Predictive and Descriptive Approaches to Learning Game Rules from Vision Data

Paulo Santos[1], Simon Colton[2], and Derek Magee[3]

[1] Department of Electrical Engineering
Centro Universitário da FEI, São Paulo, Brazil
psantos@fei.edu.br
[2] Department of Computing, Imperial College, London, UK
sgc@doc.ic.ac.uk
[3] School of Computing, Leeds University, Leeds, UK
drm@comp.leeds.ac.uk

Abstract. Systems able to learn from visual observations have a great deal of potential for autonomous robotics, scientific discovery, and many other fields as the necessity to generalise from visual observation (from a quotidian scene or from the results of a scientific enquiry) is inherent in various domains. We describe an application to learning rules of a dice game using data from a vision system observing the game being played. In this paper, we experimented with two broad approaches: (i) a predictive learning approach with the Progol system, where explicit concept learning problems are posed and solved, and (ii) a descriptive learning approach with the HR system, where a general theory is formed with no specific problem solving task in mind and rules are extracted from the theory.

1 Introduction

For full autonomy, agents will require higher level abilities to reason about low level data collected from the environment, so that they can adapt to the environment and make discoveries which are necessary for the completion of tasks. Towards this goal, [12] proposes a cognitive vision system capable of learning protocols from the visual information of dynamic scenes using an inductive logic programming system (ILP). In that work the predictive ILP learning system Progol [10] was used to induce theories from the visual input system. In a *predictive* learning system a particular categorisation problem, or a set of such problems, are specified, and the learning system derives a theory which performs well when the derived rules in the theory are used for predicting the category of unseen examples. ILP systems which perform well at these tasks include the Progol [10] and FOIL [13] systems. An alternative mode within which ILP can be applied is known as *descriptive* learning. In such approaches, there is no particular problem to solve, other than that of discovering interesting concepts relating to the data provided and interesting facts which relate these concepts. Descriptive learning programs, such as the HR system [1], the CLAUDIEN [4] and WARMR [5] systems, produce theories containing examples, concepts which categorise the examples, conjectures which relate the concepts and explanations which illustrate the truth or falsity of the conjectures. The purpose of the experimentation and the reporting of it here is

primarily to compare and contrast descriptive and predictive approaches to learning rules from vision data. More specifically, the contribution of this paper is a comparison between Progol and HR on the task of learning protocols from visual observations reported in [12] whereas, in this earlier work, only Progol was employed. While qualitative comparisons of these systems are given in [1] and [2], this is the first quantitative comparison between them.

It is worth pointing out that, while there have been many approaches to learning from visual data (as overviewed in [12]), in this work we are interested in, first, to have access to the sound and completeness of the knowledge learned (cf. [12]) and, second, to further use the learned rules for higher level spatial and temporal reasoning, as experimented in [8]. Thus, we chose to employ an ILP system for the learning tasks.

The paper is organised as follows. In Section 1.1, we describe the vision system which generated the data from observing the dice game. We also briefly describe the Progol and HR systems. In Section 2, we describe the experimental setup, including the rules of the game, and how HR and Progol were employed for this application. In Section 3, we present the results from our experiments with the vision data.

1.1 The Vision System

The vision system used in this work is composed of a video camera observing a table top where a dice game is taking place. Figure 1 shows a picture of this setup, and also depicts a schema of the system modules.
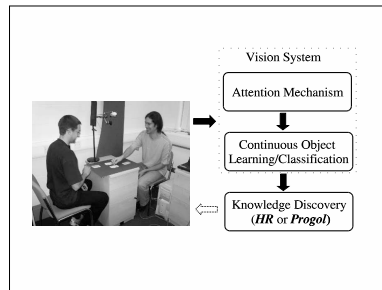

Fig. 1. A scheme of the experimental setup.

Visual information obtained by the vision system is turned into a symbolic description of states of the objects observed on the table top. This is used in turn as input data for the knowledge discovery module (i.e., HR or Progol here). Outputs from the knowledge discovery module are further used to drive a virtual agent to play the observed game.

To turn video streams into symbolic information, the vision system uses a spatio-temporal attention mechanism and an object classifier. In brief, the attention mechanism uses motion as the cue to select interesting portions of space-time. Based on a generic blob tracker [7] , this mechanism works on the principle of multi-modal background modelling and foreground pixel grouping. Thus, the bounding box, centroid location and pixel segmentation are extracted from any moving object in the scene in each frame of the video sequence. The attention mechanism identifies key frames where there is qualitatively no motion for a number of frames, which are preceded by

a number of frames containing significant motion. Each object in the selected frames is classified in two steps. First, features are extracted using banks of wavelets. Second, a set of example feature vectors is partitioned into classes using a graph partitioning method. The resulting partitions are used as supervision by a conventional statistical learning algorithm that provides models. In effect, for each object identified by the attention mechanism, a symbol is associated according to the specific model in which this object is classified. For example, the statement: $state([a, b], t_{20})$ indicates that there are two distinct objects in the scene, at time t_{20}, represented by the feature classes a and b respectively. A predicate $successor/2$ representing temporal succession is also provided for each subsequent pair of time points that appear in the argument of statements $state/2$. Atoms $successor(t_i, t_j)$ state that the time point t_i is a *successor* of the time point t_j. So, the vision system may output the following symbolic representation of the scenario: {state([a], t_{307}). successor(t_{309}, t_{307}). state([a, b], t_{309}).} This indicates that the system has observed a single object of class a on the table at time t_{307}, and the next recorded event was at time t_{309}, which was when there were two objects on the table, one of class a and the other of class b respectively. The vision system is described in more details in [12].

1.2 Progol and HR

For the experiments described here, we used CProgol4.5 [11], which is an implementation of the *definite modes language* [10]. This system allows the generalisation of a set of positive examples without the need of negative examples to guide the search. This characteristic suits well our aim to explain passive visual observation, since, in this case, negative examples cannot be easily input without supervision.

In brief, Progol works as follows. For each positive example, it generates a most specific Horn clause constructed according to user defined mode declarations. Mode declarations in Progol account for restrictions in the possible form of the proposed generalisations and hence enable the imposing of language biases on the search space. The initial most specific clause is further contrasted with the remaining examples in the search for a more general formula capable of subsuming most of the dataset. The selection of this formula is made according to statistics about its coverage [11]. The selected formula is added to the Progol's clause base, while the examples that were made redundant by this formula are retracted and the inductive process is repeated for the remaining examples until the last clause is reached. Therefore, Progol outputs the generalising rules in decreasing order of statistical significance.

In contrast to Progol, the HR system performs descriptive induction to form theories about given datasets using background information which includes a set of concepts and a set of axioms which relate the concepts. It has mostly been applied to mathematical domains where it has been used to make some interesting discoveries [2].

At the heart of HR's functionality is an ability to form new concepts from old ones, with the original concepts being supplied as background knowledge. HR uses 15 production rules to perform concept formation [2], each of which is either binary – taking two old concepts as input – or unary – taking one old concept as input, and each production rule generates a clausal, range restricted, definition of a new concept. Each production rule is parametrised so that the input set can lead to the production of multiple

concepts. Five important production rules are: the *compose* production rule, that takes the definitions of two concepts and combines them by conjoining the two sets of literals in their definitions; the *match* production rule, which takes a single old definition and unifies variables to produce a new one; the *exists* production rule introduces existential quantification by changing ground values to variables in a definition; the *split* production rule, performs instantiation in definitions; and, finally, the *negate* production rule, that negates literals within definitions.

Each induced concept is assessed using a variety of measures of interestingness [3], and this drives a heuristic search through the space of definitions. In addition to generating the definitions of new concepts, HR also calculates the success sets of the definitions and uses this data to induce hypotheses about the concepts empirically. In certain domains HR employs a third party automated theorem prover (usually Otter [9]) to attempt to prove that some of the hypotheses follow from a set of axioms supplied by the user. In other domains, HR may use the theorem proving functionality to show that a hypothesis can be proved from first principles given only some simple axioms. Such hypotheses are unlikely to be interesting to the user, hence HR uses theorem proving as a filter for dull conjectures.

New functionalities, allowing it to handle noisy data have recently been implemented in HR. As these processes were essential for this application to learning from vision data we provide a brief overview as follows. Whenever HR adds a new concept to the theory, it attempts to make certain types of conjecture involving the new concept. In particular, it attempts to make implication conjectures by finding concepts where the success set is a proper subset or superset of the success set for the new concept, and formulating the hypotheses accordingly. Previously, this was restricted to cases where the subset/superset relation was *exact*. This was because, in mathematical domains, a conjecture which is 99% true empirically is normally 100% false, as a counterexample is evident in the data. However, in non-mathematical domains, such conjectures are usually interesting, and the mis-matches may be due to noise or omissions in the data, rather than indicating the falsity of the hypothesis. Hence, we have relaxed the 100% constraint, and HR is now able to make *near-implications*. That is, given a user-specified percentage P, HR will add an implication to the theory if it is supported by at least $P\%$ of the data relating to the implication.

2 Experimental Setup

Eight distinct sessions where a human played the game (described in Section 2.1 below) were observed by the vision system. This produced eight separate datasets, which were used as inputs to the Progol and HR systems. Details of how Progol and HR were set up to perform the learning task are given in Sections 2.2.

2.1 Rules of the Dice Game

The dice game scenario assumed in this paper is described as follows. Two dice are thrown on an initially empty board. The game consists of keeping on the table the die with the highest face value while the other die is replayed. Both dice are withdrawn from the table when their faces show the same figure. Some example rounds of this game are shown in Figure 2. Typical outputs of the vision system are represented in the

subfigure captions, where a, b and d are constants for dice faces, [ ] represents an empty state (no dice on the table), and t_i ($i \in [1, \ldots, 5]$) are time points[1].

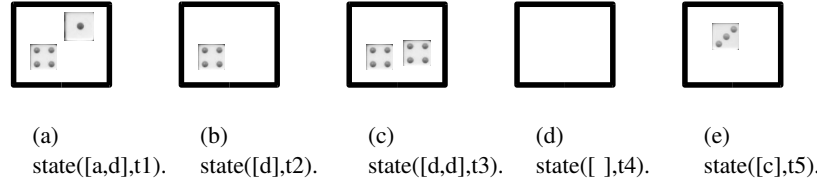| (a)	(b)	(c)	(d)	(e)
state([a,d],t1).	state([d],t2).	state([d,d],t3).	state([ ],t4).	state([c],t5).

Fig. 2. Five states of the dice game and their typical representation output from the vision system.

Given no knowledge about the numerical value of the die faces, the rules of the game can be expressed most succinctly using three general rules:

• An empty board is followed by a board with one die on it;
• A board with one die, D, on is followed by a board with D and another die on;
• A board with two equal dice faces showing is followed by an empty board;

and fifteen specific rules relating to the ordering of dice faces, as schematised by the following:

• A board with dice faces f and g showing is followed by a board with a single die with face g showing iff g > f.

2.2 Setup for Progol and HR

In order to get the best performance of Progol, we used its capability to deal with lists. Therefore, subsequent states in each dataset were re-written into a predicate representing the transition between pairs of states, $trans/4$. For instance, the sequence of states represented in Figure 2 would be re-written as:

$$trans([a, d], [d], t1, t2), trans([d], [d, d], t2, t3),$$
$$trans([d, d], [\ ], t3, t4), trans([\ ], [c], t4, t5).$$

Given that all 18 game rules we are interested in can be expressed in terms of a transition from one board state to another, this translation enabled us to use Progol in a single run to learn a set of rules for predicting such transitions.

As with Progol, some manipulation of the symbolic information provided by the vision system was required by HR. HR cannot yet handle list representations, so the state predicates such as *state([a,c],t1)* were split into three sets of background predicates. These were: (i) *0state/1* predicates, which indicate time points when the board was empty, (ii) *1state/2* predicates, which indicate the single object on a board at a particular time, and (iii) *2state/2* predicates, which indicate the pair of objects on a board

---

[1] It is worth pointing out that the assignment of dice faces to letters of the alphabet in lexicographical order, as shown in Figure 2, is for clarity purposes only; in fact, there is no preprocessing stage dedicated to this task, as the classification process assigns (consistently) any symbol to any particular scene object.

Dataset	Number of states	Number of faces	Mis-categorisations (%)	Coverage
1	55	72	25	0.47
2	76	99	4	0.73
3	69	96	25	0.67
4	108	147	8	0.73
5	58	84	12	0.47
6	72	96	8	0.47
7	61	27	7	0.47
8	61	78	11	0.53
all	560	699	12	0.93

Table 1. Details of observations made by the vision system

at a time point. In addition to the three predicates indicating states of boards, HR was also supplied with the *successor/2* predicate from the vision system.

We could have used the representation assumed for HR within Progol. However, in this case, Progol would not be capable of finding all the game rules in a single run, since different mode declarations would be needed. Therefore, we decided to compare the systems assuming the best domain representation respecting their potentialities and limitations.

3 Results

The observations from the vision system over eight distinct sessions are summarised in Table 1. In particular, we recorded the number of faces that were shown and the proportion of times a die face was mis-categorised by the system. We also recorded the proportion of all possible transitions from two faces to one face that occurred during each session and were correctly recorded (the column *coverage* in the table). For instance, if in a particular session, a board with dice faces a and d showing was followed by one with just die face d showing, this is counted in the *coverage* of 2-to-1 transitions observed during the session. This information is used to assess whether the sensitivity results obtained were due to the actual performance of the systems or were the effect of incomplete information in the datasets. Finally, details of a dataset with *all* the observations is given, as this was also used in learning sessions.

3.1 An Example Progol Theory

In the session described here, Progol was supplied with the vision data from dataset 2, represented as described in Section 2.2 above, and appropriate mode declarations. After a single run, Progol found rules (1) to (19) – given below in the order produced by Progol – to solve the learning problem. Note that the symbols A and B are variables over dice faces; C and D are variables over time points, and a, b, c, d, e and f are constants encoding the emergent classes representing the dice faces.

$(1)\ trans([\,], [A], C, D) : -successor(D, C).$
$(11)\ trans([e, a], [e], C, D).$
$(2)\ trans([A], [A, B], C, D) : -successor(D, C).$
$(12)\ trans([e, f], [f], C, D) : -successor(D, C).$
$(3)\ trans([A], [B, A], C, D) : -successor(D, C).$
$(13)\ trans([d, f], [f], C, D) : -successor(D, C).$
$(4)\ trans([b, a], [b], C, D).$
$(14)\ trans([c, a], [c], C, D).$
$(5)\ trans([A, B], [\,], C, D) : -successor(D, C).$
$(15)\ trans([c, d], [d], C, D).$
$(6)\ trans([f, d], [f], C, D).$
$(16)\ trans([f, a], [f], C, D) : -successor(D, C).$
$(7)\ trans([c, f], [f], C, D).$
$(17)\ trans([f, b], [f], C, D).$
$(8)\ trans([a, f], [f], C, D) : -successor(D, C).$
$(18)\ trans([f, e], [f], C, D) : -successor(D, C).$
$(9)\ trans([f, c], [f], C, D) : -successor(D, C).$
$(19)\ trans([e, b], [e], C, D).$
$(10)\ trans([c, e], [e], C, D).$

Rules (1) to (19) include most of the 18 rules of the dice game we identified in Section 2.1. Rules (1), (2) and (3) correctly represent two of the three general game rules we were looking for. In particular, rule (1) represents the fact that the subsequent state of an empty state is a state with only one object (one die face). Rules (2) and (3) represent the game rule that from a state with one die on the table, a state with two dice always follows. From this dataset, Progol did not find the general rule that a board state with two equal dice faces showing is followed by an empty board. Instead, it produced rule (5), which is an over generalisation of the expected rule due to noisy data. Rules (4) and (6) to (19) represent the ordering between the faces of the dice. In total, Progol reproduced 13 out of the 18 game rules we were looking for, hence scored 13/18 = 0.72 for sensitivity.

The same method applied to dataset 3, however, only produced a sensitivity value of 0.61. Part of the problem was due to the over generalisation of rules due to noise. In fact, according to table 1, dataset 2 had 4% of miscategorisation whereas dataset 3 had 25%. Therefore, Progol shows a gentle degradation with respect to noise as it managed to produce over 60% of the rules expected even when 25% of the data was corrupt, whereas it produced 72% of expected rules for a 4% noise-corrupted dataset.

It is worth noting that Progol constructed some of the rules above using the predicate $successor/2$, whereas it ignored it in some others, where probably there were fewer examples to support them. It can be argued that the relation representing the succession of time points could be dropped in this dataset as it is implicit in the representation of transitions as $trans/4$ relations. $successor/2$ statements were kept in the representation mainly in order to differentiate rules about transitions between pairs of states from rules about the transition between longer sequences of states, which are currently under investigation.

3.2 An Example HR Theory

The theory produced on running dataset 2 by HR contained 544 implicates, of which 151 were proven to follow from the axioms supplied to Otter (as described in Section 2.2). Such proved theorems were not shown to the user, and hence only 393 were presented. The following rules in the theory were identified as relating to the 18 game rules:

```
state(A,[C])    :- successor(A,B),   state(B,[nil]).
state(A,[nil])  :- successor(A,B),   state(B,[C,C]).
state(A,[b])    :- successor(A,B),   state(B,[b,a]).
state(A,[d])    :- successor(A,B),   state(B,[d,a]).
state(A,[e])    :- successor(A,B),   state(B,[e,a]).
state(A,[f])    :- successor(A,B),   state(B,[f,a]).
state(A,[d])    :- successor(A,B),   state(B,[b,d]).
state(A,[e])    :- successor(A,B),   state(B,[b,e]).
state(A,[f])    :- successor(A,B),   state(B,[b,f]).
state(A,[e])    :- successor(A,B),   state(B,[e,d]).
```

```
state(A,[f])    :- successor(A,B),    state(B,[f,d]).
state(A,[f])    :- successor(A,B),    state(B,[e,f]).
```

As these cover 12 of the 18 game rules, HR scored 12/18 = 0.67 for sensitivity. It is interesting to note that, as in the large majority of sessions, HR found the general result that a board state with two equal faces showing is followed by an empty board state, whereas in the majority of sessions, Progol did not find this. This is probably due to the different ways Progol and HR explore the hypothesis space, while the former tries to generalise most of the data given from each unexplained example (thus becoming subject to over-generalisations), the latter constructs conjectures according to some previously defined production rules.

Conversely, while Progol often found the general rule that a board state with one face showing is followed by a board state with that face showing, plus a second one, HR never found this rule. Theoretically, HR should have found this rule, and we are currently investigating its failure.

Note also that in this session, HR did not find the rule that a board state with face b and face f showing is followed by a board state with just face f showing, whereas Progol identified this as rule, (17) above. This was due to an interesting anomaly in the data: whenever face b was showing with something else, it was always showing with face f. Thus, when HR tried to invent the concept of board states with both b and f showing, it was conjectured to be equivalent to the concept of board states with b and *anything* else showing, hence the new concept was not allowed into the theory. This had the eventual effect that the rule we expected did not appear in the theory produced.

In other sessions, where the vision mis-categorisation rate was higher, like Progol, HR also produced false results such as:

```
state(A,[a])    :- successor(A,B),    state(B,[a,b])
```

These were produced because they were above the plausibility threshold of 50%. In the larger datasets, however, such incorrect rules were less common. Also like Progol, HR produced some over- generalised rules such as this:

```
state(A,[D,E])    :- successor(A,B),    state(B,[C,D])
```

which is the conjecture that a 1-object board is followed by a 2-object board, but doesn't refer to the retention of the object on the board.

3.3 An analysis of the results

To gain an impression of the sensitivity of our approaches, we ran Progol and HR as described in Section 2.2 for each of the eight vision observation datasets, and a ninth dataset consisting of all the data taken over the eight sessions[2]. We then recorded the proportion of the 18 rules identified in Section 2.1 which were found in the output from the systems. We then repeated the session with HR, but using a split-first search, where the split production rule is used greedily before any other. We ran a third session with HR using a compose-last search, where all production rules except compose are used greedily. In Table 2, we have recorded the sensitivity results from these sessions, along with the coverage of 2-state to 1-state board sequences as described above.

---

[2] The percentage of mis-categorisation for this ninth session (shown in the fourth column of Table 3.1) is a weighted mean value of the mis-categorisations of its compound datasets.

Dataset	Coverage	Progol	HR (breadth first)	HR (split first)	HR (compose last)
1	0.47	0.44	0.44	0.44	0.44
2	0.73	0.72	0.67	0.67	0.67
3	0.67	0.61	0.61	0.56	0.61
4	0.73	0.67	0.67	0.67	0.67
5	0.47	0.5	0.33	0.44	0.33
6	0.47	0.44	0.44	0.44	0.44
7	0.47	0.44	0.39	0.44	0.39
8	0.53	0.5	0.5	0.5	0.5
all	0.93	0.83	0.83	0.83	0.83

Table 2. Sensitivity analysis for HR and Progol on 9 datasets

With respect to sensitivity, Table 2 shows that HR and Progol perform as well as could be expected on the particular learning task we set, since the sensitivity obtained was always in the same range of the coverage of the dataset. Hence, as 15 of the 18 game rules were about particular sequences of boards, we could not expect Progol or HR to score much more than the coverage measure provided in Table 1 (and also reproduced in Table 2).

It is worth noting (from Table 2) that, for the task given, altering parameters for HR's search does not alter its sensitivity significantly.

We also analysed the selectivity and the efficiency of the systems. In both accounts, Progol performed better than HR. For instance, the number of rules produced by Progol for dataset 1 was 17, whereas HR produced nearly four hundred. In general the size of HR's theories were an order of magnitude larger than Progol's. Similarly, HR was more than an order of magnitude slower than Progol.

Both systems produced a number of false rules owing to noise in the data. While these do not reflect correct facts about the board game, they would not affect the way in which an agent would play the game, since the agent chooses the most statistically relevant rules first. Thus, in general, rules generated from noise are lower ranked with respect to true rules of the game (since these cover only few examples) and, thus, these do not interfere with the agent's actions.

4   Concluding remarks

We have presented and motivated the problem of learning rules of a dice game from vision data, in the larger context of the requirement for autonomous agents to learn understandable, logical, rules about protocol behaviours. We have presented a predictive ILP approach to this task using the Progol system and a descriptive ILP approach using the HR system, and we have compared and contrasted the systems in terms of their sensitivity when learning a particular required rule set. We conclude that, although Progol outperformed HR in most sensitivity results, the margin of difference does not allow us to affirm categorically that the former is a more suitable tool than the latter to the task given. Moreover, contrasted with the coverage of the observation, we can say that the two systems perform as well as could be expected given the data, since their selectivity was in the same range as the coverage.

Taking into account the selectivity and efficiency of the systems we can conclude that assuming a predictive approach for learning protocol behaviour from observations (following the footsteps described in [12]) is preferable in contrast to using a descriptive tool. However, the learning task proposed may have been more suitable for Progol than for HR, since the format of the rules sough were known *a priory* and every rule of the game could be found in a single run of Progol. In the general case of learning protocol behaviour, where there is no specific task to observe, the system does not have a well-defined learning task. As a consequence a predictive learning tool would be of no use, because it would not be possible to specify a single categorisation problem from which all the relevant rules of behaviour could be found. In this case, a descriptive learning approach, such as HR, would be more appropriate, since it is not constrained to solving a particular problem and hence can find rules relating to many different aspects of the background information. Also, as it does not have to worry about predictive accuracy, it will only supply faulty rules if they are sufficiently expressed in the given data. In practise, unfortunately, such an all-encompassing approach means searching a huge space of concepts and conjectures, and this may mean that rules of interest are not found efficiently enough.

References

1. S. Colton. *Automated Theory Formation in Pure Mathematics*. Springer-Verlag, 2002.
2. S. Colton and S. Muggleton. ILP for mathematical discovery. In *Proceedings of the 13th International Conference on Inductive Logic Programming*, 2003.
3. S. Colton, A. Bundy, and T. Walsh. On the notion of interestingness in automated mathematical discovery. *International Journal of Human Computer Studies*, 53(3):351–375, 2000.
4. L. de Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26:99–146, 1997.
5. L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, 1999.
6. R. King, K. Whelan, F. Jones, P. Reiser, C. Bryant, S. Muggleton, D. Kell, and S. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427:247–252, 2004.
7. D. R. Magee. Tracking multiple vehicles using foreground, background and motion models. *Image and Vision Computing*, 20(8):581–594, 2004.
8. D.R. Magee, C.J. Needham, P.E. Santos and S. Rao. Inducing the focus of attention by observing patterns in space In *IJCAI Workshop on Modelling Others from Observations (MOO 2005)*, pp. 47–52, 2005.
9. W McCune. The OTTER user's guide. Technical Report ANL/90/9, Argonne National Laboratories, 1990.
10. S. Muggleton. Learning from positive data. In *International Conference on Inductive Logic Programming*, 1996.
11. S. Muggleton. CProgol4.4: a tutorial introduction. In *Relational Data Mining*, pages 160–188. Springer, 2001.
12. C. Needham, P. Santos, D. Magee, V. Devin, D. Hogg and A. Cohn. Protocols from perceptual observations *Artificial Intelligence Journal*, 167: 103–136, 2005.
13. J. Quinlan and R. Cameron-Jones. FOIL: A midterm report. In *Proceedings of the 6th European Conference on Machine Learning (LNAI 667)*. Springer, 1993.
14. A. Stehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.