Combining Multiple Answers for Learning Mathematical Structures from Visual Observation

Paulo Santos and **Derek Magee** and **Anthony Cohn** and **David Hogg**[1]

Abstract.
Learning general truths from the observation of simple domains and, further, learning how to use this knowledge are essential capabilities for any intelligent agent to understand and execute informed actions in the real world. The aim of this work is the investigation of the automatic learning of mathematical structures from visual observation. This research was conducted upon a system that combines computer vision with inductive logic programming that was first designed to learn protocol behaviour from observation. In this paper we show how transitivity, reflexivity and symmetry axioms could be induced from the noisy data provided by the vision system. Noise in the data accounts for the generation of a large number of possible generalisations by the ILP system, most of which do not represent interesting concepts about the observed domain. In order to automatically choose the best answers among those generated by induction, we propose a method for combining the results of multiple ILP processes by ranking the most interesting answers.

1 Introduction

Two important requirements for an autonomous agent to understand and act in any real environment are the ability to hypothesise about perceptual information and the ability to act according to the acceptable behaviour (*protocol behaviour*) in that environment. In this paper we briefly introduce a system (in Section 2) that combines computer vision with an inductive logic programming (ILP) language, which is here used to generate high-level theories generalising the visual data. In this work we use PROGOL [11, 10] as the ILP language. Our system has been initially designed to learn protocol behaviours from sensor data that could further be used by a virtual agent. In this work, however, we use this system to show how some general mathematical structures (such as transitivity rules) can be induced from the visual data.

In the search for general rules we observed that noise in the data sets induced the ILP system to generate misleading answers that, even though generalising most of the input set, were not representing interesting concepts about the application domain. This work reports a possible solution to this issue. Our solution involves running PROGOL on multiple sets of observations of a particular aspect of the domain. The resulting *generalisation sets* are further combined into a final answer set in which, as we shall see, the most interesting formulae are ranked according to a voting criteria. This procedure falls in the class of ensemble methods in machine learning [4, 15].

The framework proposed in this this paper is evaluated on visual observation of two simple game scenarios. In the first, the system observes a dice game which is described as follows: two dice are initially thrown on a board, the game consists of keeping on the table the die with the greater face value while the other die is replayed. Both dice are withdrawn from the table when their faces show the same figure. The task of the logic engine was to find a definition of ordering between the dice faces from the action of taking one die out of the table. The challenge here is to generate this definition without any preconceived notion of number or any pseudo definition of ordering given as background knowledge to the ILP module. The only assumption underlying this domain was that there were six distinct symbols representing the different faces of a die. The rules of the second game experiment were analogous to the first but, instead of dice, the vision system was observing pairs of 3 different cards representing the figures of the game *paper scissors stone*. In this case the assumption of a one-to-one relation between the symbols given by the vision system and the objects in the game was dropped as the classifier was set to provide 15 classes for the three objects in the game (since it is unlikely that a system could reliably cluster data into exactly the right number of concepts). The task of the logic engine, in this case, is to provide the three axioms of equivalence given the cases where there was a draw (i.e. when both objects were taken out of the board).

This work makes three main contributions. First, we show how transitivity, reflexivity and symmetry axioms can be induced from the noisy data provided by a vision system. Second, in order to automatically choose the best answers among those generated by the ILP system, we propose a method for combining the results of multiple PROGOL processes by ranking the most interesting answers. Finally, the axioms sought were constructed without the assumption of any explicit background theory. The last point represents a key difference of the present investigation and previous research on high-level image interpretation [13, 5, 6], which were characterised by the interpretation of sensor data given domain-specific background theories.

By setting our system the task of learning basic axioms of mathematics we do not intend to develop a system to assist mathematicians (such as [3] and [7]) but to investigate how common sense knowledge can be automatically induced from computer vision data. The long term purpose of this research is provide an autonomous system with the necessary machinery that will allow it to formulate its own logical explanations about its environment.

2 The experimental setup

The experimental setup used in this work is composed of a video camera observing a table top where two players are engaged in playing a game, as shown in Figure 1. Figure 1 also depicts a schema of our prototype implementation.

[1] School of Computing, University of Leeds,Leeds, UK email: {psantos,drm,agc,dch}@comp.leeds.ac.uk

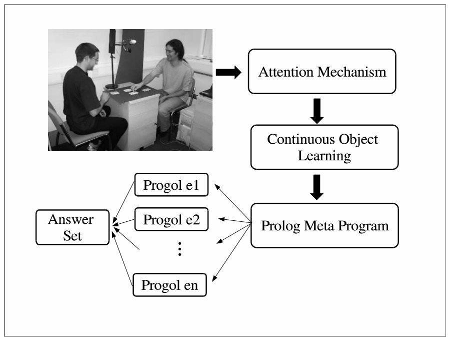


Figure 1: An scheme of the experimental setup.

The vision system consists of 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 [8], 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 [14]. The resulting partitions are used as supervision by a conventional statistical learning algorithm that provides models, which are further used to generate the symbolic information input to the ILP module. 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: $state([a, b], t_{20})$, indicates that there are two distinct objects in the scene, at time t_{20}, represented by the feature classes: a and b.

The output of the vision system is sent to a PROLOG meta program whose task is to re-write the input into multiple PROGOL experiments. As this processing stage is of central importance to the present paper, it will be discussed in detail in Section 3 below.

As a brief introduction to the PROGOL system, our prototype uses the CPROGOL4.5, which is an implementation of the *definite modes language* [11]. This 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. 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 data set.

3 Setting multiple experiments

As presented in the previous section, the output of the vision system is a sequence of state descriptions of the observed scene. The question now is to set appropriate PROGOL programs to search for general facts from this data. For instance, with appropriate mode declarations, PROGOL can be set to search for time-dependent rules involving sequences of state descriptions, or for patterns in the changes of objects in state descriptions, or for recursive rules comprising pairs of states. In this paper we concentrate on the last kind of rules. Even when assuming a single kind of rule, different PROGOL's runtime parameters (such as the maximum depth of resolutions) result in the construction of distinct formulae for the same set of data.

Changing the search settings and mode declarations when a particular kind of formulae is expected are straightforward tasks for experienced PROGOL programmers. However, with this paper our goal is to close the loop from data acquisition and data generalisation as we expect the results of this research to be one step towards the development of fully autonomous agents immersed in the real world. Therefore, a PROLOG meta program was developed to automatically set, run and evaluate different PROGOL experiments. The meta program, in this work, also has the task of selecting subsequent pairs of state descriptions from the vision data generating, for each pair, a predicate $trans(s_i, s_{i+1})$, read as *the transition from state s_i to state s_{i+1}*. Or aim, thus, is to search for pairwise recursive rules from the data, such as transitivity.

3.1 Looking for transitivity

Transitive inference is the process whereby the relation between two objects can be deduced given the relation of these objects with a third one. From reasoning about qualitative features to reasoning about social dominance relationships, transitive inference has not only been shown to be present in human adults and infants [1] but also in less evolved animals [9]. Whether or not this inference capability is innate in natural beings is not a question that we have expertise to answer. The importance and generality of transitive inference, however, seems to suggest that it lies in the foundations of reasoning. It is one of our goals, in this paper, to show that transitive rules can be induced from the data generated by a computer vision system. How such rules can be used in the inductive process to allow the generation of further, more complex, formulae and how they can be learnt from and imported to diverse domains are issues for further research.

In order to show the motivations of this paper's main argument, assume the dice scenario described in Section 1 above. In this case, for instance, the vision system provided the following sequence of state transitions, where t_i ($i \in [1, \ldots, n]$) are time points and b, c and d are three of the six classes representing the dice faces as given by the classification method: $state([b, c], t_1)$. $state([b], t_2)$. $state([b, d], t_3)$. $state([d], t_4)$. ...

Obtaining a transitive rule from this data using the meta program described seemed to be a straight forward task. Indeed, when no noise in the data is present and a fairly complete set of examples is provided, the meta program creates a PROGOL program rewriting the state facts above into: $trans([b, c], [b])$. $trans([b], [b, d])$. $trans([b, d], [d])$. ...

With this representation of the vision data, PROGOL generates (for variables A, B and C ranging over dice faces) the rule 1 below[2], which represents the transitivity of the relation that holds on the two objects in the first argument of $trans/2$ (in this paper we call formulae such as (1) *transitivity rules* for simplicity).

$$trans([A, B], [A]) :- trans([A, C], [A]), trans([B, C], [C]). \quad (1)$$

[2] Formula 1 was generated using mode declarations restricting the head of the formulae to be of the form $trans([+face, -face], [-face])$ and the bodies to be a conjunction of atoms of the form $trans([+face, -face], [+face])$ or $trans([+face, -face], [-face]))$ or $trans([-face, +face], [+face])$, where $+face$ and $-face$ are input and output variables ranging over dice faces.

However, perfect data and complete sets of examples are assumptions that do not hold in our setup. In the experiments conducted during this research, noise in the data was present as misclassification of the objects perceived. A typical generalisation set of actual vision data[3] contained formulae such as:

$$trans([A,B],[A]) :- trans([C,A],[A]), trans([C,B],[C]), trans([B,D],[D]). \quad (2)$$
$$trans([A,B],[A]) :- trans([B,A],[A]). \quad (3)$$
$$trans([A,B],[B]) :- trans([A,A],[A]), trans([A,B],[C]). \quad (4)$$

The expected transitivity rule was obtained from some data sets, amongst other (potentially spurious) rules such as (2) and (4) above. It is worth pointing out that the compressibility measurements provided by PROGOL were of an equivalent order for each data set, even though the formulae generated by PROGOL largely differed from one set to another. Moreover, the transitivity rule (when obtained) was not always given as the first rule in the results of PROGOL generalisations[4]. Another point worth noting is that some interesting formulae (initially not expected) were also obtained, such as a rule representing symmetry in the relation $trans/2$ (rule (3) above).

An obvious possible solution to enhancing the accuracy of the formulae obtained would be to consider a large data set and use the layered learning[5] feature in PROGOL to obtain the generalisations [12]. The results produced in this way (rules 5, 6 and 7 below) were better than the previous answers. However, the expected transitivity rule could not be constructed, instead we obtained a rule that is too general to be applied in practice (Formula 5). It is worth pointing out that running PROGOL on the same large data set, but without using layered learning, resulted in CPU elapsed-time failure.

$$trans([A,B],[B]) :- trans([C,A],[A]), trans([C,B],[D]). \quad (5)$$
$$trans([A,B],[A]) :- trans([B,A],[A]). \quad (6)$$
$$trans([A,B],[B]) :- trans([A,C],[D]), trans([C,E],[B]). \quad (7)$$

In effect, we wish to develop a method that automatically selects the most interesting formulae while keeping the potential spurious generalisations in a lower rank. Comparing the PROGOL answer from many different data sets, we noticed that particular spurious formulae were less frequent in the generalisation sets than the actual formulae sought. The meta program was then extended with a method for combining the resulting answers from multiple PROGOL processes (*multiple experiments*). The proposed method also ranks each resulting formula according to the number of times it subsumed, or was subsumed by, other formulae in the multiple results proposed by the PROGOL processes. The next section formalises this method.

4 Combining multiple ILP processes

The general problem in ILP is to generate the simplest consistent hypothesis H, given a background theory B and a set of examples E, such that $B, H \models E$.

However, as discussed in the previous section, if only (potentially noisy) positive examples constitute E, the simplest consistent hypothesis that best generalises the set of examples may not be the best formula for representing interesting concepts about a particular application domain. In order to obtain such formulae, we apply ILP to multiple example sets about a particular concept, various distinct mode declarations were also used for each example set. Formally, let $\{e_1, e_2, \ldots, e_n\}$ be a collection of example sets and $\{b_1, b_2, \ldots, b_m\}$ a set of mode declarations defining distinct search settings for a particular concept. Therefore, we have multiple ILP procedures each of which may be denoted as:

$$b_i, h_{i,j} \models e_j. \quad (8)$$

Thus, the symbol $h_{i,j}$ represents the generalisation set provided by running PROGOL with the data set e_j ($e_j \in [e_1, \ldots, e_n]$) and mode declarations b_i ($b_i \in [b_1, \ldots, b_m]$). The result of this procedure is, therefore, a collection of generalisations about a particular concept. The problem now is to combine each one of the $h_{i,j}$ into one single final *answer set*.

Let IAS (read as *intermediate answer set*) be the union of all generalisation sets $h_{i,j}$, i.e., $IAS = \bigcup_{i,j} h_{i,j}$. Let also $subsumption\_check(X, Y)$ be a binary relation that checks whether the formula X is more general than the formula Y. The procedure for ranking the multiple possible generalisations is described as follows. Each formula f in IAS is checked for whether $subsumption\_check(f, g)$ or $subsumption\_check(g, f)$ hold for every other formula g in IAS. If a formula subsumption is detected, the most general formula of the two is deleted from the intermediate answer set. Therefore, only the least general generalisation is kept. When there are no more formulae to be checked against f, f is deleted from IAS and inserted in the final answer set augmented with an annotation representing the number of subsumptions detected. This process is repeated until there are no remaining formulae in IAS. The last step in this algorithm is to sort the final answer set in terms of the formula annotations.

Instead of using formula subsumption, the process of ranking multiple possible generalisations could be done by counting the number of times a particular formula appears in the generalisation sets. However, due to noise in the data, PROGOL produces over-generalisations of the data (i.e. formulae that include – as particular cases – interesting concepts but, in general, predict incorrect data). Therefore, we decided to rank the rules by the number of times they subsume (or *are subsumed by*) others, keeping only the most specific, rather than the most general (usually over-general) ones. In this way, the over-general rules add in the selection of interesting formulae, but only the most specific formulae are output.

The motivation behind the method of combining the answers of multiple ILP processes proposed above, is to give more value to small but precise experiments than to large and noisy ones. Moreover, different aspects of a particular problem can be shown in different experiments. Combining the generalisation sets in the way proposed will keep the results of these particular experiments in the final answer set. This method also provides the opportunity to evaluate the multiple generalisations according to an *interestingness* criteria [2]. This possibility, however, is outside the scope of this paper.

5 Basic mathematics from visual information

In this section we recall the example domains introduced in Section 1 and discuss the results obtained by applying the ideas proposed above. Both domains involve exposing to the vision system pairs of objects that satisfy a particular relation. Having no previous knowledge about the object pair or about the possible relation between

[3] Using the same mode declarations as for (1) above.

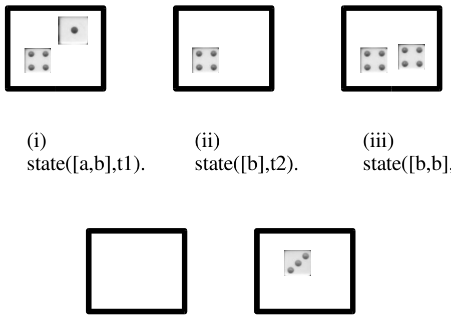
[4] PROGOL orders its output from the formula that most compresses the data set.

[5] Briefly, layered learning constructs an initial rule (from a subset of the example set) that is further tested and improved in further data subsets.

them, the system has to induce axioms about this relation by considering the transitions in the game.

Experiment 1: learning a transitivity rule

An example of some rounds of the first game considered is shown in Figure 2; typical outputs of the vision system are represented in the subfigure captions. In order to obtain a transitivity rule, the meta


(i) state([a,b],t1). (ii) state([b],t2). (iii) state([b,b],t3).

(iv) state([],t4). (v) state([d],t5).

Figure 2: Five states of the dice game.

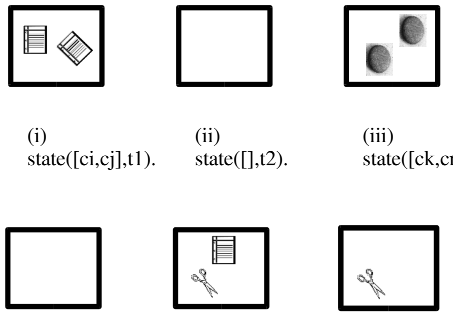
program was set to select the transitions between pairs of states, each containing at least one object. Thus, assuming Figure 2, PROGOL is input with statements of the form: $trans([a,b],[b])$. and $trans([b],[b,b])$.

Eight distinct data sets were used in this experiment, each containing an average of 50 rounds of the game. Recalling the discussion in Section 3.1, the results of running PROGOL for individual data sets generated one distinct set of answers for each set. Within the obtained answers the symmetry and transitivity axioms for the $trans/2$ relation were amongst a large set of spurious formulae generated by PROGOL (29 formulae in total). In contrast, the result of running PROGOL on the union of the eight sequences of data resulted in CPU elapsed time failure. Using the layered learning feature in PROGOL we obtained the Formulae 5, 6 and 7, which did not contain the transitivity axiom as discussed above.

By using the method described in Section 4, however, we obtained the symmetry axiom ranked as the most voted for formula (4 votes), a rule representing the transitivity axiom was ranked as the second formula (with 2 votes) along with one spurious formula, the remainder formulae received one vote each. Therefore, not only has our method given a better rank to the expected formulae but it avoided both rejecting interesting formulae (as occurred when applying layered learning), and the excessive time complexity of applying PROGOL on a large data set.

Experiment 2: learning an equivalence relation

In order to allow the system to learn the axioms of equivalence, an over classification of the paper-scissors-stone figures was assumed. The three objects of the game were classified into 15 distinct classes. From the symbolic learning standpoint, this assumption is equivalent to showing, at each round of the game, one of 15 different shapes of papers, scissors and stones. Therefore, the axioms of equivalence are obtained from analysing the states in which a draw occurs, i.e., states


(i) state([ci,cj],t1). (ii) state([],t2). (iii) state([ck,cr],t3).

(iv) state([],t4). (v) state([cs,cw],t5). (vi) state([cn],t5).

Figure 3: Six states of the paper-scissor-stone game.

containing pairs of objects that are followed by an empty state. It is worth noting that, in this experiment, we are not interested in inducing the rules of the game in question, but in obtaining characteristics of the equivalence between objects in this context.

Eight data sets (containing an average of 30 examples each) were utilised in this experiment. In contrast with the previous domain, the objects assumed in the present case were richer in textures than the dice faces. Therefore, the classification provided by the vision module was considerably more accurate in this domain than in the experiment 1. Therefore, fewer formulae were proposed as output of running PROGOL on the provided data sets. Moreover, in most cases, the sought axioms of equivalence were obtained as the first rules output. We could not find suitable mode declarations, however, that would allow PROGOL to find, on a single run, the reflexivity, symmetry and transitivity axioms. The method proposed in Section 4 was used to guarantee that the three expected rules are always given as results when implicit in the data sets. As a result, the following formulae (and their respective annotations) were obtained from our PROLOG meta program:

$$(trans([A,B],[]) :- \; trans([A,C],[]), trans([C,B],[]), 6) \tag{9}$$

$$(trans([A,A],[]), 4) \tag{10}$$

$$(trans([A,B],[C]) :- \; trans([A,D],[C]), trans([B,A],[C]), 3) \tag{11}$$

$$(trans([A,B],[]) :- \; trans([B,A],[]), 2) \tag{12}$$

$$(trans([A,B],[]) :- \; trans([C,A],[]), trans([C,B],[]), 2) \tag{13}$$

$$(trans([A,B],[]) :- \; trans([C,A],[]), trans([B,C],[]), 2) \tag{14}$$

$$(trans([A,A],[B]) :- \; trans([A,C],[B]), 1) \tag{15}$$

$$(trans([A,B],[C]) :- \; trans([A,D],[C]), trans([B,D],[C]), 1). \tag{16}$$

This answer set shows that a transitivity rule (Formula 9) was the most scored formula (6 votes) followed by the reflexivity axiom (Formula 10). The symmetry axiom (Formula 12) came in fourth, preceded by a spurious rule (Formula 11). Therefore, the ranking method does not guarantee that the most scored formulae are always the most meaningful, nor does it guarantee that a meaningful formula will not be lower ranked at the end of the process. What the ranking method actually gives is an indication of a smaller set of formulae which might be the interesting ones among a large set of possibilities.

Formula 13 and 14 and the final rule, (16), are three rules expressing the transitivity of $trans/2$ that are equivalent to (9). They were not counted in the annotation in (9) because the symmetry of the predicate $trans/2$ was not a priori assumed in the learning process, but obtained as a result. It is unclear to this author how potentially interesting formula (such as the symmetry axiom) could influence in the ranking process. Finally, (15) seems to be the result of the generalisation of a noisy portion of the data.

In addition, the process of constructing the axioms of equivalence produced a minimal set of examples of ground atomic formulae representing the equivalence between symbols in the given data set. With this minimal set, and the transitivity and symmetry axioms, a complete set of equivalences may be determined.

6 Discussion and future work

To create consists precisely in not making useless combinations and in examining only those which are useful but which are only a small minority. Invention is discernment, choice.[6]

Investigating the possibility of the automatic learning of simple general mathematical rules from computer vision data, this work was confronted with an essential issue in knowledge discovery in general, namely the selection of the most interesting explanations among equally plausible competing solutions. We proposed a simple method for ranking potentially interesting formulae according to a voting criteria. With this method, axioms for transitivity, symmetry and reflexivity were obtained from visual data. Future research will consider how the framework proposed could be used to discover more complex mathematical structures from the observation of incrementally more complex domains.

In this work, multiple ILP processes were used in order to obtain the axioms sought. This practice places the method described in the class of ensemble methods for machine learning. More, specifically, the algorithm introduced in Section 4, has characteristics of *bayesian voting* (as the possible answers are ranked according to a voting criteria over the set of hypotheses) and of *bagging* (since the input sets are manipulated to generate multiple hypotheses. A summary and comparison of these methods can be found in [4]. In particular, [15] presents a similar algorithm (so called PLCG) to the meta-program presented in the present paper. That algorithm learns multiple propositional models, from positive and negative examples, using a standart rule learning algorithm. The main purpose of PLCG is to find the simplest rules from complex data. In contrast the algorithm proposed in this paper is used to find rules from noisy (positive-only) examples.

Future research should include the design of an algorithm for automatically suggesting suitable PROGOL mode declarations and settings withing the meta-program proposed in this paper. This is an essential step towards closed-loop inductive logic programming.

In this paper, transitivity, reflexivity and symmetry axioms were induced from computer vision data obtained from the observation of two simple game playing scenarios. It is worth pointing out that the simplicity of the scenarios does not compromise the importance of these findings. The application of these axioms is not constrained to the scenarios where they were inferred, but they are general rules that are present in a variety of reasoning processes. The further use of these rules on diverse domains is an issue to be taken into account by future investigations. Moreover, the method for ranking anwers presented in this paper is not restricted to the application domain where it was developed.

[6] From Poincaré's lecture *Mathematical Invention* given at the *Société de Psychologie de Paris* in 1908.

7 Conclusion

This paper showed how general mathematical rules can be induced from the noisy data provided by a vision system. Axioms of transitivity, symmetry and reflexivity were obtained by combining the result of multiple processes of an inductive logic programming system (PROGOL) by means of a ranking method that selects the most interesting formulae according to a voting criteria. This method is part of a meta program whose task is to automatically set, run and evaluate multiple PROGOL experiments. The final aim of this research is to close the loop between data acquisition and data generalisation as one step towards the development of fully autonomous systems immersed in the real world.

Acknowledgments
Thanks to the two anonymous reviewers for the constructive comments.

REFERENCES

[1] P. Bryant and S. Squire, ‘Children’s mathematics: Lost and found in space’, in *Spatial Schemas and Abstract Thought*, ed., M. Gattis, 175–200, MIT Press, (2001).

[2] 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).

[3] Simon Colton, Alan Bundy, and Toby Walsh, ‘Automatic identification of mathematical concepts’, in *Proc. 17th International Conf. on Machine Learning*, pp. 183–190. Morgan Kaufmann, San Francisco, CA, (2000).

[4] T. G. Dietterich, ‘Ensemble methods in machine learning’, in *First International Workshop on Multiple Classifier Systems*, eds., J. Kittler and F. Roli, volume 1857 of *Lecture Notes in Computer Science*, pp. 1–15. Springer Verlag, (2000).

[5] A. Fern, J. M. Siskind, and R. Givan, ‘Learning temporal, relational, force-dynamic event definitions from video’, in *Proc. of AAAI/IAAI*, pp. 159–166, (2002).

[6] R. J. Howarth and H. Buxton, ‘Conceptual descriptions from monitoring and watching image sequences’, *Image and vision computing*, **18**, 105–135, (2000).

[7] D. Lenat, ‘Eurisko: A program which learns new heuristics and domain concepts’, *Artificial Intelligence*, **21**, 61–98, (1983).

[8] D. Magee, ‘Tracking multiple vehicles using foreground, background and motion models’, *Image and Vision Computing*, **20**(8), 581–594, (2004).

[9] B. McGonigle and M. Chalmers, ‘Are monkeys logical?’, *Nature*, **267**, 694–696, (1977).

[10] S. Muggleton, ‘Inverse entailment and Progol’, *New Generation Computing, Special issue on Inductive Logic Programming*, **13**(3-4), 245–286, (1995).

[11] S. Muggleton, ‘Learning from positive data’, in *ILP96*, ed., S. Muggleton, volume 1314 of *LNAI*, pp. 358–376. SV, (1996).

[12] S. Muggleton and J. Firth, ‘CProgol4.4: a tutorial introduction.’, in *Relational Data Mining*, eds., S. Dzeroski and N. Lavrac, 160–188, Springer-Verlag, (2001).

[13] H-H. Nagel, ‘Image sequence evaluation: 30 years and still going strong’, in *Proc. of ICPR*, pp. 1149–1158, Barcelona, Spain, (2000).

[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).

[15] G. Widmer, ‘Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries’, *Artificial Intelligence*, **146**, 129–148, (2000).