Neural-Symbolic Predicate Invention: Learning Relational Concepts from Visual Scenes*

Dmitry S. Kulyabov[1,2,*,†], Ilaria Tiddi[3,†] and Manfred Jeusfeld[4,†]

[1] *Peoples' Friendship University of Russia (RUDN University), 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation*
[2] *Joint Institute for Nuclear Research, 6 Joliot-Curie, Dubna, Moscow region, 141980, Russian Federation*
[3] *Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands*
[4] *University of Skövde, Högskolevägen 1, 541 28 Skövde, Sweden*

Abstract

The predicates used for Inductive Logic Programming (ILP) systems are typically elusive and need to be hand-crafted in advance limiting the generalization of the system when learning new rules without sufficient background knowledge. Predicate Invention (PI) for ILP is the problem of discovering new concepts that describe hidden relationships in the domain. PI can mitigate the generalization problem by inferring new concepts such that the system gains better vocabularies to compose logic rules to solve problems. Although several PI approaches for symbolic ILP systems exist, PI for NeSy ILP systems, which can deal with visual inputs to learn logic rules using differentiable reasoning, is relatively unaddressed. To this end, we propose a neural-symbolic approach to invent predicates from visual scenes for NeSy ILP systems based on clustering and extension of relational concepts. Our NeSy PI model handles visual scenes as its input using deep neural networks for the visual perception, and invents new concepts which are useful to solve the task of classifying complex visual scenes. The invented concepts can be used by NeSy ILP systems instead of hand-crafted background knowledge. Our experiments show that the PI model is capable of inventing high-level concepts and solving complex visual logical patterns. Moreover, the invented concepts are explainable and interpretable while also providing competitive results with the state of the art NeSy ILP systems with given knowledge.

Keywords

Predicate Invention, Inductive Logic Programming, Neural Symbolic Artificial Intelligence,

1. Introduction

Neural Symbolic Inductive Logical Programming (NeSy-ILP) learns logical programs from images. Such learned program sences the inherent logic in the Visual Scenes and consider them as rules for classification task. An example of such patterns is shown in figure 1. In order to

Woodstock'22: Symposium on the irreproducible science, June 07–11, 2022, Woodstock, NY

*You can use this document as the template for preparing your publication. We recommend using the latest version of the ceurart style.

*Corresponding author.

†These authors contributed equally.

kulyabov-ds@rudn.ru (D. S. Kulyabov); i.tiddi@vu.nl (I. Tiddi); Manfred.Jeusfeld@acm.org (M. Jeusfeld)
https://yamadharma.github.io/ (D. S. Kulyabov); https://kmitd.github.io/ilaria/ (I. Tiddi); http://conceptbase.sourceforge.net/mjf/ (M. Jeusfeld)
0000-0002-0877-7063 (D. S. Kulyabov); 0000-0001-7116-9338 (I. Tiddi); 0000-0002-9421-8566 (M. Jeusfeld)

Table 1
Comparison of language requirement between NeSy-ILP and NeSy-π. Meaning of abbreviations in the table: Const-Constant. BK-Background Knowledge. Ne-Pred-Neural Predicate.

	NeSy-ILP	NeSy-π
Const	$\texttt{Object : O1/O2, Color : Blue/Pink/Green}$ $\texttt{Shape : Sphere/Cube , Direction : Left/Right/Front/Behind}$	
Basic-Pred	$\texttt{color/2/obj, color; shape/2/obj, shape; dir/2/obj, obj}$	
BK	$\texttt{b\_sp(O1) : -color(O1, Blue), shape(O1, Sphere).}$ $\texttt{g\_sp(O1) : -color(O1, Green), shape(O1, Sphere).}$ $\texttt{g\_cu(O1) : -color(O1, Green), shape(O1, Cube).}$	-
Ne-Pred	$\texttt{left\_side/2/obj, obj; right\_side/2/obj, obj}$	-

solve such patterns, a set of predicates are required, for example `blue_sphere` defines the existence of a sphere with color blue.

However, to describe logical relations, these predicates in NeSy-ILP systems without PI support are either trained by neural network as pretrained neural predicates or explained by hand-crafted background knowledge. If any background knowledge is missing or neural predicates has wrong prediction, the optimal program searching can be failed. Besides, background knowledge can be hard to collect and usually provided by human experts, which limits the applying domain of ILP systems. Thus it is essential for system to conquer the problem of such dependence.

Predicate invention(PI)[?] is one direction to solve this problem, which is also a sub-problem in ILP. It works for ILP system to invent new predicates as symbols for new concepts from well designed basic predicates, which enlarge the expression of the language in ILP and consequently reduce the dependence on human experts. One simple example is the concept of a sphere with blue color, in NeSy-ILP system, the concept `blue_sphere` can be explained by a clause $\texttt{blue\_sphere(X) : -Color(X, blue), Shape(X, sphere)}$. which is given as BK knowledge, but with PI system, such concepts are learned from separate basic predicates $\texttt{Color(X, blue)}$ and $\texttt{Shape(X, sphere)}$ by concatenation.

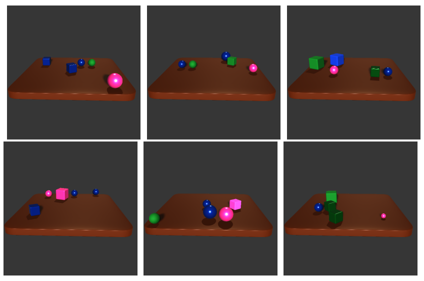


Figure 1: A logic Pattern in 3D scenes can be learned by Neural-Symbolic ILP system. Positive images on the first row, negative images on the second row. The truth pattern is: a blue sphere either locate on the left side of a green sphere or locate on the right side of a green cube.

However, most ILP system doesn't support PI. If such background knowledge is not provided, NeSy-ILP system may still find such concatenation of predicates but they are never considered as a single concept and be used as a whole in new clauses but only be search one predicate by one predicate, it has two major drawbacks, 1. it can lead to incomplete concept explanation if any single predicate is failed to be searched; 2. it can make the target clauses too long to search, which is not time efficiency at all. In our PI system, such background knowledge is not required but they can be collected during the target clause searching, which can both simplify and precise the target clause.

The goal of our work is to find a predicate invention pipeline and concatenate it with existing NeSy-ILP system, so that some high level concepts are not necessarily given from reasoning language directly. But they can be invented as needed during training. It can improve the system independence on human experts and also improves the generalization of AI models for adapting unseen tasks.

In this paper, we mainly focus on object property and their spatial relationships comparison. In order to using a single language to cover two objects spatial relations as many as possible, we designed an area division map called *target map* as shown in figure **??**, they are mapped to neural predicates as consider as basic predicates for PI model. As we shown in the experiment, the concepts like left, right, nearby supposed to be invented as new predicates during the training if any of them are needed to represent the target pattern in the positive images. This map both considers the distance and directions of the latent relation objects.

The target clause is searched in a top-bottom way, i.e. we start from most general rules and extend the rules by adding predicates as constraints. The size of searching domain for the target clause is growth exponentially over its length, which make the evaluation very time consuming. Since a naive pruning strategy can eliminate the global optimal clause. We designed an evaluation function based on the characteristics of necessity and sufficiency of the clause, which can scoring the searched clauses and keep the promising ones for further extension.

We propose NeSy-π, a neural-symbolic PI approach, based on *clustering* and *extension* of relational concepts, which is able to reasoning visual scenes without background knowledge. The knowledge can be summarized from scenes. It evaluates the clauses based on its characteristics of necessity and sufficiency. The procedure is repeated iteration by iteration, until the target clauses are found. We tested our approach on both 2D and 3D image patterns.

Comparing to existing approaches, our approach has following contributions

- We proposed a predicate invention approach for neural-symbolic ILP system.
- A formalized way to efficiently evaluate the clauses in visual scenes, which can be further used for pruning strategy.
- We discussed about *when* and *how* to invent new predicates based on *necessary* and *sufficient* judgment factors.
- The provide an implementation of this approach.

2. Background and Related Work

Although predicate invention has been proposed since 1988 by [1], it is always be considered as a major challenge. Most ILP system do not supports PI, including classic systems like Progol [2], TILDE[3] and modern system such as ATOM [4], LFIT [5].

Some works have been proposed for PI systems. However, none of these are focus on visual images and learning logical patterns existing in the visual scenes. [6] is an approach for predicate invention, which generate constrains from inconsistent hypothesis and further be used for pruning all the similar clauses. [7] use negation with PI to generalize the clause, which follows a bottom-top direction Several system uses [8], [9] meta-rules as templates for new predicates invention.

αILP (citation?) supports visual image as input but require background knowledge during the reasoning.

3. Target Map

Collect the facts from group of visual scenes and induce the common properties from the facts are can answer the question, why they are in the same group. However, common properties sometimes can be not obviously and hard to be described. and using a single word to summarize it to a suitable new concept is one of powerful ability of human-being. It simplifies the description of complex Predicate invention is an essential ability for ILP system to reasoning about

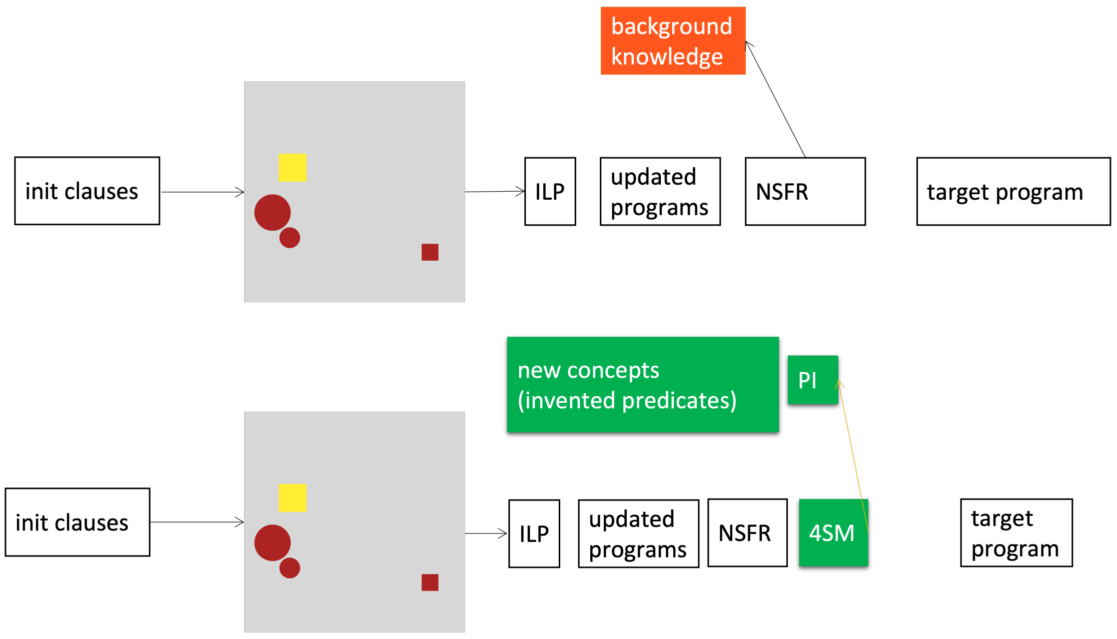


Figure 2: Predicate invention workflow.

The goal of our work is to find a predicate invention pipeline so that some high level concepts are not necessarily given from reasoning language directly. But they can be invented as needed during training. The invention is important since it is a way to acquire new knowledge and shows the ability of intelligence. On the other hand, it improves the generalization of AI models for adapting unseen tasks. The invention model improves the system to describe the problem more accurate.

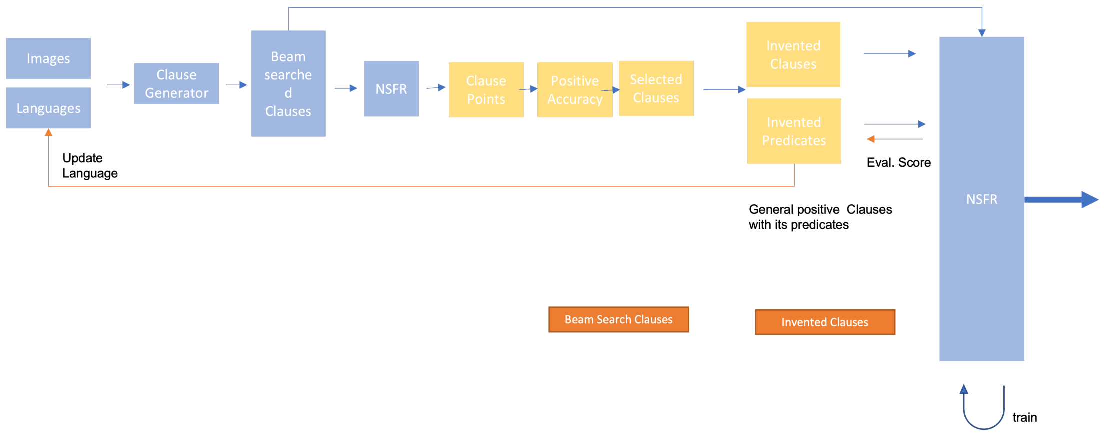


Figure 3: Predicate invention workflow.

However, the invention is still based on given background knowledge. It has to be simple, combinable, and mutually compatible, so that the new concepts are various and accurate.

In order to using a single language to cover two objects spatial relations as many as possible, we designed an area division map called *target map*. As shown in figure **??**, the surrounding area according to the reference object has been divided into 8 sub-areas. The areas in the target map are considered as background knowledge. The concepts like left, right, nearby supposed to be invented as new predicates during the training if any of them are needed to represent the target pattern in the positive images. This map both considers the distance and directions of the latent relation objects. Using the target map, multiple real-world related spatial concepts can be represented by combining some of atom areas, such as *left* (combining area 2,3,6,7), *right*(combining area 0,1,4,5), *nearby*(combining area 0,1,2,3) and so on.

4. NeSy-ILP

5. NeSy-π

The target of inductive logic programming is to find a target clause P for the positive patterns Q , such that the clause P describes some logical relations that exist and only exist in the positive patterns. Thus the target clause P is sufficient and necessary for the positive patterns Q.

$$P \Leftrightarrow Q$$

Definition 5.1 (PN pair). *A pair of positive and negative images.*

We prepare equal number of positive and negative images for evaluation, thus all the images can be paired from two groups. A new generated clause is evaluated on all the PN pairs. Evaluation on each pair getting two values, one from positive and one from negative. We can take negative value as x axis, positive value as y axis and draw all the evaluation result on a coordinate system. Thus each PN pair corresponds a point. We observed these points are only appears in the four clusters in the coordinate system(as show in figure 4 middle.), thus we fuzzy these points to only four areas (we use $f(P)$ to represent this step, where P is the points on the whole dataset.), each take one cluster. Thus for these two values, we only have $2^2 = 4$ different combinations. We named it as *four score map*.

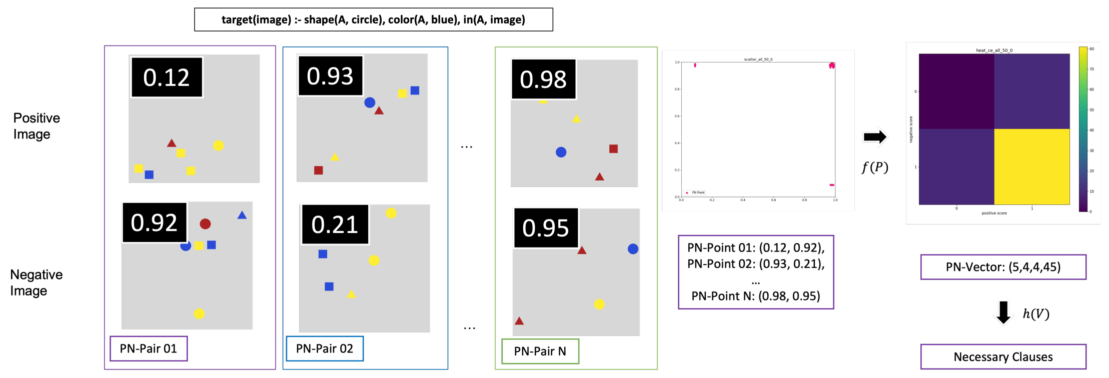


Figure 4: Left: PN pairs. **Middle**: PN pair scores on coordinate system. The points are clustered in four corners. These scores are evaluated by NSFR. **Right:**Four Score Map. The four score map illustrates the result of 4 kinds of evaluation on one clause, i.e. high positive-high negative, high positive-low negative, low positive-high negative, low positive-low negative.

A **four score map** fuzzy the evaluation result on an image to positive 1 and negative 0, map the positive image result and negative image result to four areas, namely $(0, 0)$, $(0, 1)$, $(1, 0)$ and $(1, 1)$. We found that four score map has good description for sufficient and necessary conditions in logic.

Base on the scoring areas of the predicates, we can classify them into several groups. Let N denotes the number of PN pairs.

Definition 5.2 (Sufficient and Necessary Clause). *Scores on* $(0, 1)$ *only, i.e.* $s_{01} = N$ *They are sufficient and necessary for the target pattern.*

Definition 5.3 (Necessary Clause). *Scores on* $(0, 1)$ *and* $(1, 1)$*, i.e.* $s_{01} + s_{11} = N$*. They are necessary for the target pattern. Note that they always true in the positive images, but also can be true in negative images.*

Definition 5.4 (Sufficient Clause). *Scores on* $(0, 0)$ *and* $(0, 1)$*,* $s_{00} + s_{01} = N$ *and* $s_{01} > 0$*. It induces directly some of positive patterns but can be failed on some other positive patterns. It never induces any negative patterns.*

The key idea of these definitions is to provide an evaluation metric for searched clauses. Since it is impossible to consider all the clauses for rule searching, an efficient pruning strategy is necessary. Our pruning strategy is mainly based on the satisfaction of the clauses on these definitions.

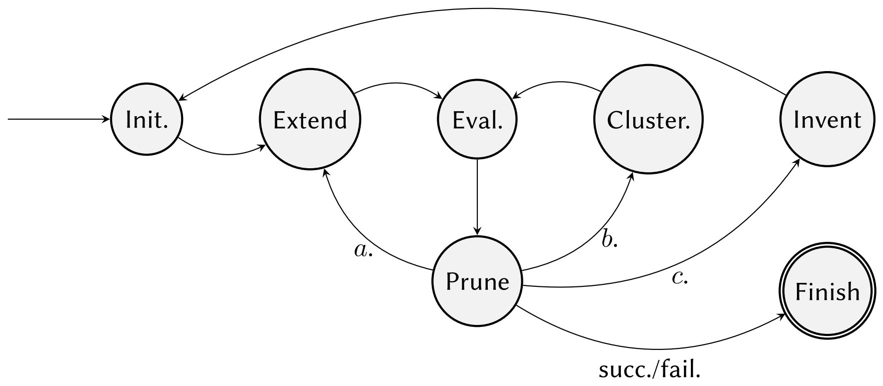


Figure 5: The workflow of NeSy-π. Edge a fires when maximum extension time is not exceed or no new predicates are invented after clustering. Edge b fires when maximum extension times is exceed. Edge c fires when new predicates are kept after pruning.

The workflow of NeSy-π is shown in figure 5. The system's work is basically to maintain a clause set $\mathcal{C}$. We add promising clauses inside and remove the abandoned ones from it. We explain each state as follows

Initial The searching is started from here. The input is the most general clause, i.e. the one that only states the existence of objects in the scene. For a scene with at most 2 objects, the initial clause set is $\mathcal{C} = \{\texttt{target(X)} : -\texttt{in(O1, X)}, \texttt{in(O2, X)}.\}$. Note that initial clause is a necessary clause since it is true on all the positive images.

Extend The input is either the initial clause set or the clauses set from pruning. For each clause in the set, we extend it with one predicate. Since there are many possible predicates to chose, any of them can be a promising one, thus we consider all the extension possibility into account. For example, to extend clause c, if there are 10 predicates in the language, the number of extended clauses of c is up to 10, if all the extended clauses are consistent with itself.

Evaluate In order to find all the promising clauses, we evaluate all the extended new clauses or clustered clauses. For each evaluation unit, we evaluate it on the whole dataset and classify it based on its four score map.

Pruning Pruning is based on scores of each clause or clause cluster getting from evaluate state. A prune strategy is required in this step. In this paper, we select only top ranked necessary/sufficient clauses/clusters, the rests are pruned.

Cluster The cluster is one step as preparation of high level concepts invention. For example, if $\texttt{Blue}, \texttt{Red}$ are the only appearing colors in all the scenes, the concept `two_objs_with_same_color` can be clustered by clauses

$$\texttt{two\_objs\_with\_same\_color}(\texttt{X}) \colon -\texttt{in}(\texttt{O1}, \texttt{X}), \texttt{in}(\texttt{O2}, \texttt{X}), \texttt{color}(\texttt{O1}, \texttt{Blue}), \texttt{color}(\texttt{O2}, \texttt{Blue}).$$
$$\texttt{two\_objs\_with\_same\_color}(\texttt{X}) : -\texttt{in}(\texttt{O1}, \texttt{X}), \texttt{in}(\texttt{O2}, \texttt{X}), \texttt{color}(\texttt{O1}, \texttt{Red}), \texttt{color}(\texttt{O2}, \texttt{Red}).$$

After pruning, if the max iteration of clause extension is exceed, the system will cluster the clauses. The input is clause set $\mathcal{C}$. We calculate all the subsets of $\mathcal{C}$. Each subset is considered as one cluster. The number of clusters is up to 2^n, where $n = |\mathcal{C}|$ i.e. the size of clause set. Since it grows exponentially, for a large n, we cannot evaluate all the subset. In order to improve the computation efficiency, the size of subset can be constrained with a threshold, to control the maximum size of the subset.

Inventing The input of inventing state is the set of clause clusters. Each cluster is considered as a new predicate, where its body is the clause clusters. After invention, the new predicates update the language and the system goes to initial state again.

Finish The system finish the rule searching if any sufficient and necessary predicate or clause is found or failed by exceeding the maximum iterations.

6. Experiments

To solve Kandinsky patterns, we provide following constants as background knowledge, shape: circle, square, triangle; color: red, yellow, blue; group_shape: line, unknown; distance: every 25 pixels as one unit. Besides, no further information is given, such as different color, different shape, etc. Some of the tested patterns are shown in figure 6.

Learning result of red-triangle pattern is shown as follows:

The learning details on each patterns in Kandinsky pattern and 3D Scenes is shown in table 2.

7. Conclusion

In this paper, we proposed an approach for Neural-Symbolic Predicate Invention. NeSy-PI is able to find the new knowledge and summarize it as new predicates, thus it requires less background

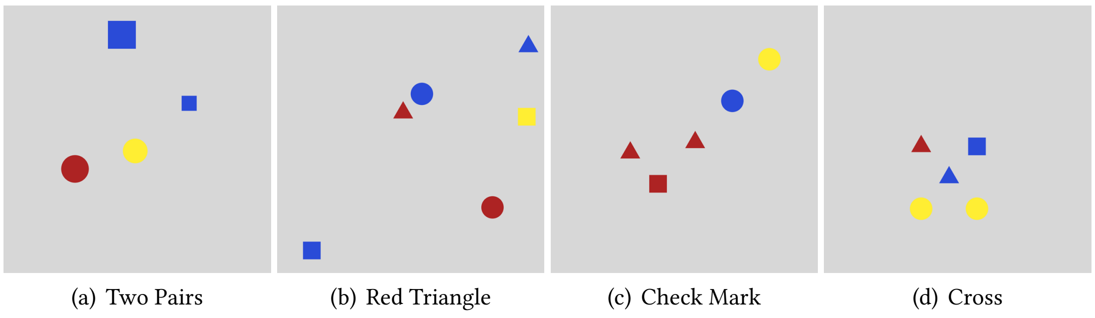


Figure 6: Kandinsky Patterns. **Two Pairs**: This pattern always have 4 objects inside, two of them has same color and same shape, the other two has same shape and different color. **Red Triangle**: This pattern always have a red triangle, and another object with different color and different shape nearby, the rest 4 objects are random objects. **Check Mark**: the positions of 5 objects consists of an outline of check mark. **Cross**: the position of 5 objects consists of an outline of a cross mark.

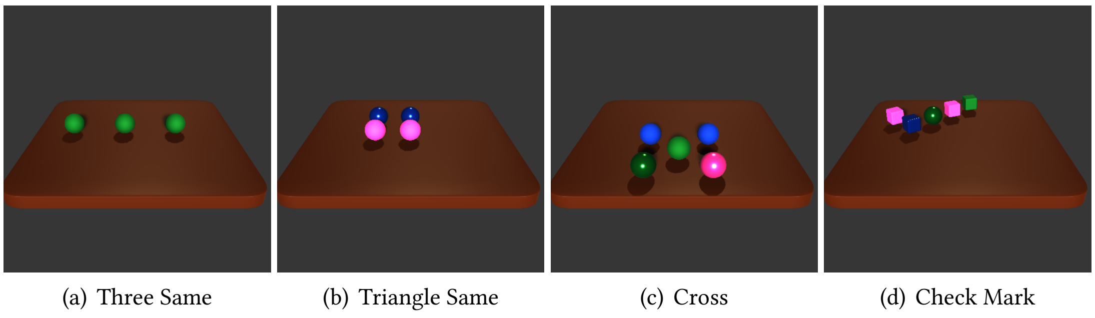


Figure 7: 3D Kandinsky Patterns. **Three Same**: This pattern always have 3 objects inside, three of them has same color and same shape. **Triangle Same**: This pattern always have 3 objects inside, three of them has same color and same shape. Besides, three objects form a triangle shape. **Cross**: this pattern always have 5 objects with same shape inside, the positions of 5 objects consists of an outline of cross mark. **Check Mark**: this pattern always have 5 objects with same color inside, the position of 5 objects consists of an outline of a cross mark.

knowledge for reasoning. In our experiment, we show that our PI model can successfully find the target program given only basic neural perception results and relevant constants, no further background knowledge is required. We also show our efficient prune strategy for predicate searching, the searching result is acquired faster and still sound.

```
(three same)
kp(X):-in(O1,X),in(O2,X),in(O3,X),inv_pred2(O1,O2,O3).
inv_pred2(O1,O2,O3):-color(O1,blue),color(O2,blue),color(O3,blue),
        in(O1,X),in(O2,X),in(O3,X),shape(O1,cube),shape(O2,cube),shape(O3,cube).
inv_pred2(O1,O2,O3):-color(O1,blue),color(O2,blue),color(O3,blue),in(O1,X),
        in(O2,X),in(O3,X),shape(O1,sphere),shape(O2,sphere),shape(O3,sphere).
inv_pred2(O1,O2,O3):-color(O1,green),color(O2,green),color(O3,green),in(O1,X),
        in(O2,X),in(O3,X),shape(O1,cube),shape(O2,cube),shape(O3,cube).
inv_pred2(O1,O2,O3):-color(O1,green),color(O2,green),color(O3,green),in(O1,X),
        in(O2,X),in(O3,X),shape(O1,sphere),shape(O2,sphere),shape(O3,sphere).
inv_pred2(O1,O2,O3):-color(O1,pink),color(O2,pink),color(O3,pink),in(O1,X),
        in(O2,X),in(O3,X),shape(O1,cube),shape(O2,cube),shape(O3,cube).
inv_pred2(O1,O2,O3):-color(O1,pink),color(O2,pink),color(O3,pink),in(O1,X),
        in(O2,X),in(O3,X),shape(O1,sphere),shape(O2,sphere),shape(O3,sphere).


(two pairs)
kp(X):-in(O1,X),in(O2,X),in(O3,X),in(O4,X),inv_pred30(O1,O2,O3,O4),
        inv_pred7(O3,O4).
inv_pred30(O1,O2,O3,O4):-in(O1,X),in(O2,X),in(O3,X),in(O4,X),inv_pred2(O1,O2),
        inv_pred2(O3,O4),inv_pred7(O1,O2).
inv_pred7(O1,O2):-in(O1,X),in(O2,X),shape(O1,cube),shape(O2,cube).
inv_pred7(O1,O2):-in(O1,X),in(O2,X),shape(O1,sphere),shape(O2,sphere).
inv_pred2(O1,O2):-color(O1,blue),color(O2,blue),in(O1,X),in(O2,X).
inv_pred2(O1,O2):-color(O1,green),color(O2,green),in(O1,X),in(O2,X).
inv_pred2(O1,O2):-color(O1,pink),color(O2,pink),in(O1,X),in(O2,X).

(cross mark)
kp(X):-in(O1,X),in(O2,X),in(O3,X),in(O4,X),in(O5,X),inv_pred15(O1,O2,O3,O4,O5).
inv_pred3(O2,O3,O4,O5):-in(O2,X),in(O3,X),in(O4,X),in(O5,X),rho(O2,O5,rho2),
        rho(O3,O4,rho2).
inv_pred6(O1,O3,O4,O5):-in(O1,X),in(O3,X),in(O4,X),in(O5,X),
        inv_pred3(O1,O3,O4,O5),phi(O3,O5,phi6),rho(O3,O5,rho1).
inv_pred15(O1,O2,O3,O4,O5):-in(O1,X),in(O2,X),in(O3,X),in(O4,X),in(O5,X),
        inv_pred6(O2,O3,O4,O5),shape(O1,cube),shape(O2,cube),shape(O3,cube).
inv_pred15(O1,O2,O3,O4,O5):-in(O1,X),in(O2,X),in(O3,X),in(O4,X),in(O5,X),
        inv_pred6(O2,O3,O4,O5),shape(O1,sphere),shape(O2,sphere),shape(O3,sphere).

(check mark)
kp(X):-in(O1,X),in(O2,X),in(O3,X),in(O4,X),in(O5,X),inv_pred3(O1,O2,O3,O5),
        inv_pred4(O2,O4,O5).
inv_pred3(O1,O2,O3,O5):-in(O1,X),in(O2,X),in(O3,X),in(O5,X),rho(O1,O2,rho0),
        rho(O3,O5,rho0).
inv_pred4(O1,O2,O5):-in(O1,X),in(O2,X),in(O5,X),rho(O1,O2,rho0),rho(O1,O5,rho0).
```

Figure 8: Learned rules by NeSy-π on 3D Kandinsky Pattern scenes.

To solve the nearby pattern, we invent the concept $inv\_pred\_1O\_1, O\_2$ by cluster clauses, which actually represent the concept *nearby*. It is necessary for the target patterns since it is true in all the positive images. It is also directly a sufficient predicate since no negative image has this pattern. The target clauses can be described as follows:

$$target(X) : -in(O_1, X), in(O_2, X), inv\_pred\_1(O_1, O_2).$$
$$inv\_pred\_1(O_1, O_2) : -in(O_1, X), in(O_2, X), a\_0(O_1, O_2).$$

Table 2
PI requirement and result on each patterns, where Ne-Preds means neural predicates.

Dataset	Patterns	Iterations	Times		Accuracy		# of Ne-Preds	
			αILP	NeSy-π	αILP	NeSy-π	αILP	NeSy-π
Kandinsky	red-triangle	3	0				0	4
	two-pairs	2	0				0	3
	check-mark		0				0	
	cross-mark		0				0	
3D Scenes	three same	2	0				0	1
	two-pairs	2	0				0	3
	cross mark	2	0				0	3
	check mark	1	0				0	2

$$inv\_pred\_1(O_1, O_2) : -in(O_1, X), in(O_2, X), a\_1(O_1, O_2).$$

7.0.1. Red Triangle

To solve the red triangle pattern, we require the concept *nearby* first by cluster clauses. It is necessary for the target patterns but not yet sufficient. Then we use beam search to extend this necessary clause with adding predicates, a sufficient and necessary clause is generated in several steps.

$$\begin{aligned} target(X) : -&in(O_1, X), in(O_2, X), inv\_pred\_1(O_1, O_2),\\ &shape(O_1, triangle), color(O_1, red),\\ &diff\_shape\_pair(O_1, O_2), diff\_color\_pair(O_1, O_2) \end{aligned}$$

$$\begin{aligned} inv\_pred\_1(O_1, O_2) &: -in(O_1, X), in(O_2, X), a\_0(O_1, O_2).\\ inv\_pred\_1(O_1, O_2) &: -in(O_1, X), in(O_2, X), a\_1(O_1, O_2).\\ inv\_pred\_1(O_1, O_2) &: -in(O_1, X), in(O_2, X), a\_2(O_1, O_2).\\ inv\_pred\_1(O_1, O_2) &: -in(O_1, X), in(O_2, X), a\_3(O_1, O_2). \end{aligned}$$

7.0.2. Online-Pair

For online-pair patterns, five objects are align on a line, whereas two of them have the same color and same pair. In this pattern, no necessary clauses can be found by clustering predicates. Thus we have to extend the clauses using beam search, until it is sufficient. Then for the sufficient clauses that we have found, we perform clustering for necessity.

A possible target clauses searching by predicate invention system is shown as follows

$$\begin{aligned} target(X) : -&in(O_1, X), in(O_2, X), in(O_3, X), in(O_4, X), in(O_5, X)\\ &inv\_pred\_1(O_1, O_2, O_3, O_4, O_5) \end{aligned}$$

$$\begin{aligned}
inv\_pred\_1(O_1,O_2,O_3,O_4,O_5) :& -in(O_1,X), in(O_2,X), in(O_3,X), in(O_4,X), in(O_5,X)\\
& a\_0(O_4,O_1), a\_2(O_2,O_1), same\_shape\_pair(O_2,O_4)\\
inv\_pred\_1(O_1,O_2,O_3,O_4,O_5) :& -in(O_1,X), in(O_2,X), in(O_3,X), in(O_4,X), in(O_5,X)\\
& a\_2(O_2,O_5), a\_2(O_5,O_3), color(O_3, blue)\\
inv\_pred\_1(O_1,O_2,O_3,O_4,O_5) :& -in(O_1,X), in(O_2,X), in(O_3,X), in(O_4,X), in(O_5,X)\\
& a\_3(O_2,O_1), a\_3(O_4,O_2), a\_3(O_5,O_4)
\end{aligned}$$

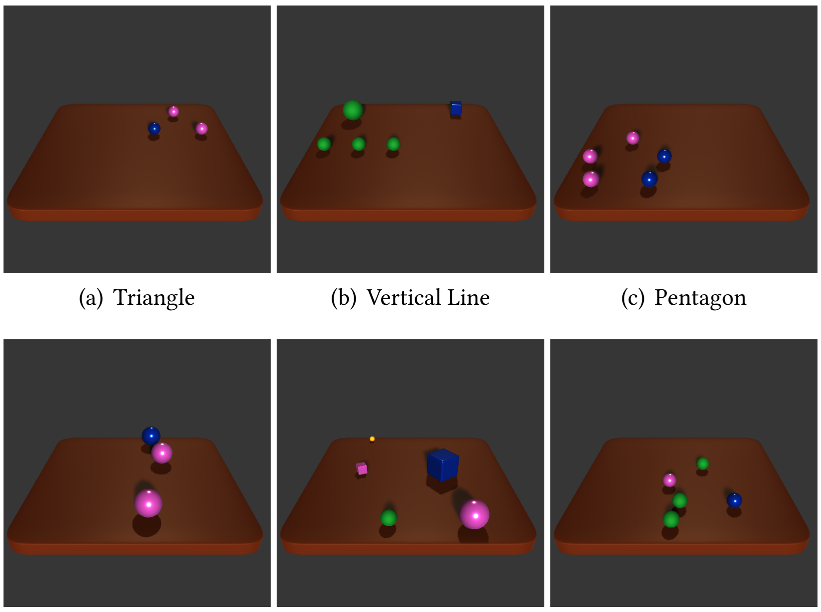
$$\begin{aligned}
inv\_pred\_2(O_1,O_2,O_3,O_4,O_5) :& -in(O_1,X), in(O_2,X), in(O_3,X), in(O_4,X), in(O_5,X)\\
& a\_0(O_4,O_1), a\_2(O_2,O_1), same\_shape\_pair(O_2,O_4)\\
inv\_pred\_2(O_1,O_2,O_3,O_4,O_5) :& -in(O_1,X), in(O_2,X), in(O_3,X), in(O_4,X), in(O_5,X)\\
& a\_1(O_3,O_1), a\_3(O_2,O_4), same\_color\_pair(O_2,O_4)\\
inv\_pred\_2(O_1,O_2,O_3,O_4,O_5) :& -in(O_1,X), in(O_2,X), in(O_3,X), in(O_4,X), in(O_5,X)\\
& a\_2(O_2,O_5), a\_2(O_5,O_3), color(O\_3, blue)\\
inv\_pred\_2(O_1,O_2,O_3,O_4,O_5) :& -in(O_1,X), in(O_2,X), in(O_3,X), in(O_4,X), in(O_5,X)\\
& a\_3(O_2,O_1), a\_3(O_4,O_2), a\_3(O_5,O_4)
\end{aligned}$$

Note that the predicate $inv\_pred\_1(O_1,O_2,O_3,O_4,O_5)$ share the same bodies $same\_shape\_pair(O_1,O_2), same\_color\_pair(O_1,O_2)$, which corresponds the concept $five\_object\_on\_the\_same\_line\_with\_one\_pair\_of\_same\_color\_and\_shape\_objects$. We can remove shared bodies in the clause definition. Then the predicate corresponds a simpler concept $five\_object\_on\_the\_same\_line$, which is a necessary predicate. It becomes sufficient after adding the deleted predicate back to. Removing shared bodies is not necessary but it can simplify the corresponding concepts of invented predicates.

Necessary Condition: Necessary conditions can be entailed by all positive patterns. They can be insufficient, thus they can also be entailed by negative patterns. Necessary predicates are invented as prerequisite for sufficient predicate invention. The necessity guarantees the searching for invented predicates is controlled in a proper scale, since only limited necessary conditions exist in the positive patterns. If we loose the necessity assumption, the searching domain for invented predicates can be huge since the conditions that doesn't exist in the positive patterns are irregular and countless. In order to satisfy the necessary condition, we can **cluster** independent clauses. Independent means the clauses do not share same predicates. It usually leads to invent high level concepts. For example, cluster low level concept *south*, *east*, *north*, *west*, those are independent concept with each other, we can have a high level concept *directions*. In this case, we need a new predicate to represent the cluster. Therefore the predicates are invented by necessary condition satisfied.

Think about "How" to cluster the existing clauses. Simply cluster all the clauses as one single concept cannot handle complicate rules.

If NSFR gives following clauses, then which of those clauses should be clustered as the new concept?


(a) Triangle (b) Vertical Line (c) Pentagon

(d) Triangle (e) Vertical Line (f) Pentagon

Figure 9: Hide Patterns.

If there is a pattern with multiple rules, a iteration-based approach should be proposed. First several iteration, it should be able to learn some new predicates, then use new predicates to describe the new scenarios.

New predicate has to be able to 100% describe the positive images. Otherwise, reconsider it.

$$
\begin{aligned}
target(X, Y) &\leftarrow atArea0(X, Y), atArea2(Y, X)\\
target(X, Y) &\leftarrow atArea1(X, Y), atArea3(Y, X)\\
target(X, Y) &\leftarrow atArea1(X, Y)\\
target(X, Y) &\leftarrow atArea2(X, Y)\\
target(X, Y) &\leftarrow atArea4(X, Y), atArea6(Y, X)\\
target(X, Y) &\leftarrow atArea5(X, Y), atArea7(Y, X)\\
target(X, Y) &\leftarrow atArea5(X, Y)\\
target(X, Y) &\leftarrow atArea6(X, Y)\\
target(X, Y) &\leftarrow pred1(X, Y)
\end{aligned}
$$

7.1. Chaining

[10] supports predicate invention.

8. Experiments

Using the baseline αILP, the nearby concept test accuracy is upto xxx, after xxx iterations.

9. Modifications

Modifying the template — including but not limited to: adjusting margins, typeface sizes, line spacing, paragraph and list definitions, and the use of the `\vspace` command to manually adjust the vertical spacing between elements of your work — is not allowed.

10. Template parameters

There are a number of template parameters which modify some part of the `ceurart` document class. This parameters are enclosed in square brackets and are a part of the `\documentclass` command:

```
\documentclass[parameter]{ceurart}
```

Frequently-used parameters, or combinations of parameters, include:

- `twocolumn` : Two column layout.
- `hf` : Enable header and footer[1].

11. Front matter

11.1. Title Information

The titles of papers should be either all use the emphasizing capitalized style or they should all use the regular English (or native language) style. It does not make a good impression if you or your authors mix the styles.

Use the `\title` command to define the title of your work. Do not insert line breaks in your title.

11.2. Title variants

`\title` command have the below options:

- `title`: Document title. This is default option.

  ```
  \title[mode=title]{This is a title}
  ```

  You can just omit it, like as follows:

  ```
  \title{This is a title}
  ```

[1]You can enable the display of page numbers in the final version of the entire collection. In this case, you should adhere to the end-to-end pagination of individual papers.

- `alt`: Alternate title.

```
\title[mode=alt]{This is a alternate title}
```

- `sub`: Sub title.

```
\title[mode=sub]{This is a sub title}
```

  You can just use `\subtitle` command, as follows:

```
\subtitle{This is a sub title}
```

- `trans`: Translated title.

```
\title[mode=trans]{This is a translated title}
```

- `transsub`: Translated sub title.

```
\title[mode=transsub]{This is a translated sub title}
```

11.3. Authors and Affiliations

Each author must be defined separately for accurate metadata identification. Multiple authors may share one affiliation. Authors' names should not be abbreviated; use full first names wherever possible. Include authors' e-mail addresses whenever possible.

`\author` command have the below options:

- `style` : Style of author name (chinese)
- `prefix` : Prefix
- `suffix` : Suffix
- `degree` : Degree
- `role` : Role
- `orcid` : ORCID
- `email` : E-mail
- `url` : URL

Author names can have some kinds of marks and notes:

- affiliation mark: `\author[<num>]`.

The author names and affiliations could be formatted in two ways:

1. Group the authors per affiliation.
2. Use an explicit mark to indicate the affiliations.

Author block example:

```
\author[1,2]{Author Name}[%
    prefix=Prof.,
    degree=D.Sc.,
    role=Researcher,
    orcid=0000-0000-000-0000,
    email=name@example.com,
    url=https://name.example.com
]

\address[1]{Affiliation #1}
\address[2]{Affiliation #2}
```

11.4. Abstract and Keywords

Abstract shall be entered in an environment that starts with `\begin{abstract}` and ends with `\end{abstract}`.

```
\begin{abstract}
  This is an abstract.
\end{abstract}
```

The key words are enclosed in a `keywords` environment. Use `\sep` to separate keywords.

```
\begin{keywords}
  First keyword \sep
  Second keyword \sep
  Third keyword \sep
  Fourth keyword
\end{keywords}
```

At the end of front matter add `\maketitle` command.

11.5. Various Marks in the Front Matter

The front matter becomes complicated due to various kinds of notes and marks to the title and author names. Marks in the title will be denoted by a star (⋆) mark; footnotes are denoted by super scripted Arabic numerals, corresponding author by an Conformal asterisk (*) mark.

11.5.1. Title marks

Title mark can be entered by the command, `\tnotemark[<num>]` and the corresponding text can be entered with the command `\tnotetext[<num>]{<text>}`. An example will be:

```
\title{A better way to format your document for CEUR-WS}

\tnotemark[1]
\tnotetext[1]{You can use this document as the template for preparing your
  publication. We recommend using the latest version of the ceurart style.}
```

`\tnotemark` and `\tnotetext` can be anywhere in the front matter, but should be before `\maketitle` command.

11.5.2. Author marks

Author names can have some kinds of marks and notes:

- footnote mark : `\fnmark[<num>]`
- footnote text : `\fntext[<num>]{<text>}`
- corresponding author mark : `\cormark[<num>]`
- corresponding author text : `\cortext[<num>]{<text>}`

11.5.3. Other marks

At times, authors want footnotes which leave no marks in the author names. The note text shall be listed as part of the front matter notes. Class files provides `\nonumnote` for this purpose. The usage

```
\nonumnote{<text>}
```

and should be entered anywhere before the `\maketitle` command for this to take effect.

12. Sectioning Commands

Your work should use standard LaTeX sectioning commands: `\section`, `\subsection`, `\subsubsection`, and `\paragraph`. They should be numbered; do not remove the numbering from the commands.

Simulating a sectioning command by setting the first word or words of a paragraph in boldface or italicized text is not allowed.

13. Tables

The "`ceurart`" document class includes the "`booktabs`" package — https://ctan.org/pkg/booktabs — for preparing high-quality tables.

Table captions are placed *above* the table.

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of tables, use the environment `table` to enclose the table's contents and the table caption. The contents of the table itself must go in the `tabular` environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules.

Immediately following this sentence is the point at which Table 3 is included in the input file; compare the placement of the table here with the table in the printed output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment `table*` to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 4 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed output of this document.

Table 3
Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
$	4 in 5	Used in business
Ψ^2_1	1 in 40,000	Unexplained usage

Table 4
Some Typical Commands

Command	A Number	Comments
`\author`	100	Author
`\table`	300	For tables
`\table*`	400	For wider tables

14. Math Equations

You may want to display math equations in three distinct styles: inline, numbered or non-numbered display. Each of the three are discussed in the next sections.

14.1. Inline (In-text) Equations

A formula that appears in the running text is called an inline or in-text formula. It is produced by the `math` environment, which can be invoked with the usual `\begin` ... `\end` construction or with the short form `$` ... `$`. You can use any of the symbols and structures, from α to ω, available in LaTeX [11]; this section will simply show a few examples of in-text equations in context. Notice how this equation: $\lim_{n\rightarrow \infty}\frac{1}{n} = 0$, set here in in-line math style, looks slightly different when set in display style. (See next section).

14.2. Display Equations

A numbered display equation—one set off by vertical space from the text and centered horizontally—is produced by the `equation` environment. An unnumbered display equation is produced by the `displaymath` environment.

Again, in either environment, you can use any of the symbols and structures available in LaTeX; this section will just give a couple of examples of display equations in context. First, consider the equation, shown as an inline equation above:

$$\lim_{n\rightarrow \infty}\frac{1}{n} = 0. \tag{1}$$

Notice how it is formatted somewhat differently in the `displaymath` environment. Now, we'll

Figure 10: 1907 Franklin Model D roadster. Photograph by Harris & Ewing, Inc. [Public domain], via Wikimedia Commons. (https://goo.gl/VLCRBB).

enter an unnumbered equation:

$$S_n = \sum_{i=1}^{n} x_i,$$

and follow it with another numbered equation:

$$\lim_{x \to 0} (1 + x)^{1/x} = e \tag{2}$$

just to demonstrate LaTeX's able handling of numbering.

15. Figures

The "`figure`" environment should be used for figures. One or more images can be placed within a figure. If your figure contains third-party material, you must clearly identify it as such, as shown in the example below.

Your figures should contain a caption which describes the figure to the reader. Figure captions go below the figure. Your figures should also include a description suitable for screen readers, to assist the visually-challenged to better understand your work.

Figure captions are placed below the figure.

16. Introduction

CEUR-WS's article template provides a consistent LaTeX style for use across CEUR-WS publications, and incorporates accessibility and metadata-extraction functionality. This document will explain the major features of the document class.

If you are new to publishing with CEUR-WS, this document is a valuable guide to the process of preparing your work for publication.

The "`ceurart`" document class can be used to prepare articles for any CEUR-WS publication, and for any stage of publication, from review to final "camera-ready" copy with *very* few changes to the source.

This class depends on the following packages for its proper functioning:

- `natbib.sty` for citation processing;
- `geometry.sty` for margin settings;
- `graphicx.sty` for graphics inclusion;
- `hyperref.sty` optional package if hyperlinking is required in the document;
- `fontawesome5.sty` optional package for bells and whistles.

All the above packages are part of any standard LaTeX installation. Therefore, the users need not be bothered about downloading any extra packages.

17. Citations and Bibliographies

The use of BibTeX for the preparation and formatting of one's references is strongly recommended. Authors' names should be complete — use full first names ("Donald E. Knuth") not initials ("D. E. Knuth") — and the salient identifying features of a reference should be included: title, year, volume, number, pages, article DOI, etc.

The bibliography is included in your source document with these two commands, placed just before the `\end{document}` command:

```
\bibliography{bibfile}
```

where "`bibfile`" is the name, without the "`.bib`" suffix, of the BibTeX file.

17.1. Some examples

A paginated journal article [12], an enumerated journal article [13], a reference to an entire issue [14], a monograph (whole book) [15], a monograph/whole book in a series (see 2a in spec. document) [16], a divisible-book such as an anthology or compilation [17] followed by the same example, however we only output the series if the volume number is given [18] (so series should not be present since it has no vol. no.), a chapter in a divisible book [19], a chapter in a divisible book in a series [20], a multi-volume work as book [21], an article in a proceedings (of a conference, symposium, workshop for example) (paginated proceedings article) [22], a proceedings article with all possible elements [23], an example of an enumerated proceedings article [24], an informally published work [25], a doctoral dissertation [26], a master's thesis: [27], an online document / world wide web resource [28, 29, 30], a video game (Case 1) [31] and (Case 2) [32] and [33] and (Case 3) a patent [34], work accepted for publication [35], prolific author [36] and [37]. Other cites might contain 'duplicate' DOI and URLs (some SIAM articles) [38]. Multi-volume works as books [39] and [40]. A couple of citations with DOIs: [41, 38]. Online citations: [42, 28, 43, 44].

18. Acknowledgments

Identification of funding sources and other support, and thanks to individuals and groups that assisted in the research and the preparation of the work should be included in an acknowledgment section, which is placed just before the reference section in your document.

This section has a special environment:

```
\begin{acknowledgments}
  These are different acknowledgments.
\end{acknowledgments}
```

so that the information contained therein can be more easily collected during the article metadata extraction phase, and to ensure consistency in the spelling of the section heading.

Authors should not prepare this section as a numbered or unnumbered `\section`; please use the "`acknowledgments`" environment.

19. Appendices

If your work needs an appendix, add it before the "`\end{document}`" command at the conclusion of your source document.

Start the appendix with the "`\appendix`" command:

```
\appendix
```

and note that in the appendix, sections are lettered, not numbered.

Acknowledgments

Thanks to the developers of ACM consolidated LaTeX styles https://github.com/borisveytsman/acmart and to the developers of Elsevier updated LaTeX templates https://www.ctan.org/tex-archive/macros/latex/contrib/els-cas-templates.

References

[1] Machine invention of first-order predicates by inverting resolution, in: J. Laird (Ed.), Machine Learning Proceedings 1988, Morgan Kaufmann, San Francisco (CA), 1988, pp. 339–352. URL: https://www.sciencedirect.com/science/article/pii/B9780934613644500402. doi:https://doi.org/10.1016/B978-0-934613-64-4.50040-2.

[2] S. H. Muggleton, Inverse entailment and progol, New Generation Computing 13 (1995) 245–286.

[3] H. Blockeel, L. De Raedt, Top-down induction of first-order logical decision trees, Artificial Intelligence 101 (1998) 285–297. URL: https://www.sciencedirect.com/science/article/pii/S0004370298000344. doi:https://doi.org/10.1016/S0004-3702(98)00034-4.

[4] J. Ahlgren, S. Y. Yuen, Efficient program synthesis using constraint satisfaction in inductive logic programming, J. Mach. Learn. Res. 14 (2013) 3649–3682.

[5] K. Inoue, T. Ribeiro, C. Sakama, Learning from interpretation transition, Machine Learning 94 (2014). doi:10.1007/s10994-013-5353-8.

[6] A. Cropper, R. Morel, Predicate invention by learning from failures, 2021. URL: https://arxiv.org/abs/2104.14426. doi:10.48550/ARXIV.2104.14426.

[7] D. M. Cerna, A. Cropper, Generalisation through negation and predicate invention, 2023. URL: https://arxiv.org/abs/2301.07629. doi:10.48550/ARXIV.2301.07629.

[8] R. Evans, E. Grefenstette, Learning explanatory rules from noisy data, CoRR abs/1711.04574 (2017). URL: http://arxiv.org/abs/1711.04574. arXiv:1711.04574.

[9] T. KAMINSKI, T. EITER, K. INOUE, Exploiting answer set programming with external sources for meta-interpretive learning, Theory and Practice of Logic Programming 18 (2018) 571–588. doi:10.1017/S1471068418000261.

[10] R. Evans, E. Grefenstette, Learning explanatory rules from noisy data (extended abstract), in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 5598–5602. URL: https://doi.org/10.24963/ijcai.2018/792. doi:10.24963/ijcai.2018/792.

[11] L. Lamport, LaTeX: A Document Preparation System, Addison-Wesley, Reading, MA., 1986.

[12] P. S. Abril, R. Plant, The patent holder's dilemma: Buy, sell, or troll?, Communications of the ACM 50 (2007) 36–44. doi:10.1145/1188913.1188915.

[13] S. Cohen, W. Nutt, Y. Sagic, Deciding equivalances among conjunctive aggregate queries, J. ACM 54 (2007). doi:10.1145/1219092.1219093.

[14] J. Cohen (Ed.), Special issue: Digital Libraries, volume 39, 1996.

[15] D. Kosiur, Understanding Policy-Based Networking, 2nd. ed., Wiley, New York, NY, 2001.

[16] D. Harel, First-Order Dynamic Logic, volume 68 of *Lecture Notes in Computer Science*, Springer-Verlag, New York, NY, 1979. doi:10.1007/3-540-09237-4.

[17] I. Editor (Ed.), The title of book one, volume 9 of *The name of the series one*, 1st. ed., University of Chicago Press, Chicago, 2007. doi:10.1007/3-540-09237-4.

[18] I. Editor (Ed.), The title of book two, The name of the series two, 2nd. ed., University of Chicago Press, Chicago, 2008. doi:10.1007/3-540-09237-4.

[19] A. Z. Spector, Achieving application requirements, in: S. Mullender (Ed.), Distributed Systems, 2nd. ed., ACM Press, New York, NY, 1990, pp. 19–33. doi:10.1145/90417.90738.

[20] B. P. Douglass, D. Harel, M. B. Trakhtenbrot, Statecarts in use: structured analysis and object-orientation, in: G. Rozenberg, F. W. Vaandrager (Eds.), Lectures on Embedded Systems, volume 1494 of *Lecture Notes in Computer Science*, Springer-Verlag, London, 1998, pp. 368–394. doi:10.1007/3-540-65193-4_29.

[21] D. E. Knuth, The Art of Computer Programming, Vol. 1: Fundamental Algorithms (3rd. ed.), Addison Wesley Longman Publishing Co., Inc., 1997.

[22] S. Andler, Predicate path expressions, in: Proceedings of the 6th. ACM SIGACT-SIGPLAN symposium on Principles of Programming Languages, POPL '79, ACM Press, New York, NY, 1979, pp. 226–236. doi:10.1145/567752.567774.

[23] S. W. Smith, An experiment in bibliographic mark-up: Parsing metadata for xml export, in: R. N. Smythe, A. Noble (Eds.), Proceedings of the 3rd. annual workshop on Librarians and Computers, volume 3 of *LAC '10*, Paparazzi Press, Milan Italy, 2010, pp. 422–431. doi:99.9999/woot07-S422.

[24] M. V. Gundy, D. Balzarotti, G. Vigna, Catch me, if you can: Evading network signatures with web-based polymorphic worms, in: Proceedings of the first USENIX workshop on Offensive Technologies, WOOT '07, USENIX Association, Berkley, CA, 2007.

[25] D. Harel, LOGICS of Programs: AXIOMATICS and DESCRIPTIVE POWER, MIT Research Lab Technical Report TR-200, Massachusetts Institute of Technology, Cambridge, MA, 1978.

[26] K. L. Clarkson, Algorithms for Closest-Point Problems (Computational Geometry), Ph.D. thesis, Stanford University, Palo Alto, CA, 1985. UMI Order Number: AAT 8506171.

[27] D. A. Anisi, Optimal Motion Control of a Ground Vehicle, Master's thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, 2003.

[28] H. Thornburg, Introduction to bayesian statistics, 2001. URL: http://ccrma.stanford.edu/~jos/bayes/bayes.html.

[29] R. Ablamowicz, B. Fauser, Clifford: a maple 11 package for clifford algebra computations, version 11, 2007. URL: http://math.tntech.edu/rafal/cliff11/index.html.

[30] Poker-Edge.Com, Stats and analysis, 2006. URL: http://www.poker-edge.com/stats.php.

[31] B. Obama, A more perfect union, Video, 2008. URL: http://video.google.com/videoplay?docid=6528042696351994555.

[32] D. Novak, Solder man, in: ACM SIGGRAPH 2003 Video Review on Animation theater Program: Part I - Vol. 145 (July 27–27, 2003), ACM Press, New York, NY, 2003, p. 4. URL: http://video.google.com/videoplay?docid=6528042696351994555. doi:99.9999/woot07-S422.

[33] N. Lee, Interview with bill kinder: January 13, 2005, Comput. Entertain. 3 (2005). doi:10.1145/1057270.1057278.

[34] J. Scientist, The fountain of youth, 2009. Patent No. 12345, Filed July 1st., 2008, Issued Aug. 9th., 2009.

[35] B. Rous, The enabling of digital libraries, Digital Libraries 12 (2008). To appear.

[36] M. Saeedi, M. S. Zamani, M. Sedighi, A library-based synthesis methodology for reversible logic, Microelectron. J. 41 (2010) 185–194.

[37] M. Saeedi, M. S. Zamani, M. Sedighi, Z. Sasanian, Synthesis of reversible circuit using cycle-based approach, J. Emerg. Technol. Comput. Syst. 6 (2010).

[38] M. Kirschmer, J. Voight, Algorithmic enumeration of ideal classes for quaternion orders, SIAM J. Comput. 39 (2010) 1714–1747. URL: http://dx.doi.org/10.1137/080734467. doi:10.

1137/080734467.

[39] L. Hörmander, The analysis of linear partial differential operators. IV, volume 275 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*, Springer-Verlag, Berlin, Germany, 1985. Fourier integral operators.

[40] L. Hörmander, The analysis of linear partial differential operators. III, volume 275 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*, Springer-Verlag, Berlin, Germany, 1985. Pseudodifferential operators.

[41] IEEE, Ieee tcsc executive committee, in: Proceedings of the IEEE International Conference on Web Services, ICWS '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 21–22. doi:10.1109/ICWS.2004.64.

[42] TUG, Institutional members of the TEX users group, 2017. URL: http://www.tug.org/instmem.html.

[43] R Core Team, R: A language and environment for statistical computing, 2019. URL: https://www.R-project.org/.

[44] S. Anzaroot, A. McCallum, UMass citation field extraction dataset, 2013. URL: http://www.iesl.cs.umass.edu/data/data-umasscitationfield.

A. Online Resources

The sources for the ceur-art style are available via

- GitHub,
- Overleaf template.