper (Cropper & Morel, 2021a) is a modern symbolic ILP system, which is, to the best of our knowledge, the only symbolic system capable of simultaneously learning recursive rules, inventing predicates (Cropper & Morel, 2021b), handling noise (Hocquette et al., 2024), and scaling better, though still very expensive to do these.

Neural-symbolic methods or differentiable methods, make continuous relaxation of the discrete hypotheses space and induce solutions by minimizing loss function via gradient-based optimizer. While the early-stage methods require user-defined language templates task-by-task to restrict hypotheses (Rocktäschel & Riedel, 2017; Campero et al., 2018), the following works tend to automatically deal with more general language biases (Evans & Grefenstette, 2018; Si et al., 2019; Glanois et al., 2022). As learning interpretable solutions is the outstanding property of ILP, the methods that cannot produce human-readable rules are out of the scope of this paper (e.g., Dong et al. (2019)).

2.2 ILP DATASETS

Real-world datasets Real-world datasets collect background knowledge and examples from real-world observation. The application scenarios cover knowledge base completion (Bordes et al., 2013; Toutanova & Chen, 2015; Yang et al., 2017; Hudson & Manning, 2019), drug design (Inoue et al., 2013; Tamaddoni-Nezhad et al., 2006), ecology (Bohan et al., 2017), etc. The main demerit of real-world datasets is lacking reference hypotheses. Consequently, an ILP system failing on these datasets would have little idea about where to improve.

Synthetic datasets Synthetic datasets covering mathematical formal systems (Evans & Grefenstette, 2018), grammar learning (Muggleton et al., 2014; Law et al., 2019), games (Cropper et al., 2020b), program analysis (Sivaraman et al., 2019; Bartha & Cheney, 2020), etc. They can provide reference hypotheses to guide resolving the limitations of ILP systems. However, current synthetic datasets are small-scale, whose hypotheses typically contain less than 10 rules, and have already been solved by existing ILP. Our work extends this line of work to much larger scenarios.

3 Background

We first introduce necessary logic notions and then define inductive logic programming (ILP). Further terminology is illustrated in the next section as well.

3.1 Logic preliminaries

We assume basic knowledge about first-order logic.

Horn clause Every formula in first-order language can be transformed into its semantically equivalent *conjunctive normal form*, a conjunction of clauses. A *clause* is a disjunction of literals. A *literal* is an atom (*positive literal*) or its negation (*negative literal*). An atom is called *ground* if it contains no variable. *Horn clause* is a widely used subset of clauses that allow at most one positive literal. Horn clause involve *facts*, which are atoms, and *rules* that can be semantically equivalently represented as (assumed function-free here)

$$\mathrm{H}(\boldsymbol{X}) \leftarrow \mathrm{B}_1(\boldsymbol{X}) \wedge \mathrm{B}_2(\boldsymbol{X}) \wedge \cdots \wedge \mathrm{B}_k(\boldsymbol{X})$$

where $\boldsymbol{X}$ denotes a vector of variables, the atom $\mathrm{H}(\boldsymbol{X})$ is the *head* atom of the rule, and the atoms $\mathrm{B}_1(\boldsymbol{X}), \mathrm{B}_2(\boldsymbol{X}), \ldots, \mathrm{B}_k(\boldsymbol{X})$ are the *body* atoms of the rule. A *program* is a set of Horn clauses. We define *rule size* as the number of atoms in a rule and *program size* as the sum of rule size.

Note that the variables in a clause are implicitly quantified by universal quantifiers that are supposed to be placed at the beginning. The variables appearing only in the body but not the head are called *existentially quantified*. [1]

Forward chaining *Forward chaining* can be used to deduce all the true ground facts from given rules and background facts. Formally, given a set of ground atoms $\mathcal{A}$ and a Horn rule set $\mathcal{R}$, the immediate consequence through one-step forward chaining is defined as the set

$$\mathrm{con}_{\mathcal{R}}(\mathcal{A}) = \mathcal{A} \cup \left\{ \alpha \;\middle|\; \alpha \leftarrow \alpha_1, \ldots, \alpha_k \in \mathrm{ground}(\mathcal{R}), \bigwedge_{i=1}^{k} \alpha_i \in \mathcal{A} \right\}$$

where $\mathrm{ground}(\mathcal{R})$ consists of all the *ground rules* instantiated from $\mathcal{R}$. Then, we recursively define the consequence through t steps $\mathrm{C}_{\mathcal{R},t}(\mathcal{A})$

$$\mathrm{C}_{\mathcal{R},0}(\mathcal{A}) = \mathcal{A}, \quad \mathrm{C}_{\mathcal{R},t+1}(\mathcal{A}) = \mathrm{con}_{\mathcal{R}}\big(\mathrm{C}_{\mathcal{R},t}(\mathcal{A})\big)$$

We say the *fix point* is reached at step T if T is the smallest natural number satisfying $\mathrm{C}_{\mathcal{R},T}(\mathcal{A}) = \mathrm{C}_{\mathcal{R},T+1}(\mathcal{A})$, and $\mathrm{C}_{\mathcal{R},T}(\mathcal{A})$ is the set of all the consequences of the forward chaining.

3.2 Inductive logic programming

We adopt the most popular ILP setting *learning from entailment* (LFE) (Cropper & Dumančić, 2022). The training data is a tuple $(\mathcal{B}, \mathcal{E}^+, \mathcal{E}^-)$ of *background knowledge* $\mathcal{B}$, *positive examples* of the concept $\mathcal{E}^+$, and *negative examples* of the concept $\mathcal{E}^-$. $\mathcal{E}^+, \mathcal{E}^-$ are sets of ground atoms relevant to the target predicate we want to learn. $\mathcal{B}$ is a set of clauses that act as background knowledge (BK), typically a set of ground atoms irrelevant to the target predicate (rules can also be in BK). The goal of ILP is to induce a *hypothesis* $\mathcal{H}$, consisting also of clauses, satisfying the following conditions

$$\forall e \in \mathcal{E}^+, H \cup B \vDash e \qquad \text{(completeness)}$$
$$\forall e \in \mathcal{E}^-, H \cup B \nvDash e \qquad \text{(consistency)}$$

The *completeness* condition states that the hypothesis and BK entail all positive examples. The *consistency* condition states that the hypothesis and BK do not entail any negative examples.

[1] Take an example to explain the name, $\forall X \forall Y \forall Z\ \mathrm{H}(X,Y) \leftarrow \mathrm{B}_1(X,Z) \wedge \mathrm{B}_2(Z,Y)$ is equivalent to $\forall X \forall Y\ \mathrm{H}(X,Y) \leftarrow \exists Z\ (\mathrm{B}_1(X,Z) \wedge \mathrm{B}_2(Z,Y))$. Z is existentially quantified in this rule.

For example, given

$$\mathcal{B} = \{\text{Father}(\text{John}, \text{Mary}), \text{Father}(\text{Tom}, \text{John})\}$$
$$\mathcal{E}^{+} = \{\text{Grandfather}(\text{Tom}, \text{Mary})\}$$
$$\mathcal{E}^{-} = \{\text{Grandfather}(\text{Mary}, \text{Tom}), \text{Grandfather}(\text{John}, \text{Mary})\}$$

, an ILP task learner may learn the following hypothesis $\mathcal{H}$ (upper italic letters denote variables)

$$\text{Grandfather}(X, Y) \leftarrow \text{Father}(X, Z) \wedge \text{Father}(Z, Y)$$

With the learned hypothesis, an automated theorem prover can derive all the facts regarding the *target predicate* Grandfather from the background facts regarding the relation $Father$.

4 LIMITATIONS OF CURRENT ILP

In this section, we identify the critical limitations of current ILP that impede the development of broader applications. Our proposed dataset is intended to cover all these challenges and is thus a good testbed for elaborating more sophisticated rule induction systems.

4.1 HAND-CRAFTED LANGUAGE BIAS

Language bias is used to limit the hypothesis space in symbolic ILP. As calculated by Cropper & Morel (2021a), the number of possible hypotheses grows combinatorially fast.

Without carefully human-determined language bias, such as the predicates allowed to appear in the rule's head, the predicates allowed to appear in the rule's body, enabling recursion or not, enabling predicate invention or not, the maximum number of clauses allowed in a hypothesis, the maximum number of unique variables in a clause, the maximum number of body atoms in a clause, the number of allowed existentially quantified variables, the maximum times a predicate can appear in a rule, symbolic ILP tends to be extremely slow, even useless (Cropper & Dumančić, 2022). Determining a good language bias is onerous and requires a vast amount of trial and error.

Below, we introduce the most dominant language biases, which notably increase hypothesis space and should thus be completely automatically determined by ILP systems.

Predicate arity Real-world relations may involve several entities. For instance, the dyadic relation $\text{GoodAt}(\text{student}, \text{math})$ asserts that a student is good at math (subject), while the triadic relation $\text{Course}(\text{teacher}, \text{math}, \text{student})$ asserts that a teacher teaches math to a student. However, current neuro-symbolic methods typically support arity lower than two (Evans & Grefenstette, 2018; Campero et al., 2018; Glanois et al., 2022). Several symbolic methods can support arbitrary arity but exceedingly increase search complexity (Cropper & Morel, 2021a).

Argument symmetry Argument symmetry may exist for predicates. For example, if John is Mary's cousin, then Mary must also be John's cousin. As this example, argument symmetry can be represented as a Horn rule, yet complex symmetry may yield too many rules. For instance, the triadic atom asserting whether 3 people queue in a straight line evaluates to the same truth value if permuting all 3 arguments (any 3 people), which yields $\binom{3!}{2} = 15$ Horn rules. To the best of our knowledge, there is no specialized way to learn compact representations for argument symmetry.

Predicate constraint Atoms' truth value may be forced to be opposite under some constraints. Asymmetry constraint can be considered as Horn *goal* (clause with only negative literals) $\leftarrow \text{Pred}(X, Y) \wedge \text{Pred}(Y, X)$. The representation of other constraints (e.g., irreflexivity, anti-transitivity, anti-triangularity, functionality, exclusivity) can be found in Cropper & Hocquette (2023). Current neuro-symbolic methods do not cover this aspect.

Recursion There are two types of recursion in Horn programs: recursion and mutual recursion (Bancilhon & Ramakrishnan, 1986). *Recursion* refers to the phenomenon that the same predicate appears simultaneously in a rule's head and body. For instance, $\text{Even}(X) \leftarrow \text{Even}(Y) \wedge \text{Succ}_2(Y, X)$ is recursive, where Even asserts whether a natural number is even and $\text{Succ}_2(Y, X)$ asserts whether

$X = Y + 2$. Even is the *recursive predicate* in this case. Besides, this recursive rule is also called *linear* because the recursive predicate only appears once in the body. *Mutual recursion* refers to the phenomenon that two predicates mutually derive from each other. We say a predicate Pred_1 *derives* another predicate Pred_2 if there exists such a set of rules (variables omitted)

$$\mathrm{Pred}_2 \leftarrow \cdots \wedge \mathrm{Q}_1 \qquad \mathrm{Q}_1 \leftarrow \cdots \wedge \mathrm{Q}_2 \qquad \mathrm{Q}_2 \leftarrow \cdots \wedge \mathrm{Q}_3 \qquad \cdots \qquad \mathrm{Q}_n \leftarrow \cdots \wedge \mathrm{Pred}_1$$

where Qs denote other predicates and $\ldots$ denote any other body atoms. Therefore, two mutually recursive predicates can be mutually deduced via forward chaining. For instance,

$$\mathrm{Even}(X) \leftarrow \mathrm{Odd}(Y) \wedge \mathrm{Succ}(Y, X) \qquad \mathrm{Odd}(X) \leftarrow \mathrm{Even}(Y) \wedge \mathrm{Succ}(Y, X)$$

, where $\mathrm{Succ}(X, Y)$ asserts whether $Y = X + 1$, show that Even and Odd (asserting whether a natural number is odd) are mutually recursive. A rule is also called *recursive* if the head predicate is mutually recursive with one of its body predicates, and the rule is called *linear* if only one mutually recursive predicate appears in the body. Enabling recursion is expensive for symbolic ILP, while the state-of-the-art neuro-symbolic ILP does not support mutual recursion (Glanois et al., 2022).

Predicate invention Predicate invention is a crucial part of automatically discovering new concepts, which may lead to breakthroughs in AI development (Russell, 2019, chap. 3). Specifically, predicate invention enables predicates that are unused in BK & target examples appearing in the hypothesis. For example, learning Even from the BK $\{\mathrm{Zero}(0), \mathrm{Succ}(0, 1), \mathrm{Succ}(1, 2), \mathrm{Succ}(2, 3), \ldots\}$ may require inventing dyadic relation Succ_2 and the following rules

$$\begin{aligned} &\mathrm{Even}(X) \leftarrow \mathrm{Even}(Y) \wedge \mathrm{Succ}_2(Y, X) \qquad \mathrm{Even}(0) \leftarrow \mathrm{Zero}(0) \\ &\mathrm{Succ}_2(Y, X) \leftarrow \mathrm{Succ}(Z, X) \wedge \mathrm{Succ}(Y, Z) \end{aligned}$$

While inventing such auxiliary predicates substantially reduces hypotheses (Cropper & Dumančić, 2022) and improves learning performance (Cropper, 2019), predicate invention is expensive, inaccurate, restricted to low-arity invention (Cropper & Morel, 2021b).

4.2 INSUFFICIENT NOISE HANDLING

Mislabeled & ambiguous data Noise is ubiquitous in realistic data. While symbolic methods struggle to learn from noisy data (Hocquette et al., 2024), neuro-symbolic methods can deal with mislabeled examples (Glanois et al., 2022) and ambiguous BK (Evans & Grefenstette, 2018). Handling mislabeled BK is still an open problem (Cropper & Dumančić, 2022). Further study on large-scale noisy tasks lacks synthesized datasets to control the noise rate in experiments.

Open-world assumption While the *closed-world assumption* (CWA) asserts any ground atom, whose predicate appears in the BK, to be false if it is not given in BK, the *open-world assumption* (OWA) allows those ground atoms that are not known to be true to have the possibility of being true (Reiter, 1981). Almost all existing ILP systems assume CWA. However, OWA is a more realistic setup since a complete BK is inaccessible in real-world applications. A set of incomplete background ground atoms is considered noisy if an ILP system assumes CWA.

4.3 MULTI-TASK LEARNING

Existing ILP focuses on once learning one target predicate. However, simultaneously learning several target predicates can share common rules and capture mutual recursions among target predicates. Glanois et al. (2022) proposes an iterative multi-task learning scheme for their neuro-symbolic model and successfully learns certain hypotheses at a small program size. Large-scale multi-task learning is a promising direction for building broader applications, which remains unexplored.

4.4 UNABLE TO LEARN FROM RAW INPUT

One major gap between ILP and modern ML systems is the ability to induce knowledge from raw sensory input. Most ILP systems only receive symbolic data as input, while raw data is usually images, speech, natural language, etc. There are initial works that use neural networks to perceive and transform raw input into symbolic form and use symbolic deduction to do reasoning (Manhaeve et al., 2018; Dai et al., 2019; Dai & Muggleton, 2021). Efficiently jointly training perception network and ILP system in large-scale tasks would be a promising direction for robust and reliable artificial intelligence. See Evans (2020); Evans et al. (2021; 2022) for more possibilities to induce rules from raw sensory input.

5 GEOILP

In this section, we formally introduce GeoILP, our proposed dataset for elaborating large-scale sophisticated rule induction systems, in four parts. First, a general guide for synthesizing ILP tasks from predefined rules. Second, the steps for identifying the examples (*deduction* step) and the BK (*traceback* step) of GeoILP. Third, the features of GeoILP critically differ from other ILP testbeds. Fourth, the approach to transform symbolic input into raw sensory input.

5.1 A GENERAL GUIDE FOR ILP TASK SYNTHESIS

Inductive reasoning can be seen as a "reverse" procedure of deductive reasoning in the sense that the former learns rules from premises and conclusions while the latter infers conclusions from premises and valid rules. In ILP, premises correspond to the BK, and the conclusions correspond to the target examples. Therefore, to construct ILP data, we can derive target examples from selected premises and predefined rules using a deduction engine.

Concretely, the data-synthesizing procedure works as follows:

1. Randomly or intentionally choose several ground atoms as premises.
2. Define a consistent set of rules. [2]
3. Deduce all the conclusions from the premises and rules using any deduction engine.
4. Select a part or all of the conclusions (also ground atoms) as **positive examples**. [3]
5. Trace back from the conclusions to identify a minimum set of premises contributing to deducing the conclusions as ILP **BK**.
6. Optionally, obtain **negative examples** by removing the conclusions from all syntactically possible ground atoms (all combinations of target predicate and constants).

5.2 SYNTHESIZING GEOILP

We choose *plane geometry* as the application domain as its formal system covers all the difficulties indicated in section 4 (see section 5.3).

Formalizing plane geometry and building a corresponding symbolic deduction engine are challenging works outside this paper's scope. We adopt an expert-designed deduction engine based on deductive database theory (Gallaire et al., 1984), similar to the ones used in automated geometry theorem proving (e.g., GEX (Chou et al., 2000), JGEX (Ye et al., 2010a;b; 2011), AlphaGeometry (Trinh et al., 2024)). [4] The only constants are the points in the plane. The engine leverages a set of Horn rules for deduction. Table 1 shows the characteristics of predicates.

Deduction To initialize *deduction* step with premises, we use 231 plane geometry problems given in JGEX (Ye et al., 2010a;b; 2011), ranging from textbook exercises, regional olympiads, and famous theorems. An example of premises is depicted in Figure 1. The final dataset, built from such selected premises, can effectively help construct automated geometry theorem provers without needing expert-defined rules, as the rules learned by ILP can be useful. Then, the deduction engine

[2] A set of rules is *consistent* if the rules do not contradict each other.

[3] Merely selecting a part of the conclusions as positive examples may yield fewer rules with more body atoms. $\mathrm{P}_1 \leftarrow \mathrm{P}_2 \wedge \mathrm{Q}$ and $\mathrm{P}_2 \leftarrow \mathrm{R}_1 \wedge \mathrm{R}_2$ may be reduced to $\mathrm{P}_1 \leftarrow \mathrm{R}_1 \wedge \mathrm{R}_2 \wedge \mathrm{Q}$ if the premises involve $\mathrm{Q}, \mathrm{R}_1, \mathrm{R}_2$ and only select P_1 as conclusions.

[4] Note that GEX, JGEX, and AlphaGeometry are deductive reasoning algorithms and are not applicable to ILP, which is inductive reasoning.

Table 1: Predicates used in GeoILP (Target=can be used as target predicate; Head=can be used in the head; Body=can be used in the body). Details see Appendix A.1.

Name	Arity	Target	Argument Symmetry	Constraints	Head	Body	Full Name
Coll	3	✗	✓	✓	✓	✓	collinear
Ncoll	3	✗	✓	✓	✓	✓	not collinear
Sameside	6	✗	✓	✓	✗	✓	same side
Midp	3	✓	✓	✓	✓	✓	midpoint
Cong	4	✓	✓	✓	✓	✓	segment congruent
Perp	4	✓	✓	✓	✓	✓	perpendicular
Para	4	✓	✓	✓	✓	✓	parallel
Cyclic	4	✓	✓	✓	✓	✓	concyclic
Circle	4	✓	✓	✓	✓	✓	circle
Eqangle	8	✓	✓	✓	✓	✓	equal angle
Eqratio	8	✓	✓	✓	✓	✓	equal ratio

uses forward chaining to reach the fix point. In the single-task setting, we separate conclusions with different predicates.

Note that the deduction engine regards argument-permutation equivalent atoms as the same, which substantially reduces deduction costs since argument symmetry is omnipresent. Therefore, rules for argument permutation and several trivial rules are not explicitly listed. See Appendix A.4 for details.

Traceback Since the conclusions deduced from one set of premises may involve all 8 predicates, synthesizing a single ILP task should filter out those premises irrelevant to the target predicate. To achieve this, the deduction engine constructs a deduction graph when doing forward chaining, which illustrates the immediate dependence of the ground atoms in the graph. Every body atom in a matched (ground) rule has a directed edge pointing to the head atom (see Figure 1). Starting from the conclusions involving only the target predicate, we trace back along the directed edges in the reverse direction until reaching the premises. The trace-backed premises are regarded as the BK. The directed edges alongside (red arrows in Figure 1) constitute a reference hypothesis.

After deduction and traceback, we repeat the BK and target examples ten times, retaining predicates unchanged but mapping every point (constants) to new, unique points. In other words, the initial group of points is duplicated into ten groups. Then, the data are divided into training set and evaluation set according to 8:2 point groups.

5.3 Dataset features

In total, 65 expert-defined rules and much more trivial rules encoded in the deductive database are used for synthesizing GeoILP (see Appendix A.4). Several rules are listed here to demonstrate how GeoILP covers the limitations mentioned in section 4.

$$\langle 1\rangle \qquad \mathrm{Midp}(M,C,D) \leftarrow \mathrm{Midp}(M,A,B) \wedge \mathrm{Para}(A,C,B,D) \wedge \mathrm{Para}(A,D,B,C)$$

$$\langle 2\rangle \qquad \mathrm{Para}(A,B,E,F) \leftarrow \mathrm{Para}(A,B,C,D) \wedge \mathrm{Para}(C,D,E,F)$$

$$\langle 3\rangle \qquad \mathrm{Cong}(A,M,B,M) \leftarrow \mathrm{Perp}(A,B,B,C) \wedge \mathrm{Midp}(M,A,C)$$

$$\langle 4\rangle \qquad \mathrm{Cong}(O,A,O,B) \leftarrow \mathrm{Midp}(M,A,B) \wedge \mathrm{Perp}(O,M,A,B)$$

$$\langle 5\rangle \qquad \mathrm{Perp}(A,B,P,Q) \leftarrow \mathrm{Cong}(A,P,B,P) \wedge \mathrm{Cong}(A,Q,B,Q)$$

We briefly introduce the meaning of the predicates appearing in the above rules and Figure 1 to facilitate understanding them. See Appendix A.1 for exhaustive descriptions. $\mathrm{Midp(M,A,B)}$ asserts M is the midpoint of segment AB. $\mathrm{Para(A,B,C,D)}$ asserts lines AB & CD are parallel. $\mathrm{Perp(A,B,C,D)}$ asserts lines AB & CD are perpendicular. $\mathrm{Cong(A,B,C,D)}$ asserts segments AB & CD are of same length. $\mathrm{Coll(A,B,C)}$ asserts A,B,C are collinear. $\mathrm{Ncoll(A,B,C)}$ asserts A,B,C are not collinear. $\mathrm{Cyclic(A,B,C,D)}$ asserts A,B,C,D are con-

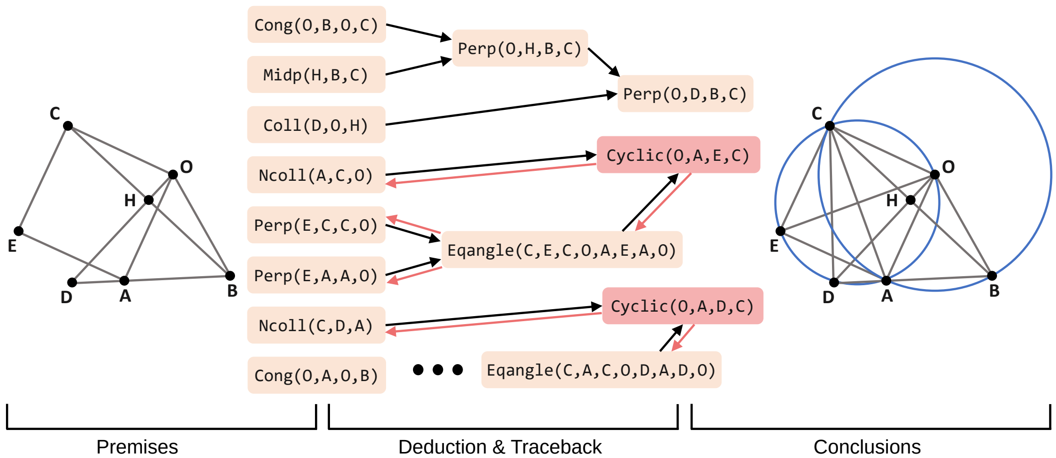

Figure 1: Synthesizing one of GeoILP (single) tasks from one set of premises with Cyclic as target predicate. Black arrows denote deduction (forward chaining), and red arrows denote traceback.

cyclic. $\mathrm{Eqangle(A, B, C, D, E, F, G, H)}$ asserts full-angles $[AB, CD]$ & $[EF, GH]$ are equal. Full-angle is defined by two lines and, intuitively, two full-angles $[AB, CD]$ & $[EF, GH]$ are equal if, supposing Rot denotes a rotation, $Rot(AB) \parallel EF$ and $Rot(EF) \parallel GH$. Refer to Ye et al. (2010b) for the formal definition.

Predicate Characteristics of predicates are provided in Table 1. The predicates are of arity from 3 to 8, all involving argument symmetry (see Appendix A.2) and different constraints (see Appendix A.3). Note that, among constraints, Midp and Circle are functional; thus, our dataset can be easily adapted to study the setting with functions. In addition, nearly all the predicates can be used in the head and body of a rule.

Rules The number of rules in a task ranges from 10 to 91, which do not consider rules for argument symmetry and trivial rules. See Appendix A.4 for details about trivial rules. The maximum number of body atoms is 5. the maximum number of variables in a rule is 12. Existentially quantified variables usually exist. For instance, A, B of Rule$\langle 1 \rangle$, C, D of Rule$\langle 2 \rangle$, C of Rule$\langle 3 \rangle$, M of Rule$\langle 4 \rangle$.

Recursion All types of recursion are omnipresent in the whole rule set, and almost any two predicates are mutually recursive. In the example rule subset above, Rule$\langle 1 \rangle$ is recursive and linear; Rule$\langle 2 \rangle$ is recursive but not linear; Rule$\langle 3 \rangle$ & Rule$\langle 5 \rangle$ (or Rule$\langle 4 \rangle$ & Rule$\langle 5 \rangle$) justify that they are recursive because Cong and Perp are mutually recursive; while Rule$\langle 3 \rangle$ (also Rule$\langle 4 \rangle$) is linear, Rule$\langle 5 \rangle$ is not linear.

Predicate invention Almost all GeoILP tasks require predicate invention. For example, a task learning Para with Cong, Coll, Ncoll, $\mathrm{Eqangle}$, Perp may need to invent Midp as auxiliary predicate (Appendix B.1). In addition, extra meaningful relations that are not used in GeoILP may also be invented to reduce hypothesis space, e.g., congruent triangle, similar triangle (Appendix B.1).

Noisy data First, our synthetic data makes the open-world assumption. For instance, in Figure 1, Perp appears twice in the BK (premises), while two new atoms of Perp not given in the BK also appear in the deduced conclusions, which means that atoms not given in the BK can also be true. In GeoILP, it is common for all true atoms not to be given in BK. Second, the negative examples are noisy since our rule set is incomplete for the entire *plane geometry* (i.e., several true atoms may not be deduced based on the incomplete rule set).

Multi-task In the multi-task setting, we trace back premises from all conclusions; thus, all predicates in the deduction graph are considered target predicates for an ILP multi-task. In most cases, the trace-backed premises are the same as the initial premises.

Table 2: Detailed comparison between GeoILP and the existing dataset. (# denotes *number of*)

Dataset	**∂ILP**	**GeoILP**
#Tasks	20 single-tasks	836 single-tasks; 207 multi-tasks
Predicate Arity	1 - 2	3 - 8
Predicate Invention	$\sim$ 60%	96% for single-tasks
Max #Variables	6	12
Max #Body Atoms	2	9
#Rules	1 - 5	10-100 for 85% single-tasks and 50% multi-tasks; up to 2000 for single-tasks; up to 3500 for multi-tasks
Argument Symmetry	in just a few predicates; limited to dyadic predicates	in every predicate; various symmetries in up to ogdoadic predicates
Predicate Constraints	in a few predicates	in every predicate
Recursion	linear recursion: 30% non-linear recursion: 10% linear mutual recursion: 10%	recursion: $\sim$ 100% mutual recursion: $\sim$ 90% all 4 kinds of recursion are common

5.3.1 Comparison with other ILP datasets

We compare GeoILP with the dataset proposed in ∂ILP (Evans & Grefenstette, 2018), which is the only synthetic dataset used by recent neuro-symbolic methods (e.g., HRI [20]). Table 2 reveals GeoILP's extremely strong complexity from various aspects. [5]

5.4 Constructing raw input

An essential difference distinguishing it from other datasets is that GeoILP additionally provides raw inputs corresponding to each task. Like in Figure 1, the BK (premises) is transformed into an image like in the plane geometry textbook. We adopt the constructive diagram builder language developed in AlphaGeometry (Trinh et al., 2024) to construct the image point by point from a given set of premises, which works well with the symbolic deduction engine. The goal is to provide data for learning rules from raw sub-symbolic inputs (images) and symbolic target examples. The images only contain basic geometry objects, reducing the burden of perception and making them a good testbed for this immature research topic. We also attach a corresponding image of conclusions (BK + positive examples) to each task, like the rightmost diagram in Figure 1. Handling GeoILP in geometric form at least requires the ability to identify geometric objects, to identify the relations among objects, and to induce interpretable rules. Developing such a complex system requires great effort, which is out of the scope of a dataset constructing work and is left to future work. [6]

More details about datasets and example data are provided in Appendix A & B, respectively.

6 Experiments

6.1 Setup

Considering the great difficulty of GeoILP, we divide it into four progressive levels, which provides chances for gradually strengthening ILP systems. Table 3 illustrates the specification of each level.

[5]Multi-tasks do not involve predicate invention since they regard all available predicates as target predicates.

[6]For readers interested in the induction ability of large language models, we provide a guide on how to translate GeoILP into natural-language form in Appendix C.

Table 3: GeoILP's specification of four progressive levels for single-tasks & specification of multi-tasks.

Level	basic	simple	advanced	complex	multi-task
(single) tasks%	2%	7%	23%	66%	-
arity	≤ 8	≤ 8	≤ 8	≤ 8	≤ 8
# body atoms	≤ 2	≤ 9	≤ 9	≤ 9	≤ 9
mutual recursion?	✗	✗	✓	✓	✓
# rules	≤ 20	≤ 20	≤ 20	≤ 2014	≤ 3553

The four levels are set according to four dimensionalities: predicate arity, number of body atoms, involving mutual recursion or not, and number of rules. The former three are critical bottlenecks of many neuro-symbolic ILP methods, e.g., Glanois et al. (2022). The last one is a critical aspect affecting search complexity for both symbolic and neuro-symbolic ILP methods.

6.2 Results

Symbolic Among symbolic methods, Popper (Cropper & Morel, 2021a) is the most powerful one that simultaneously supports learning recursion, involving hypothesis constraints, inventing predicates (Cropper & Morel, 2021b), and handling noise (Hocquette et al., 2024), and scales better as well. We conduct experiments using Popper [7], enabling predicate invention, recursion and noise handling. Noise handling is turned on because GeoILP follows OWA, while Popper follows CWA. The maximum number of variables in a rule is set to 12, which is the maximum value in every four levels. When conducting experiments on different levels, we set the maximum number of body atoms and the maximum number of rules to the maximum values of the learning level. This setup aligns with our purpose of not injecting many priors into training. After 1-day searching, Popper does not return any hypothesis, even at the *basic* level. We regard GeoILP as unsolvable by Popper since the searching time is already about two orders of magnitude longer than in previous works (about hundreds of seconds or less, e.g., Cropper & Morel (2021a); Glanois et al. (2022))

Neuro-symbolic Several neuro-symbolic methods (Evans & Grefenstette, 2018; Glanois et al., 2022) do not support higher-arity predicates, rules with more than two body atoms, or using target predicate in the rule's body, which are their primary bottlenecks of being inapplicable to GeoILP. Several others (Rocktäschel & Riedel, 2017; Campero et al., 2018) require expert-defined rule templates task-by-task, which is inappropriate for large-scale applications like GeoILP. Difflog (Si et al., 2019) is a neural-symbolic method that supports arbitrary hypothesis space. However, our experiments show that, even at the *basic* level, Difflog [8] throws an out-of-memory error on a server with 500GB of memory, an order of magnitude larger than in the original paper (64GB). We leave further investigation on improving memory usage for future work.

7 Discussion and Conclusion

We propose GeoILP, a large-scale synthetic dataset for inductive logic programming involving all challenging language biases in reference hypotheses. GeoILP is, in terms of the hypothesis size, at least one magnitude larger than existing datasets that can provide guiding hypotheses. Although GeoILP may be biased towards plane geometry, it is still a good testbed for large-scale ILP. Besides, we also provide image-form background knowledge, aiming to boost the development of joint learning of neural perception and symbolic rule induction.

[7] Version 4.3.0: `https://github.com/logic-and-learning-lab/Popper/tree/v4.3.0`

[8] We leverage the implementation and recommend parameter setting in `https://github.com/petablox/difflog/tree/3c2d5218d9a0a1e200ebbf2d6a1e5d077fb18826`.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Francois Bancilhon and Raghu Ramakrishnan. An amateur's introduction to recursive query processing strategies. In *Proceedings of the 1986 ACM SIGMOD international conference on Management of data*, pp. 16–52, 1986.

Sándor Bartha and James Cheney. Towards meta-interpretive learning of programming language semantics. In *Inductive Logic Programming: 29th International Conference, ILP 2019, Plovdiv, Bulgaria, September 3–5, 2019, Proceedings 29*, pp. 16–25. Springer, 2020.

Hendrik Blockeel and Luc De Raedt. Lookahead and discretization in ilp. In *International Conference on Inductive Logic Programming*, pp. 77–84. Springer, 1997.

David A Bohan, Corinne Vacher, Alireza Tamaddoni-Nezhad, Alan Raybould, Alex J Dumbrell, and Guy Woodward. Next-generation global biomonitoring: large-scale, automated reconstruction of ecological networks. *Trends in ecology & evolution*, 32(7):477–487, 2017.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

Andres Campero, Aldo Pareja, Tim Klinger, Josh Tenenbaum, and Sebastian Riedel. Logical rule induction and theory learning using neural theorem proving. *arXiv preprint arXiv:1809.02193*, 2018.

Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. A deductive database approach to automated geometry theorem proving and discovering. *Journal of Automated Reasoning*, 25(3): 219–246, 2000.

Andrew Cropper. Playgol: Learning programs through play. *arXiv preprint arXiv:1904.08993*, 2019.

Andrew Cropper and Sebastijan Dumančić. Inductive logic programming at 30: a new introduction. *Journal of Artificial Intelligence Research*, 74:765–850, 2022.

Andrew Cropper and Céline Hocquette. Learning logic programs by discovering where not to search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6289–6296, 2023.

Andrew Cropper and Rolf Morel. Learning programs by learning from failures. *Machine Learning*, 110(4):801–856, 2021a.

Andrew Cropper and Rolf Morel. Predicate invention by learning from failures. *arXiv preprint arXiv:2104.14426*, 2021b.

Andrew Cropper, Sebastijan Dumančić, and Stephen H. Muggleton. Turning 30: New ideas in inductive logic programming. In Christian Bessiere (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 4833–4839. International Joint Conferences on Artificial Intelligence Organization, 7 2020a. doi: 10.24963/ijcai.2020/673. URL `https://doi.org/10.24963/ijcai.2020/673`. Survey track.

Andrew Cropper, Richard Evans, and Mark Law. Inductive general game playing. *Machine Learning*, 109:1393–1434, 2020b.

Wang-Zhou Dai and Stephen Muggleton. Abductive knowledge induction from raw data. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 1845–1851. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/254. URL `https://doi.org/10.24963/ijcai.2021/254`. Main Track.

Wang-Zhou Dai, Qiuling Xu, Yang Yu, and Zhi-Hua Zhou. Bridging machine learning and logical reasoning by abductive learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/9c19a2aa1d84e04b0bd4bc888792bd1e-Paper.pdf.

Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural logic machines. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=B1xY-hRctX.

Richard Evans. *Kant's cognitive architecture*. PhD thesis, Imperial College London, 2020.

Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

Richard Evans, José Hernández-Orallo, Johannes Welbl, Pushmeet Kohli, and Marek Sergot. Making sense of sensory input. *Artificial Intelligence*, 293:103438, 2021. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2020.103438. URL https://www.sciencedirect.com/science/article/pii/S0004370220301855.

Richard Evans, Matko Bošnjak, Lars Buesing, Kevin Ellis, David Pfau, Pushmeet Kohli, and Marek Sergot. Making sense of raw input (extended abstract). In Lud De Raedt (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 5727–5731. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/799. URL https://doi.org/10.24963/ijcai.2022/799. Journal Track.

Herve Gallaire, Jack Minker, and Jean-Marie Nicolas. Logic and databases: A deductive approach. *ACM Computing Surveys (CSUR)*, 16(2):153–185, 1984.

Claire Glanois, Zhaohui Jiang, Xuening Feng, Paul Weng, Matthieu Zimmer, Dong Li, Wulong Liu, and Jianye Hao. Neuro-symbolic hierarchical rule induction. In *International Conference on Machine Learning*, pp. 7583–7615. PMLR, 2022.

Céline Hocquette, Andreas Niskanen, Matti Järvisalo, and Andrew Cropper. Learning mdl logic programs from noisy data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10553–10561, 2024.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6700–6709, 2019.

Katsumi Inoue, Andrei Doncescu, and Hidetomo Nabeshima. Completing causal networks by meta-level abduction. *Machine learning*, 91(2):239–277, 2013.

Samina Khalid, Tehmina Khalil, and Shamila Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In *2014 science and information conference*, pp. 372–378. IEEE, 2014.

Mark Law, Alessandra Russo, and Krysia Broda. Inductive learning of answer set programs from noisy examples. *arXiv preprint arXiv:1808.08441*, 2018.

Mark Law, Alessandra Russo, Elisa Bertino, Krysia Broda, and Jorge Lobo. Representing and learning grammars in answer set programming. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):2919–2928, Jul. 2019. doi: 10.1609/aaai.v33i01.33012919. URL https://ojs.aaai.org/index.php/AAAI/article/view/4147.

Mark Law, Alessandra Russo, and Krysia Broda. The ilasp system for inductive learning of answer set programs. *arXiv preprint arXiv:2005.00904*, 2020.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/dc5d637ed5e62c36ecb73b654b05ba2a-Paper.pdf.

Stephen Muggleton. Inverse entailment and progol. *New generation computing*, 13:245–286, 1995.

Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

Stephen Muggleton, Luc De Raedt, David Poole, Ivan Bratko, Peter Flach, Katsumi Inoue, and Ashwin Srinivasan. Ilp turns 20: biography and future challenges. *Machine learning*, 86:3–23, 2012.

Stephen H Muggleton and Christopher H Bryant. Theory completion using inverse entailment. In *International conference on inductive logic programming*, pp. 130–146. Springer, 2000.

Stephen H Muggleton, Dianhuan Lin, Niels Pahlavi, and Alireza Tamaddoni-Nezhad. Meta-interpretive learning: application to grammatical inference. *Machine learning*, 94:25–49, 2014.

Stephen H Muggleton, Dianhuan Lin, and Alireza Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. *Machine Learning*, 100(1):49–73, 2015.

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

Raymond Reiter. On closed world data bases. In Bonnie Lynn Webber and Nils J. Nilsson (eds.), *Readings in Artificial Intelligence*, pp. 119–140. Morgan Kaufmann, 1981. ISBN 978-0-934613-03-3. doi: https://doi.org/10.1016/B978-0-934613-03-3.50014-3. URL https://www.sciencedirect.com/science/article/pii/B9780934613033500143.

Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/b2ab001909a8a6f04b51920306046ce5-Paper.pdf.

Stuart Russell. *Human compatible: AI and the problem of control*. Penguin Uk, 2019.

Xujie Si, Mukund Raghothaman, Kihong Heo, and Mayur Naik. Synthesizing datalog programs using numerical relaxation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 6117–6124. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/847. URL https://doi.org/10.24963/ijcai.2019/847.

Aishwarya Sivaraman, Tianyi Zhang, Guy Van den Broeck, and Miryung Kim. Active inductive logic programming for code search. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pp. 292–303. IEEE, 2019.

A SRINIVASAN. The aleph manual. *http://www.cs.ox.ac.uk/activities/programinduction/Aleph/*, 2001.

Alireza Tamaddoni-Nezhad, Raphael Chaleil, Antonis Kakas, and Stephen Muggleton. Application of abductive ilp to learning metabolic network inhibition from temporal data. *Machine Learning*, 64:209–230, 2006.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In Alexandre Allauzen, Edward Grefenstette, Karl Moritz Hermann, Hugo Larochelle, and Scott Wen-tau Yih (eds.), *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-4007. URL https://aclanthology.org/W15-4007.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. *Advances in neural information processing systems*, 30, 2017.

Zheng Ye, Shang-Ching Chou, and Xiao-Shan Gao. Visually dynamic presentation of proofs in plane geometry: Part 1. basic features and the manual input method. *Journal of Automated Reasoning*, 45:213–241, 2010a.

Zheng Ye, Shang-Ching Chou, and Xiao-Shan Gao. Visually dynamic presentation of proofs in plane geometry: Part 2. automated generation of visually dynamic presentations with the full-angle method and the deductive database method. *Journal of Automated Reasoning*, 45:243–266, 2010b.

Zheng Ye, Shang-Ching Chou, and Xiao-Shan Gao. An introduction to java geometry expert. In *Automated Deduction in Geometry: 7th International Workshop, ADG 2008, Shanghai, China, September 22-24, 2008. Revised Papers 7*, pp. 189–195. Springer, 2011.

Zheng Zhang, Levent Yilmaz, and Bo Liu. A critical review of inductive logic programming techniques for explainable ai. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

A DETAILS OF GEOILP

A.1 PREDICATES DESCRIPTION

In this section, we exhaustively describe the predicates.

The constants in GeoILP are points in the plane. The exact meaning of each predicate is listed below.

- $\mathrm{Midp(M, A, B)}$ asserts M is the midpoint of segment AB.
- $\mathrm{Para(A, B, C, D)}$ asserts line AB and line CD are parallel.
- $\mathrm{Perp(A, B, C, D)}$ asserts line AB and line CD are perpendicular.
- $\mathrm{Cong(A, B, C, D)}$ asserts segment AB and segment CD are of same length.
- $\mathrm{Coll(A, B, C)}$ asserts A, B, C are collinear.
- $\mathrm{Ncoll(A, B, C)}$ asserts A, B, C are not collinear.
- $\mathrm{Cyclic(A, B, C, D)}$ asserts A, B, C, D are concyclic.
- $\mathrm{Eqangle(A, B, C, D, E, F, G, H)}$ asserts full-angles $[AB, CD]$ & $[EF, GH]$ are equal.
  - Full-angle is defined by two lines. Intuitively, two full-angles $[AB, CD]$ & $[EF, GH]$ are equal if, supposing Rot denotes a rotation, $Rot(AB) \parallel EF$ and $Rot(CD) \parallel GH$.
  - Refer to Ye et al. (2010b) for the formal definition.
- $\mathrm{Eqratio(A, B, C, D, E, F, G, H)}$ asserts the ratio of segment AB and segment CD equals to the ratio of segment EF and GH.
- $\mathrm{Circle(O, A, B, C)}$ asserts A, B, C are concyclic and the center is O.
- $\mathrm{Sameside}(B, A, C, Y, X, Z)$ asserts B is to the same side of $A\&C$ as Y is to $X\&Z$. Mathematically, this is equivalent to $(\vec{BA} \cdot \vec{BC})(\vec{YX} \cdot \vec{YZ}) > 0$.

$\mathrm{Sameside}$ are not allowed as target predicates because it does not appear in the head of any rules. Coll and Ncoll are not allowed as target predicates because the rules where they appear in the head are either too few or trivial (see Appendix A.4 for trivial rules).

A.2 ARGUMENT SYMMETRY OF PREDICATES

We use biconditional to describe the argument symmetry of predicates.

- $\mathrm{Midp(M, A, B)} \leftrightarrow \mathrm{Midp}(M, B, A)$
- $\mathrm{Para(A, B, C, D)} \leftrightarrow \mathrm{Para}(C, D, A, B)$
- $\mathrm{Para(A, B, C, D)} \leftrightarrow \mathrm{Para}(B, A, C, D)$
- $\mathrm{Perp(A, B, C, D)} \leftrightarrow \mathrm{Perp}(C, D, A, B)$
- $\mathrm{Perp(A, B, C, D)} \leftrightarrow \mathrm{Perp}(B, A, C, D)$
- $\mathrm{Cong(A, B, C, D)} \leftrightarrow \mathrm{Cong}(C, D, A, B)$
- $\mathrm{Cong(A, B, C, D)} \leftrightarrow \mathrm{Cong}(B, A, C, D)$
- $\mathrm{Coll(A, B, C)} \leftrightarrow \mathrm{Coll}(A, C, B)$
- $\mathrm{Coll(A, B, C)} \leftrightarrow \mathrm{Coll}(B, C, A)$
- $\mathrm{Ncoll(A, B, C)} \leftrightarrow \mathrm{Ncoll}(A, C, B)$
- $\mathrm{Ncoll(A, B, C)} \leftrightarrow \mathrm{Ncoll}(B, C, A)$
- $\mathrm{Cyclic(A, B, C, D)} \leftrightarrow \mathrm{Cyclic}(B, C, D, A)$
- $\mathrm{Cyclic(A, B, C, D)} \leftrightarrow \mathrm{Cyclic}(B, A, C, D)$
- $\mathrm{Circle}(O, A, B, C) \leftrightarrow \mathrm{Circle}(O, B, C, A)$
- $\mathrm{Circle}(O, A, B, C) \leftrightarrow \mathrm{Circle}(O, A, C, B)$
- $\mathrm{Sameside}(B, A, C, Y, X, Z) \leftrightarrow \mathrm{Sameside}(Y, X, Z, B, A, C)$

- Sameside$(B, A, C, Y, X, Z) \leftrightarrow$ Sameside(B, C, A, Y, X, Z)
- Eqangle$(A, B, C, D, E, F, G, H) \leftrightarrow$ Eqangle(E, F, G, H, A, B, C, D)
- Eqangle$(A, B, C, D, E, F, G, H) \leftrightarrow$ Eqangle(C, D, A, B, G, H, E, F)
- Eqangle$(A, B, C, D, E, F, G, H) \leftrightarrow$ Eqangle(B, A, C, D, E, F, G, H)
- Eqratio$(A, B, C, D, E, F, G, H) \leftrightarrow$ Eqratio(E, F, G, H, A, B, C, D)
- Eqratio$(A, B, C, D, E, F, G, H) \leftrightarrow$ Eqratio(C, D, A, B, G, H, E, F)
- Eqratio$(A, B, C, D, E, F, G, H) \leftrightarrow$ Eqratio(B, A, C, D, E, F, G, H)

Note that these biconditional formulae constitute a minimal representation of argument symmetry. Much more symmetries can be derived by combining several of them.

The rule set would be too large if these argument symmetries were all represented as Horn rules. A compact representation and effective handling should be thus considered by ILP methods.

A.3 Constraints of predicates

We describe the constraints of predicates. We express the constraints by Horn goal, as in Cropper & Hocquette (2023).

- $\leftarrow$ Sameside$(B, A, C, Y, X, Z) \wedge$ Sameside(A, B, C, Y, X, Z)
- $\leftarrow$ Midp$(M, A, B) \wedge$ Midp$(N, A, B) \wedge M \neq N$
- $\leftarrow$ Circle$(O, A, B, C) \wedge$ Circle$(Q, A, B, C) \wedge O \neq Q$
- $\leftarrow$ Perp$(A, B, C, D) \wedge$ Perp$(C, D, E, F) \wedge$ Perp(A, B, E, F)
- $\leftarrow$ Eqangle$(A, B, C, D, E, F, G, H) \wedge$ Eqangle$(C, D, A, B, E, F, G, H) \wedge$ $Npara(A, B, C, D)$
- $\leftarrow$ Eqangle$(A, B, C, D, E, F, G, H) \wedge$ Eqangle$(C, D, A, B, E, F, G, H) \wedge$ $Ncong(A, B, C, D)$

(Npara for not parallel; Ncong for not congruent)

A.4 Rules

All the Horn rules utilized to synthesize GeoILP are divided into expert-defined rules, trivial rules, and argument-symmetry rules (if representing argument symmetries by Horn rules).

Trivial rules are Horn rules trivial for human. For example, Coll$(A, B, C) \leftarrow$ Coll$(A, D, B) \wedge$ Coll(A, D, C). Note that, though trivial for human, an ILP learner may suffer from the large number of trivial rules. A whole set of such trivial rules is illustrated in Chou et al. (2000) and is encoded into the deductive database.

The 65 expert-defined rules are

1. Para$(A, B, E, F) \leftarrow$ Perp$(A, B, C, D) \wedge$ Perp$(C, D, E, F) \wedge$ Ncoll(A, B, E)
2. Cyclic$(A, B, C, D) \leftarrow$ Cong$(O, A, O, B) \wedge$ Cong$(O, B, O, C) \wedge$ Cong(O, C, O, D)
3. Para$(A, B, C, D) \leftarrow$ Eqangle(A, B, P, Q, C, D, P, Q)
4. Eqangle$(P, A, P, B, Q, A, Q, B) \leftarrow$ Cyclic(A, B, P, Q)
5. Cyclic$(A, B, P, Q) \leftarrow$ Eqangle$(P, A, P, B, Q, A, Q, B) \wedge$ Ncoll(P, Q, A)
6. Cong$(A, B, P, Q) \leftarrow$ Cyclic$(A, B, C, P) \wedge$ Cyclic$(A, B, C, Q) \wedge$ Cyclic$(A, B, C, R) \wedge$ Eqangle(C, A, C, B, R, P, R, Q)
7. Para$(E, F, B, C) \leftarrow$ Midp$(E, A, B) \wedge$ Midp(F, A, C)
8. Eqratio3$(A, B, C, D, O, O) \leftarrow$ Para$(A, B, C, D) \wedge$ Coll$(O, A, C) \wedge$ Coll(O, B, D)
9. Eqangle$(A, B, E, F, C, D, G, H) \leftarrow$ Perp$(A, B, C, D) \wedge$ Perp(E, F, G, H)
10. Eqangle$(A, B, E, F, M, N, R, U) \leftarrow$ Eqangle$(A, B, C, D, M, N, P, Q) \wedge$ Eqangle(C, D, E, F, P, Q, R, U)

11. Eqratio$(A, B, E, F, M, N, R, U) \leftarrow$ Eqratio$(A, B, C, D, M, N, P, Q) \wedge$ Eqratio(C, D, E, F, P, Q, R, U)
12. Eqangle$(A, B, A, D, A, D, A, C) \leftarrow$ Eqratio$(D, B, D, C, A, B, A, C) \wedge$Coll$(D, B, C) \wedge$ Ncoll(A, B, C)
13. Eqratio$(D, B, D, C, A, B, A, C) \leftarrow$ Eqangle$(A, B, A, D, A, D, A, C) \wedge$Coll$(D, B, C) \wedge$ Ncoll(A, B, C)
14. Eqangle$(O, A, A, B, A, B, O, B) \leftarrow$ Cong$(O, A, O, B) \wedge$ Ncoll(O, A, B)
15. Cong$(O, A, O, B) \leftarrow$ Eqangle$(A, O, A, B, B, A, B, O) \wedge$ Ncoll(O, A, B)
16. Eqangle$(A, X, A, B, C, A, C, B) \leftarrow$ Circle$(O, A, B, C) \wedge$ Perp(O, A, A, X)
17. Perp$(O, A, A, X) \leftarrow$ Circle$(O, A, B, C) \wedge$ Eqangle(A, X, A, B, C, A, C, B)
18. Eqangle$(A, B, A, C, O, B, O, M) \leftarrow$ Circle$(O, A, B, C) \wedge$ Midp(M, B, C)
19. Midp$(M, B, C) \leftarrow$ Circle$(O, A, B, C) \wedge$ Coll$(M, B, C) \wedge$ Eqangle(A, B, A, C, O, B, O, M)
20. Cong$(A, M, B, M) \leftarrow$ Perp$(A, B, B, C) \wedge$ Midp(M, A, C)
21. Perp$(A, B, B, C) \leftarrow$ Circle$(O, A, B, C) \wedge$ Coll(O, A, C)
22. Eqangle$(A, D, C, D, C, D, C, B) \leftarrow$ Cyclic$(A, B, C, D) \wedge$ Para(A, B, C, D)
23. Cong$(O, A, O, B) \leftarrow$ Midp$(M, A, B) \wedge$ Perp(O, M, A, B)
24. Perp$(A, B, P, Q) \leftarrow$ Cong$(A, P, B, P) \wedge$ Cong(A, Q, B, Q)
25. Perp$(P, A, A, Q) \leftarrow$ Cong$(A, P, B, P) \wedge$ Cong$(A, Q, B, Q) \wedge$ Cyclic(A, B, P, Q)
26. Para$(A, C, B, D) \leftarrow$ Midp$(M, A, B) \wedge$ Midp(M, C, D)
27. Midp$(M, C, D) \leftarrow$ Midp$(M, A, B) \wedge$ Para$(A, C, B, D) \wedge$ Para(A, D, B, C)
28. Para$(A, B, C, D) \leftarrow$ Eqratio$(O, A, A, C, O, B, B, D) \wedge$Coll$(O, A, C) \wedge$Coll$(O, B, D) \wedge$ Ncoll$(A, B, C) \wedge$ Sameside(A, O, C, B, O, D)
29. Coll$(A, B, C) \leftarrow$ Para(A, B, A, C)
30. Eqratio$(M, A, A, B, N, C, C, D) \leftarrow$ Midp$(M, A, B) \wedge$ Midp(N, C, D)
31. Perp$(A, B, C, D) \leftarrow$ Eqangle$(A, B, P, Q, C, D, U, V) \wedge$ Perp(P, Q, U, V)
32. Cong$(A, B, C, D) \leftarrow$ Eqratio$(A, B, P, Q, C, D, U, V) \wedge$ Cong(P, Q, U, V)
33. Eqangle$(A, B, A, C, P, Q, P, R) \leftarrow$ Cong$(A, B, P, Q) \wedge$ Cong$(B, C, Q, R) \wedge$ Cong$(C, A, R, P) \wedge$ Ncoll(A, B, C)
34. Eqangle$(B, A, B, C, Q, P, Q, R) \leftarrow$ Cong$(A, B, P, Q) \wedge$ Cong$(B, C, Q, R) \wedge$ Cong$(C, A, R, P) \wedge$ Ncoll(A, B, C)
35. Eqangle$(C, A, C, B, R, P, R, Q) \leftarrow$ Cong$(A, B, P, Q) \wedge$ Cong$(B, C, Q, R) \wedge$ Cong$(C, A, R, P) \wedge$ Ncoll(A, B, C)
36. Cong$(A, C, P, R) \leftarrow$ Cong$(A, B, P, Q) \wedge$ Cong$(B, C, Q, R) \wedge$ Eqangle$(B, A, B, C, Q, P, Q, R) \wedge$ Ncoll(A, B, C)
37. Eqangle$(A, B, A, C, P, Q, P, R) \leftarrow$ Cong$(A, B, P, Q) \wedge$ Cong$(B, C, Q, R) \wedge$ Eqangle$(B, A, B, C, Q, P, Q, R) \wedge$ Ncoll(A, B, C)
38. Eqangle$(C, A, C, B, R, P, R, Q) \leftarrow$ Cong$(A, B, P, Q) \wedge$ Cong$(B, C, Q, R) \wedge$ Eqangle$(B, A, B, C, Q, P, Q, R) \wedge$ Ncoll(A, B, C)
39. Eqangle$(A, B, A, C, P, Q, P, R) \leftarrow$ Eqangle$(B, A, B, C, Q, P, Q, R) \wedge$ Eqangle$(C, A, C, B, R, P, R, Q) \wedge$ Ncoll(A, B, C)
40. Eqratio$(A, B, P, Q, B, C, Q, R) \leftarrow$ Eqangle$(B, A, B, C, Q, P, Q, R) \wedge$ Eqangle$(C, A, C, B, R, P, R, Q) \wedge$ Ncoll(A, B, C)
41. Eqratio$(B, C, Q, R, C, A, R, P) \leftarrow$ Eqangle$(B, A, B, C, Q, P, Q, R) \wedge$ Eqangle$(C, A, C, B, R, P, R, Q) \wedge$ Ncoll(A, B, C)
42. Eqratio$(C, A, R, P, A, B, P, Q) \leftarrow$ Eqangle$(B, A, B, C, Q, P, Q, R) \wedge$ Eqangle$(C, A, C, B, R, P, R, Q) \wedge$ Ncoll(A, B, C)

43. $\text{Cong}(B, C, Q, R) \leftarrow \text{Eqangle}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqangle}(C, A, C, B, R, P, R, Q) \wedge \text{Ncoll}(A, B, C) \wedge \text{Cong}(A, B, P, Q)$
44. $\text{Cong}(A, C, P, R) \leftarrow \text{Eqangle}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqangle}(C, A, C, B, R, P, R, Q) \wedge \text{Ncoll}(A, B, C) \wedge \text{Cong}(A, B, P, Q)$
45. $\text{Eqangle}(A, B, A, C, P, Q, P, R) \leftarrow \text{Eqangle}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqangle}(C, A, C, B, R, P, R, Q) \wedge \text{Ncoll}(A, B, C) \wedge \text{Cong}(A, B, P, Q)$
46. $\text{Cong}(A, C, P, R) \leftarrow \text{Eqangle}(B, A, B, C, Q, R, Q, P) \wedge \text{Eqangle}(C, A, C, B, R, Q, R, P) \wedge \text{Ncoll}(A, B, C) \wedge \text{Cong}(A, B, P, Q)$
47. $\text{Cong}(B, C, Q, R) \leftarrow \text{Eqangle}(B, A, B, C, Q, R, Q, P) \wedge \text{Eqangle}(C, A, C, B, R, Q, R, P) \wedge \text{Ncoll}(A, B, C) \wedge \text{Cong}(A, B, P, Q)$
48. $\text{Eqangle}(A, B, A, C, P, Q, P, R) \leftarrow \text{Eqangle}(B, A, B, C, Q, R, Q, P) \wedge \text{Eqangle}(C, A, C, B, R, Q, R, P) \wedge \text{Ncoll}(A, B, C) \wedge \text{Cong}(A, B, P, Q)$
49. $\text{Eqangle}(A, B, A, C, P, Q, P, R) \leftarrow \text{Eqratio}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqratio}(C, A, C, B, R, P, R, Q) \wedge \text{Ncoll}(A, B, C)$
50. $\text{Eqratio}(C, A, R, P, A, B, P, Q) \leftarrow \text{Eqratio}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqratio}(C, A, C, B, R, P, R, Q) \wedge \text{Ncoll}(A, B, C)$
51. $\text{Eqangle}(B, A, B, C, Q, P, Q, R) \leftarrow \text{Eqratio}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqratio}(C, A, C, B, R, P, R, Q) \wedge \text{Ncoll}(A, B, C)$
52. $\text{Eqangle}(C, A, C, B, R, P, R, Q) \leftarrow \text{Eqratio}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqratio}(C, A, C, B, R, P, R, Q) \wedge \text{Ncoll}(A, B, C)$
53. $\text{Eqangle}(A, B, A, C, P, Q, P, R) \leftarrow \text{Eqratio}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqangle}(B, A, B, C, Q, P, Q, R) \wedge \text{Ncoll}(A, B, C)$
54. $\text{Eqangle}(C, A, C, B, R, P, R, Q) \leftarrow \text{Eqratio}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqangle}(B, A, B, C, Q, P, Q, R) \wedge \text{Ncoll}(A, B, C)$
55. $\text{Eqratio}(B, C, C, A, Q, R, R, P) \leftarrow \text{Eqratio}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqangle}(B, A, B, C, Q, P, Q, R) \wedge \text{Ncoll}(A, B, C)$
56. $\text{Eqratio}(C, A, A, B, R, P, P, Q) \leftarrow \text{Eqratio}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqangle}(B, A, B, C, Q, P, Q, R) \wedge \text{Ncoll}(A, B, C)$
57. $\text{Cong}(B, C, Q, R) \leftarrow \text{Eqratio}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqratio}(C, A, C, B, R, P, R, Q) \wedge \text{Ncoll}(A, B, C) \wedge \text{Cong}(A, B, P, Q)$
58. $\text{Cong}(A, C, P, R) \leftarrow \text{Eqratio}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqratio}(C, A, C, B, R, P, R, Q) \wedge \text{Ncoll}(A, B, C) \wedge \text{Cong}(A, B, P, Q)$
59. $\text{Eqangle}(A, B, A, C, P, Q, P, R) \leftarrow \text{Eqratio}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqratio}(C, A, C, B, R, P, R, Q) \wedge \text{Ncoll}(A, B, C) \wedge \text{Cong}(A, B, P, Q)$
60. $\text{Eqangle}(B, A, B, C, Q, P, Q, R) \leftarrow \text{Eqratio}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqratio}(C, A, C, B, R, P, R, Q) \wedge \text{Ncoll}(A, B, C) \wedge \text{Cong}(A, B, P, Q)$
61. $\text{Eqangle}(C, A, C, B, R, Q, R, P) \leftarrow \text{Eqratio}(B, A, B, C, Q, P, Q, R) \wedge \text{Eqratio}(C, A, C, B, R, P, R, Q) \wedge \text{Ncoll}(A, B, C) \wedge \text{Cong}(A, B, P, Q)$
62. $\text{Para}(M, N, A, B) \leftarrow \text{Para}(A, B, C, D) \wedge \text{Coll}(M, A, D) \wedge \text{Coll}(N, B, C) \wedge \text{Eqratio}(M, A, M, D, N, B, N, C) \wedge \text{Sameside}(M, A, D, N, B, C)$
63. $\text{Eqratio}(M, A, M, D, N, B, N, C) \leftarrow \text{Para}(A, B, C, D) \wedge \text{Coll}(M, A, D) \wedge \text{Coll}(N, B, C) \wedge \text{Para}(M, N, A, B)$

These rules are adapted from those in AlphaGeometry. [9]

[9] `https://github.com/google-deepmind/alphageometry/blob/main/jgex_ag_231.txt`

B DATA EXAMPLES FROM GEOILP

B.1 PREDICATE INVENTION

The example below requires the invention of Eqangle

Background Knowledge:

- Cong(D8,A8,D8,B8)
- Cong(D8,B8,D8,C8)
- Cong(D8,E8,D8,A8)
- Coll(C8,A8,F8)
- Perp(E8,F8,A8,C8)
- Coll(G8,A8,B8)
- Perp(E8,G8,A8,B8)
- Ncoll(F8,G8,A8)
- Cong(D9,A9,D9,B9)
- Cong(D9,B9,D9,C9)
- Cong(D9,E9,D9,A9)
- Coll(C9,A9,F9)
- Perp(E9,F9,A9,C9)
- Coll(G9,A9,B9)
- Perp(E9,G9,A9,B9)
- Ncoll(F9,G9,A9)

Positive Examples:

- Cyclic(C8,E8,A8,B8)
- Cyclic(E8,A8,F8,G8)
- Cyclic(C9,E9,A9,B9)
- Cyclic(E9,A9,F9,G9)

Proof steps:

- Cyclic(C8,E8,A8,B8) $\leftarrow$ Cong(D8,A8,D8,B8) $\wedge$ Cong(D8,E8,D8,A8) $\wedge$ Cong(D8,B8,D8,C8)
- Coll(F8,A8,C8) $\leftarrow$ Coll(C8,A8,F8)
- Coll(G8,A8,B8) $\leftarrow$ Coll(G8,A8,B8)
- Eqangle(A8,C8,E8,F8,A8,B8,E8,G8) $\leftarrow$ Perp(E8,G8,A8,B8) $\wedge$ Perp(E8,F8,A8,C8)
- Eqangle(F8,A8,F8,E8,G8,A8,G8,E8) $\leftarrow$ Coll(F8,A8,C8) $\wedge$ Coll(G8,A8,B8) $\wedge$ Eqangle(A8,C8,E8,F8,A8,B8,E8,G8)
- Cyclic(E8,A8,F8,G8) $\leftarrow$ Eqangle(F8,A8,F8,E8,G8,A8,G8,E8) $\wedge$ Ncoll(F8,G8,A8)
- Cyclic(C9,E9,A9,B9) $\leftarrow$ Cong(D9,A9,D9,B9) $\wedge$ Cong(D9,E9,D9,A9) $\wedge$ Cong(D9,B9,D9,C9)
- Coll(F9,A9,C9) $\leftarrow$ Coll(C9,A9,F9)
- Coll(G9,A9,B9) $\leftarrow$ Coll(G9,A9,B9)
- Eqangle(A9,C9,E9,F9,A9,B9,E9,G9) $\leftarrow$ Perp(E9,G9,A9,B9) $\wedge$ Perp(E9,F9,A9,C9)
- Eqangle(F9,A9,F9,E9,G9,A9,G9,E9) $\leftarrow$ Coll(F9,A9,C9) $\wedge$ Coll(G9,A9,B9) $\wedge$ Eqangle(A9,C9,E9,F9,A9,B9,E9,G9)
- Cyclic(E9,A9,F9,G9) $\leftarrow$ Eqangle(F9,A9,F9,E9,G9,A9,G9,E9) $\wedge$ Ncoll(F9,G9,A9)

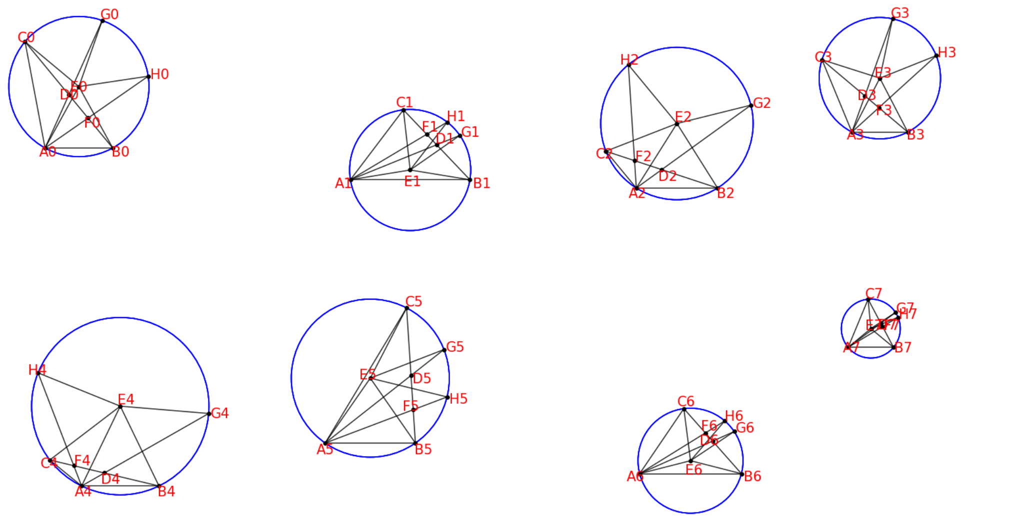

Figure 2: An example of image-form background knowledge.

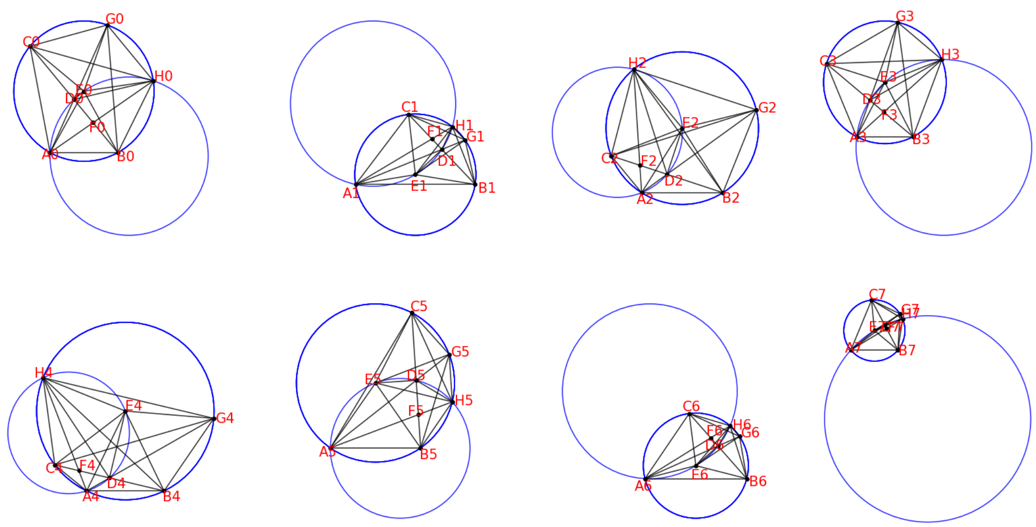

Figure 3: An example of image-form background knowledge + positive examples.

B.2 Raw sensory data

See Figure 2 for an example of image-form background knowledge. See Figure 3 for an example of image-form conclusions, which is a combination of background knowledge and positive examples.

C Translating GeoILP into natural-language form

For readers interested in the induction ability of large language models (LLMs), we provide guidance for translating GeoILP into natural-language form.

The prompt fed to LLMs should consist of three parts: task description, task data, and a command requiring LLMs to induce a hypothesis. The difficult part is the task data, containing background knowledge and positive & negative examples, which are ground atoms. The translation from symbolic atoms to natural-language forms varies in different domains. For *plane geometry*, AlphaGeometry (Trinh et al., 2024) provides several templates. For example, $\mathrm{Coll}(\mathrm{A}, \mathrm{B}, \mathrm{C})$ is translated into *A,B,C are collinear*. All the predicates used in GeoILP can be found in AlphaGeometry, and thus all GeoILP's atoms can be translated into natural-language forms. In addition, to enforce LLMs generating Horn rules (in natural-language forms), the command could ask LLMs to generate rules in the form of "If ... and ..., then ...", where each "..." corresponds to a ground atom. "..." can be repeated multiple times and must be concatenated by "and".