Incremental Graph-Based Discovery of Relational Concepts

No Author Given

No Institute Given

Abstract. Robots are becoming increasingly popular, however, they normally require careful programming by the user, of identifiable concepts, to perform simple tasks. In this paper, we describe a system, called ADC, that automatically discovers concepts in a robotic's domain, performing predicate invention. The robot creates an incremental graph-based representation with the information it gathers while exploring its environment, from which common sub-graphs are identified and relational concepts are induced using Inductive Logic Programming. Several concepts can be induced concurrently and the learned concepts can form arbitrarily hierarchies. The approach was tested for learning concepts of polygons, furniture in rooms, and places, like rooms and corridors, from office-like environments with a simulated robot.

1 Introduction

There is an increasing number of robots with more capabilities, however, before they can be used to perform simple tasks, the user normally needs to do some programming effort to simplify the reasoning process of the robot. This can be a time consuming process and involve several iterations until the robot is able to achieve the intended goals. In this paper, we describe how concepts can be automatically learned by a robot while it is exploring its environment.

Robot learning has been a very active research area. Much of the research has been based on reinforcement learning and programming by demonstration and, to a smaller extend, on concept learning. In concept learning, however, normally the agent learns a single concept at a time and the user is heavily involved in carefully preparing the learning settings.

Among the most commonly used approaches for concept discovery are those based on Inductive Logic Programming (ILP) with predicate invention. Systems using predicate invention can be classified into approaches based on reformulation and demand-driven approaches [21]. Systems using reformulation introduce new predicates produced by combining or restructuring other predicates to produce a more compact and precise theory (e.g., [13, 24, 25, 5]). The demand-driven approaches introduce new predicates when the vocabulary is not enough to induce a theory (e.g., [12, 1, 2, 4, 9, 22, 19, 11, 10]).

Although these systems have been successful for inducing concept definitions, there is a strong dependency on the user, the data is in many cases collected by the user before the learning process, the learning scenarios must be prepared, and there is no simultaneous concept learning, i.e., only one concept is learned at a time. In our research, we learn several concepts at the same time with demand-driven predicate invention, what the system learns depends on the information it gathers during its exploration and it is not known in advance, and we are able to learn hierarchies of concepts.

Other approaches have been proposed to learn hierarchical concepts (e.g., [4, 8, 3, 18, 17, 26, 23, 6]. But, these approaches usually are designed to work on databases or in controlled environments. In this paper, we proposed an algorithm for the discovery of concepts in a robotic domain. The robot gathers information while exploring its environment, identifies similar instances of potential concepts and learns relational concepts using ILP. Unlike previous systems, the proposed algorithm is designed to learn multiple concepts about objects and their relations, building hierarchies of concepts, based on data obtained incrementally from the direct experience of an agent with an unknown environment. We applied the proposed approach to learn concepts involving polygons, furniture, and spaces in an office-like environment that could be used for manipulation and navigation tasks.

The rest of the paper is organized as follows. In Section 2, the proposed system is described in detail. Section 3 presents the experimental results and Section 4 gives general conclusions and future research directions.

2 Automatic discovery of concepts through exploration

In our approach, called ADC, an agent automatically finds and collects instances of potentially relevant concepts while exploring its environment. Similar instances are clustered together and new concepts are induced from the clusters. These steps allow to learn concepts by demand-driven predicate invention from exploration. More specifically, given a set of primitive predicates to describe the environment (objects and relations among objects), ADC performs the following steps:

1. Explore: Traverse and sense the environment following an exploration strategy.
2. Transform: Given the information from the sensor readings verify which of the known predicates are satisfied with those readings, and represent them in an incrementally constructed graph, with arcs corresponding to relations and nodes to objects.
3. Induce: After reaching a goal state, find approximately equal frequent subgraphs from the constructed graph and create sets of similar subgraphs. For each set induce a new concept using an Inductive Logic Programming algorithm.
4. Simplify: Replace the induced concept in the original graph by a node and repeat the process until no more common subgraphs can be found.

The pseudo code of the proposed method is presented in Algorithm 1. These steps are detailed in the following subsections.

Algorithm 1 The ADC algorithm.

```
Given definitions for objects $O$ and relations among objects $R$ as background knowledge $BK$, a set of primitive actions $A$, an exploration strategy $E$, and a set of target goals $T$
Set groups $Cl = \emptyset$ and graph $G$ = null
repeat
  Perform action $a \in A$ to explore the environment using $E$
  Capture information from the robot's sensors and identify objects and their relations in the environment using its current $BK$ and add them to graph $G$, where objects/attributes = nodes, relations = edges
until current state $\in T$
while A frequent subgraph $g$ is found in $G$ do
  if $g$ is similar to the instances of an existing group $c_i \in Cl$ then
    Add $g$ to group $c_i$
    Let $D = Induce\ Concept(g, c_i, Cl)$ and $BK = D \cup BK$
  else
    Create a new group $c_j = \{g\}$, $Cl = c_j \cup Cl$
    Let $D = Induce\ Concept(g, c_j, Cl)$ and $BK = D \cup BK$
  end if
  $G \leftarrow$ Compress graph $G$ using $D$ (with instantiated variables)
  if $G$ can not be compressed using $D$ then
    $G \leftarrow$ Compress graph $G$ using $g$
  end if
end while
```

2.1 Graph Construction

During the first step, ADC incrementally builds a graph representing objects and relations identified in the environment. A set of basic objects (O) and relations (R) between objects identified from sensor readings (e.g., touches, on, near etc.) is provided as background knowledge by the user as well as some basic actions (e.g., move-forward, turn-right, etc.) to explore the environment. While the agent is exploring its environment with its current actions, it applies its current relations and use them to incrementally build a graph-based representation where objects and their attributes are represented as vertices and relations as edges. When the arity of a relation is greater than two and there is not direct mapping to a simple graph, we use conceptual graphs [20] to represent objects and their relations as concepts and conceptual relations (both, as vertices of a graph). In this way, the agent incrementally builds a graph until it reaches an intrinsic goal (defined by a function that considers the current state interesting in some way) or an extrinsic goal (defined normally by the user representing a goal of the current task), this process is ilustrated in the Figure 1.

2.2 Identification of Potential Concepts

Given the graph-based representation from the previous step, we use Subdue [8] to discover frequent subgraphs which best compress an input graph using

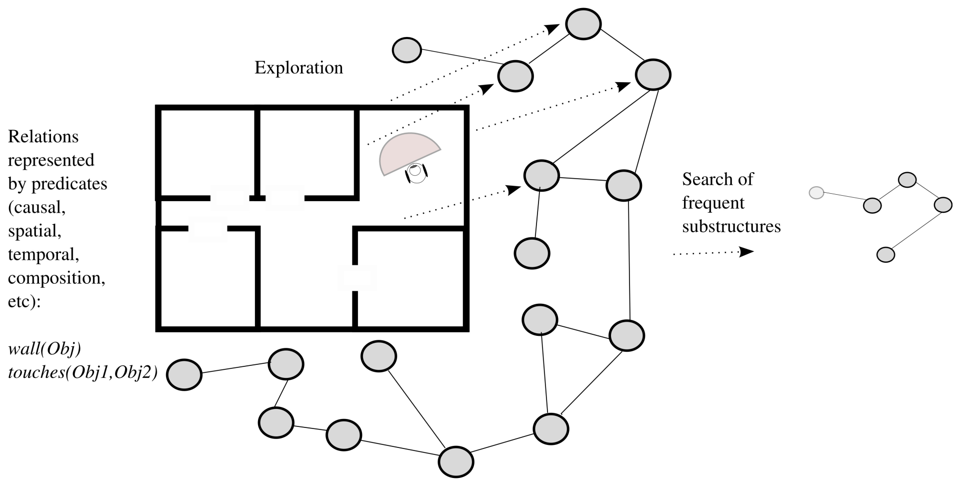
Exploration
Relations represented by predicates (causal, spatial, temporal, composition, etc):
wall(Obj)
touches(Obj1,Obj2)
Search of frequent substructures

Fig. 1. A robot explores a structural environment, using an initial background knowledge identifies basic elements and builds a graph from which frequent substructures are identified as instances of potential concepts.

an inexact matching measure and the minimum description length principle (MDL). The similarity between the substructures found by Subdue depends on a threshold value, which measures the fraction of the size (vertexes and edges) by which graphs can differ. A value of 0 (default value) implies that the graphs must match exactly. In our experiments this value was empirically set to 0.1 to promote finding similar (with small differences) instances of the same concept. All similar subgraphs are grouped together.

2.3 Grouping substructures in sets

When a substructure is discovered, it is added to an existing group or it is used to create a new group. Each group has similar subgraphs representing instances of potential new concepts. We use two similarity measures for adding a subgraph into a group.

With the first measure, a subgraph g_i is added to a group of subgraphs G_k, if most of the objects and relations of g_i can also be found in the most common objects and relations (R_k) found in the instances of that group. Each subgraph has its representation as first-order clause, where objects (vertices) and relations (edges) are literals in the body of the clause. In this way, given a new subgraph g_i with L_g literals. Let R_k be the most common literals in the subgraphs of group G_k (according to certain threshold value th_3). Let CL be the number of literals in g_i that are common to R_k. g_i is added to group G_k if there is a large proportion of the literals in g_i that is common to R_k, i.e.: $CL/|L_g| \geq th_1$, and if a large proportion of the most common literals in G_k (i.e., R_k) is part of the common literals in g_i, i.e.: $CL/|R_k| \geq th_2$.

With the second measure, a subgraph g_i is added to a group of subgraphs G_k if the average cost of structural changes (deletion, insertion and substitution of vertices and edges) to make $\underline{gi = g_k \in G_k \text{ for }}$ all the subgraphs in G_k is smaller than a threshold value, i.e.: $\overline{matchcost(g_i, g_k)} \leq th_4$. The $matchcost(g_i, g_k)$ is calculated based on the inexact matching measure of Subdue [8].

2.4 Induction of definitions of potential concepts

All the graphs in a group are represented as first-order clauses too, where the body of the clause is constructed with all elements in the subgraph, where each object $o_i \in O$ and relation $r_i \in R$ correspond to a literal of the body of a clause, and the head of the clause is defined with a new predicate name with its arguments composed by the distinctive arguments used in the body of the clause. Each time a new subgraph is included in a group, for inducing a new definition for the group, a set of negative examples is formed. This set includes instances and definitions of other groups and previously learned concepts. Also, the set includes negative examples artificially created by deleting relations from positive examples of the group where the new graph was added. This prevents ADC from over-generalizations from few examples. The group concept is re-induced each time a new subgraph is added to the group. The induction process is performed by Progol system based on inverse entailment and a general-to-specific search [15] (see Algorithm 2).

Algorithm 2 Induce definitions of potential concept

```
Induce Concept($g, c, Cl$)
Let $D_1$ = clause definition constructed from $g$
if group $c$ has associated a definition $D$ in $BK$ then
  Let $PEx$ = instances in $c$
  Add existing $D_N$ in $BK$ and instances from other groups in $Cl$ to negative examples $NEx$
  Create artificial negative examples, guided by current instances (positive examples) of $c$, and add them to $NEx$
  $NewD \leftarrow progol(PEx, NEx)$
  if $NewD$ could not be induced then
    Discard $NewD$
    Create a new group $c_j = \{g\}$, $Cl = c_j \cup Cl$
    Set $D = D_1$
  else
    Set $D = NewD$
  end if
else
  Set $D = D_1$
end if
Return $D$
```

2.5 Hierarchical concepts

The new induced definition can be used to compress the current graph by substituting the whole subgraph by a simple node with its corresponding arcs. If the new definition has variables, they must be instantiated before the concept can be used by Subdue for compressing the graph, see Figure 2. In our case, the variable arguments of the predicates are arbitrarily, although consistently, instantiated. The instantiations of the graph covered by the new definition are replaced by a single (new) predicate. The definition, with its original arguments (variables), however, is kept and can be used for further learning. The compressed graph is then used as new input to the algorithm to find new common subgraphs, from which new, hierarchical concepts can be induced.

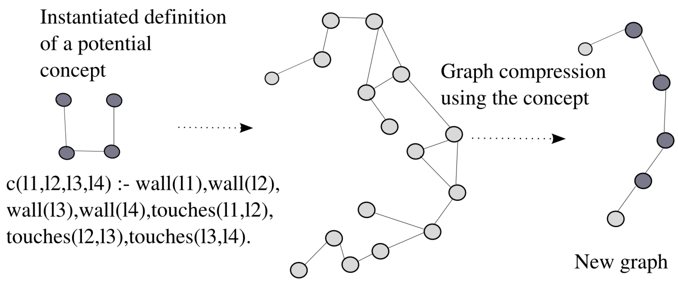
Instantiated definition of a potential concept
c(l1,l2,l3,l4) :- wall(l1),wall(l2), wall(l3),wall(l4),touches(l1,l2), touches(l2,l3),touches(l3,l4).
Graph compression using the concept
New graph

Fig. 2. Compression of a graph and generation of a new input graph.

3 Experiments

ADC was tested in three domains: polygons, furniture in rooms and structural domains (floors of buildings), to learn concepts that could be used for manipulation and navigation tasks.

3.1 Floor Maps

The first experiments were performed on simulation using *Player/Stage* [7] with a *Pioneer 2* robot equipped with a laser sensor (with a range of 180°). In this domain, the robot explored five maps of floors of buildings, shown in Figure 3. The robot built five graphs with 24, 38, 135, 45, and 50 nodes for each map, respectively. Different strategies can be used to explore the environment. They can be guided by unexplored areas, using an intrinsically motivated reward function or be guided by the user. In these experiments, the user guided the robot to cover the whole maps once.

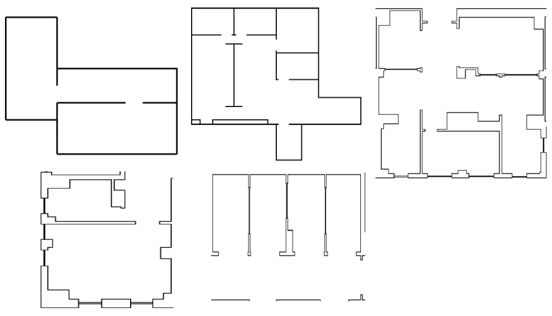

Fig. 3. Maps used by the robot to learn "structural" concepts.

BK	Name	Description
Object	wall/1	representing a wall/line segment in the environment with different lengths (short, normal, large and extra-large)
Relation	touches/2	when two walls form a corner

Table 1. Background knowledge provided to learn concepts about office-like environments.

The initial background knowledge provided to the robot consisted of two predicates, wall and touches (see Table 1). An incremental algorithm was used for the identification of lines from laser readings [16].

In this first set of experiments we contrasted the results obtained with ADC with those that can be obtained by Subdue, a graph-based discovery algorithm. Many induction systems, such as in Subdue, cannot be used in an incremental way or significantly decrease their performance if the data is given incrementally. Table 2 shows the number of concepts induced by ADC and by Subdue if graphs are given sequentialy (as in ADC) or when all the graphs are given at the same time as a single graph to Subdue. In the second and third columns, Single and HSingle refer to groups with a single subgraph and with a single hierarchical subgraph and Multiple refers to generalized subgraphs using ILP over a set of subgraphs in a single group.

In Subdue, if the graphs are given sequentially, as in our learning setting, a large number of isolated subgraphs are induced, many of which represent equivalent concepts. This behavior is reduced, to a certain extend, when all the graphs are given to Subdue at the same time as a single graph, as it is able to join some equivalent subgraphs from different graphs. Nevertheless, it still suffers from creating too specific concepts. Among the concepts learned by ADC, a user could identify concepts such as "room", "hallway" and "set of rooms" among others, some of them are illustrated in Figure 4. Some examples of definitions of concepts:

$$c1(A,B,C,D,E) : -wall(A), wall(B), wall(C), wall(D), wall(E),\\ touches(A,E), touches(B,C), touches(C,D), touches(D,E).$$

$$c4(A,B,C,D) :- c1(A), c1(B), c1(C), c1(D),$$
$$touches(A,D), touches(B,C), touches(C,D).$$

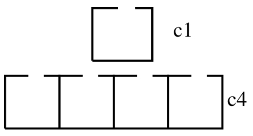

Fig. 4. Examples of induced concepts in the floor maps domain.

Table 2. Results from the office-like environment. In ADC the graphs were incrementally constructed from exploring the environment. In Subdue the complete graphs were given either as a single graph or sequentially.

Floor Maps	Single	HSingle	Multiple	Total
ADC	2	12	2	16
Subdue: Single graph ($G1 \cup \ldots \cup G5$)	3	20	–	23
Subdue: Sequential graphs (G1, ..., G5)	12	30	–	42

3.2 Polygons

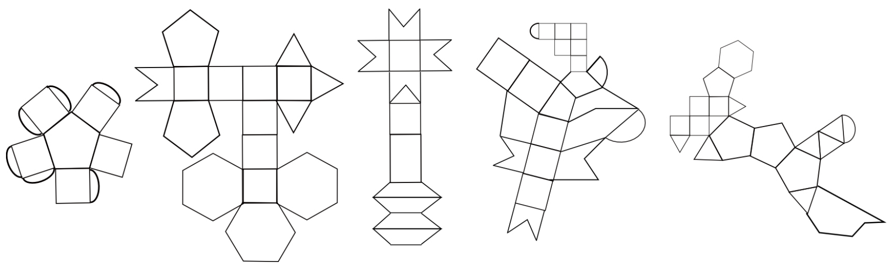

Fig. 5. Five figures used in the first domain, in the learning concepts about polygons, are illustrated.

In this case, we artificially created five graphs, shown in Figure 5, to learn different concepts related with polygons. Geometric figures are commonly encountered in robotic applications, so we wanted to know if ADC could discover basic geometric figures. The number of nodes in the graphs is 26, 55, 41, 61, and

56, respectively. The predicates used as background knowledge are line, curve and two relations representing angles (see Table 3).

BK	Name	Description
Object	line/1	representing a line segment
	curve/1	representing a curve segment
Relation	angcc$N/2$	a relation between two segments forming an angle greater than 180°, where N is the angle
	angcx$N/2$	a relation between two segments forming an angle less than 180°, where N is the angle

Table 3. Background knowledge provided to learn concepts about polygons.

ADC was able to identified several common subgraphs some of which were grouped to learn 5 general concepts: "triangle", "pentagon","square", "hexagon" and "irregular polygon with a concave angle". The number of concepts induced by ADC are given in Table 4, some of which are illustrated in Figure 6.

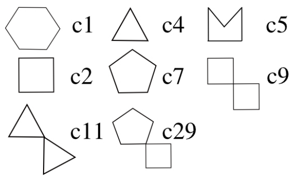

Fig. 6. Examples of induced concepts in the geometry domain.

Table 4. Concepts learned by ADC in the polygon and the furniture domains.

Domain	Single	HSingle	Multiple	Total
Polygons	5	24	5	34
Furniture	1	18	5	24

3.3 Furniture in rooms

In the last experiment, we used four graphs based on a real "Reading and Internet" room, shown in Figure 7. The number of nodes in the graphs is 77, 70, 31, and 22, respectively. The predicates provided as background knowledge represent several objects that can be identified in rooms, like flat surfaces, legs, lamps, etc., and relations among them, like next-to, above, etc. (see Table 5).

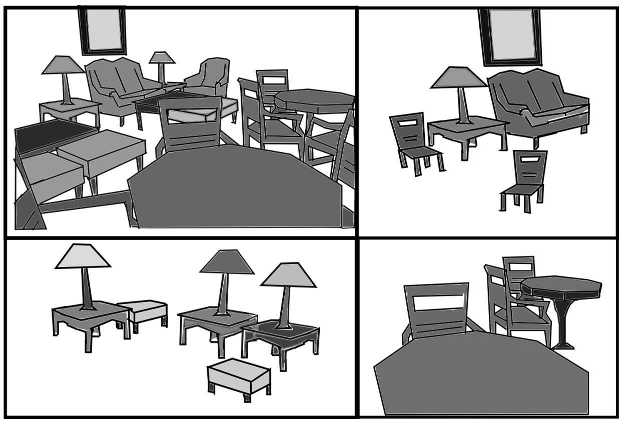

Fig. 7. Drawings based on a real "Internet and Reading" room used as basis for create four graphs of the second domain.

Table 4 presents the number of concepts discovered by ADC. The general concepts include: "table" (a flat board on four legs), "footstool" (a footstool on four legs), "chair" (back and seat on four legs), "lamp on table" (lampshade on light bulb base on a table), "chair in front of table" and "table next to table". Some examples are illustrated in Figure 8. In this domain the concept of "chair" also includes "armchair" and "sofa".

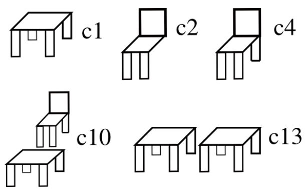

Fig. 8. Examples of induced concepts in the furniture domain.

4 Conclusions and future work

We presented an algorithm that incrementally learns concepts, by predicate invention, while exploring an environment. It identifies potential concepts using an incremental graph-based representation and a subgraph discovery algorithm. Similar subgraphs are grouped together and general concept definitions are induced using an ILP algorithm. We tested the approach on three domains with encouraging results.

BK	Name	Description
Object	flat_board/1	top of a table
	leg/1	legs as those present in tables, chairs, sofa, armchair or footstool
	light_bulb_base/1	bottom of a lamp
	lampshade/1	top of a lamp
	framed_picture/1	a framed picture
	footrest/1	top of a footstool
	seat/1	a seat for a chair, sofa or armchair
	back/1	the back of a sofa, armchair or chair
	arm_support/1	a support for arms in armchairs and sofas
Relation	on/2	when one object is on another object
	next_to/2	when one object is next to another object
	behind/2	when one object is behind another object
	in_front_of/2	when one object is in front of another object
	above/2	when one object is above another object

Table 5. Background knowledge provided to learn concepts about furniture.

As future work we would like to include an intelligent exploration strategy that could be coupled with the concept learning algorithm, borrowing some ideas from intrinsically motivated reinforcement learning. We are currently not inducing recursive definitions, including functional symbols or negated literals. We plan to extend the expressiveness of the induced concepts. We would also like to test the usefulness of the learned concepts to perform simple navigation and manipulation tasks. We are also exploring how to learn compound actions to learn complex tasks.

References

1. Bain, M., Muggleton, S.: Non-monotonic learning. In: Inductive Logic Programming, pages 145-161. Academic Press (1992)
2. Bostrom, H.: Predicate invention and learning from positive examples only. In: Tenth European Conference on Machine Learning, volume 1398 of Lecture Notes in Computer Science, pages 226-237. Springer (1998)
3. Chien, B., Hu, C., Ju, M.: Learning fuzzy concept hierarchy and measurement with node labeling. Information Systems Frontiers, 11:551-559 (2009)
4. Davis, J., Berg, E., Page, D., Santos Costa, V., Peissig, P., Caldwell, P.: Discovering latent structure in clinical databases. In: NIPS 2011 Workshop: From Statistical Genetics to Predictive Models in Personalized Medicine, Granada, Spain (2011)
5. Flach, P.: Predicate invention in inductive data engineering. In: European Conference on Machine Learning, volume 667, pages 83-94, Springer-Verlag, London, UK, UK (1993)
6. Fu, L.,Buchanan, B.: Learning intermediate concepts in constructing a hierarchical knowledge base. In: 9th international joint conference on Artificial intelligence - Volume 1, pages 659-666, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA (1985)

7. Gerkey, B., Vaughan, R., Howard, A.: The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In: 11th International Conference on Advanced Robotics, pages 317-326, Coimbra, Portugal (2003)
8. Holder, L. B., Cook, D., Djoko, S.: Substructure discovery in the subdue system. In: AAAI Workshop on Knowledge Discovery in Databases, pages 169-180 (1994)
9. Inoue, K., Furukawa, K., Kobayashi, I.: Abducing rules with predicate invention. In: 19th International Conference on Inductive Logic Programming, volume 667, pages 83-94, Springer-Verlag, Leuven, Belgium (2009)
10. Leban, G., Zabkar, J., Bratko, I.: An experiment in robot discovery with ILP. Zelezn, F. and Lavrac, N. editors, In: Inductive Logic Programing, volume 5194 of Lecture Notes in Computer Science, pages 77-90, Springer, Berlin, Heidelberg (2008)
11. Li, N., Stracuzzi, D., Langley, P.: Learning conceptual predicates for teleoreactive logic programs. In: Late-Breaking Papers Track at the Eighteenth International Conference on Inductive Logic Programming, Prague, Czech Republic (2008)
12. Morik, K., Wrobel, S., Kietz, J., Emde, W.: Knowledge acquisition and machine learning: theory, methods, and applications. Knowledge-based systems. Academic Press (1993)
13. Muggleton, S., Buntine, W.: Machine invention of first-order predicates by inverting resolution. In: 5th International Conference on Machine Learning (ICML 88), pages 339-352. Morgan Kaufmann (1988)
14. Muggleton, S., Feng, C.: Efficient induction of logic programs. New Generation Computing. Academic Press (1990)
15. Muggleton, S.: Inverse Entailment and Progol, New Generation Computing Journal, Vol. 13, pp. 245-286 (1995)
16. Nguyen, V., Gchter, S., Martinelli, A., Tomatis, N., Siegwart, R.: A comparison of line extraction algorithms using 2d range data for indoor mobile robotics. Autonomous Robots, 23(2):97-111 (2007)
17. Rivest, R., Sloan, R.: A formal model of hierarchical concept learning. Information and Computation, 114:88-114 (1994)
18. Rosca, J.: Hierarchical Learning with Procedural Abstraction Mechanisms. PhD thesis, Department of Computer Science, The College of Arts and Sciences, University of Rochester, Rochester, NY 14627, USA (1997)
19. Schmid, U., Hofmann, M., Kitzelmann, E.: Inductive programming: Example-driven construction of functional programs. Knstliche Intelligenz, 23(2):38-41 (2009)
20. Sowa, J.: Handbook of knowledge representation (Conceptual Graphs, Ch.5). Elsevier B. V. (2008)
21. Stahl, I.: Predicate invention in inductive logic programming. Advances in Inductive Logic Programming, Luc DeRaedt, editor, pages 34-47. IOS Press (1996)
22. Stanley, K., Domingos, P.: Statistical predicate invention. In: 24th annual international conference on machine learning ICML-2007, pages 433-440 (2007)
23. Tani, J. Nolfi, S.: Learning to perceive the world as articulated: An approach for hierarchical learning in sensory-motor systems. Neural Networks, pages 1131-1141. MIT Press (1999)
24. Wirth, R.: Learning by failure to prove. In: European Working Session on Learning, pages 237-251 (1988)
25. Wogulis, J., Langley, P.: Improving efficiency by learning intermediate concepts. In: 11th International joint conference on Artificial intelligence, volume 1, pages 657-662, Morgan Kaufmann, San Francisco, CA, USA (1989)
26. Zupan, B., Bohanec, M., Bratko, I., Demar, J.: Learning by discovering concept hierarchies. Artificial Intelligence, pages 211-242 (1999)