**Machine learning manuscript No.**
(will be inserted by the editor)

# Ultra-Strong Machine Learning – Comprehensibility of Programs Learned with ILP

**Stephen H. Muggleton · Ute Schmid · Christina Zeller · Alireza Tamaddoni-Nezhad · Tarek Besold**

Received: date / Accepted: date

During the 1980s Michie defined Machine Learning in terms of two orthogonal axes of performance: predictive accuracy and comprehensibility of generated hypotheses. Since predictive accuracy was readily measurable and comprehensibility not so, later definitions in the 1990s, such as that of Mitchell, tended to use a one-dimensional approach to Machine Learning based solely on predictive accuracy, ultimately favouring statistical over symbolic Machine Learning approaches. In this paper we provide a definition of comprehensibility of hypotheses which can be estimated using human participant trials. We present two sets of experiments testing human comprehensibility of logic programs. In the first experiment we test human comprehensibility with and without predicate invention. Results indicate that comprehensibility is affected not only by the complexity of the presented program but also by the existence of anonymous predicate symbols. In the second experiment we directly test whether any state-of-the-art ILP systems are ultra-strong learners in Michie's sense, and select the Metagol system for use in humans trials. Results show that participants were not able to learn the relational concept on their own from a set of examples but they were able to apply the relational definition provided by the ILP system correctly. This implies the existence of a class of relational concepts which are hard to acquire for humans, though easy to understand given an abstract explanation. We believe improved understanding of this class could have potential relevance to contexts involving human learning, teaching and verbal interaction.

S.H. Muggleton and A. Tamaddoni-Nezhad
Department of Computing, Imperial College London, London, UK
E-mail: {s.muggleton,a.tamaddoni-nezhad}@imperial.ac.uk

U. Schmid and C. Zeller
Cognitive Systems Group, University of Bamberg, Germany
E-mail: {ute.schmid,christina.zeller}@uni-bamberg.de

T. R. Besold
Digital Media Lab, University of Bremen, Germany
E-mail: tbesold@uni-bremen.de

## 1 Introduction

In a recent paper [37] the authors introduced an operational definition for comprehensibility of logic programs and conducted human trials to determine how properties of a program affect its ease of comprehension. This paper builds on and extends the earlier work by investigating whether machines can not only learn new concepts but explain those concepts to humans thereby improving human task performance. The definition of comprehensibility allows, for the first time, experimental demonstration of Donald Michie's *Ultra-strong Machine Learning* criterion.

In 1988 Michie [20] provided *weak*, *strong* and *ultra-strong* criteria for Machine Learning. Michie's aim was to provide operational criteria for various qualities of machine learning which include not only predictive performance but also comprehensibility of learned knowledge. His *weak* criterion identifies the case in which the machine learner produces improved predictive performance with increasing amounts of data. The *strong* criterion additionally requires that the learning system must provide its hypotheses in symbolic form. Lastly, the *ultra-strong* criterion extends the strong criterion by requiring the learner to teach the hypothesis to a human, whose performance is consequently increased to a level beyond that produced by the human studying the training data alone.

Most of modern Machine Learning can be viewed as consistent with Michie's *weak* criterion. By contrast, the *strong* criterion plays an ongoing role within Inductive Logic Programming. However, to date no documented attempt has been made, even within Inductive Logic Programming, to apply, or demonstrate Michie's *ultra-strong* criterion to a Machine Learning system. As argued in [37] the major barrier to doing so has been the lack of an operational notion of human comprehension of symbolic concepts.

Within Artificial Intelligence (AI) *comprehensibility* of symbolic knowledge is viewed as one of the defining factors which distinguishes logic-based representations from those employed in statistical machine learning. In [37] the issue is addressed by introducing a definition of comprehensibility which is inspired by "Comprehension Tests", administered to school children. Such a test (see Figure 1) comprises the presentation of a piece of text, followed by questions which probe the child's understanding. Answers to questions in some cases may not be directly stated, but instead inferred from the text. Once the test is scored, the proportion of questions correctly answered provides the measure of a pupil's textual comprehension.

In the same fashion, our operational definition of comprehensibility is based on presentation of a logic program to an experimental participant (see Figure 2), who is given time to study it, after which the score is used to assess their degree of comprehension. The detailed results of such a test can be used to identify factors in

*For many years people believed the cleverest animals after man were chimpanzees. Now, however, there is proof that dolphins may be even cleverer than these big apes.*

**Question:** Which animals do people think may be the cleverest?

Fig. 1: Text comprehension test (Credit: http://englishteststore.net)

```
p(X,Y) :- p1(X,Z), p1(Z,Y).
p1(X,Y) :- father(X,Y).
p1(X,Y) :- mother(X,Y).
father(john,mary). mother(mary,harry).
```

**Question:** p(john,harry)?

Fig. 2: Program comprehension test

the program which affect its comprehensibility both for individuals and for groups of participants. The existence of an experimental methodology for testing comprehensibility has the potential to provide empirical input for improvement of Machine Learning systems for which the generated hypotheses are intended to provide insights.

The paper is arranged as follows. In Section 2 we discuss existing work relevant to the paper. The framework, including relevant definitions and their relationship to experimental hypotheses is described in Section 3. Section 4 describes two experiments involving human participants. The first experiment tests the degree to which predicate invention affects human comprehensibility of concepts. The second experiment tests whether an ILP system can pass Michie's Ultra-Strong Learning criterion. Finally in Section 5 we conclude the paper and discuss further work.

## 2 Related work

This section offers framing information concerning research into comprehensibility and explainability of systems in general, and familiarizes the reader with the core notions motivating our work in particular. We first present a short overview of related lines of investigation in AI and Machine Learning, respectively, before specifically discussing cognitive and computational aspects of predicate invention in the context of the hierarchical structuring of complex concepts, and of induction, abduction, and recursion as important mechanisms for concept formation and representation.

### 2.1 Comprehensibility and Explanation in AI

Studies of the comprehensibility—and relatedly explainability—of computational systems have a long tradition, dating back at least to research into expert and decision support systems in the early 1980s. Clancey [5] questioned the view that expert knowledge can be encoded as a uniform, weakly-structured set of if/then associations (as, e.g., done in the MYCIN system [38]) if the rules shall still be meaningfully modifiable by people other than the original author, or shall be justified and used in teaching (i.e., support active learning). Consequently, efforts were undertaken to improve expert system explanations (see, e.g., [4] and [43]). One resulting line of work addressed the representation formalisms used [9]: among others, smaller structures were taken to be more comprehensible, coherent structures were considered more meaningful, and compositions of familiar concepts were perceived as easier accessible.

In the context of AI testing and evaluation the importance of human comprehensibility of intelligent systems has very recently been emphasised by Forbus [6]. For his

*software social organisms*, comprehensibility of the system's behaviour and outputs is paramount, since only efficient communication enables participation in human society. In general when looking at the original Turing Test [42] and discussions of new and updated versions or substitutes for it, comprehensibility plays a crucial role. While there frequently are suggestions to abandon the Turing Test and focus on more clearly specified tasks in well-defined domains, putting emphasis on making systems and their output comprehensible for humans offers an alternative approach to overcoming limitations of the original test, while still maintaining domain and task generality.

### 2.2 Comprehensibility and Explanation in Machine Learning

In Machine Learning, comprehensibility has for instance been discussed in the context of Argument-Based Machine Learning (ABML) [24], which applies methods from argumentation in combination with a rule-learning approach. Explanations provided by domain experts concerning positive or negative arguments are included in the learning data and serve to enrich selected examples. Although ABML enhances the degree of explanation, it still fails to pass Michie's ultra-strong test since no demonstration of user comprehensibility of learned hypotheses is guaranteed. Moreover, questions and discussions about comprehensibility have also entered the study of classification models [7,18,19]. However, while the need for comprehensibility is emphasized, no definitive test of the kind provided by our definition in Section 3 is offered.

Another approach which engages with aspects of comprehensibility, logical reasoning and to some extent, predicate invention (i.e., the automated introduction of auxiliary predicates)—discussed in more detail in the next subsection due to its role in our first experiment in Section 4—is Explanation-Based Learning (EBL) (e.g. [22]). EBL uses background knowledge in a mainly deductive inference mechanism to "explain" how each training example is an instance of the target concept. The deductive proof of an example yields a specialisation of the given domain theory leading to the generation of a special-purpose sub-theory described in a user-defined operational language. The learning process in EBL is comparable to the use of proof-completion in the context of Meta-Interpretive Learning (MIL) [28,29], with EBL assuming a complete (first-order) domain theory and using deduction (rather than induction or abduction) as key differences. Some EBL systems can discover new features that are not explicit in the training examples but required in the general rule describing the former. For example, Prolog-EBG [14] automatically formulates meaningful constraints required in the rules underlying the training examples. EBL is also realised in the inductive functional programming system Igor where observed program traces are explained as unfolding of an unknown recursive function following some program scheme [16]. Still, existing EBL systems do not pass Michie's ultra-strong test either: again there is no guarantee of user comprehensibility of learned hypotheses. The deductively generated syntactic explanations (i.e. formal proofs) could be far from human comprehensible explanations in a semantic sense (causal, mechanistic, etc.).

### 2.3 Hierarchical structuring of complex concepts through predicate invention

Research into the inner structure of complex concepts found these to be strongly hierarchically organised in a tree-like structure, with more general categories higher in

the hierarchy dominating lower-level categories via IS-A links [31]. This hierarchical structure is presumably acquired by successive abstractions based on sets of instances from lower-level categories. Emulating these generalisation processes, predicate invention has been viewed as an important problem since the early days of ILP (e.g. [26,35,40]), though limited progress has been made in this topic recently [30]. Early approaches were based on the use of $W$-operators within the inverting resolution framework (e.g., [26,35]). However, the completeness of these approaches was never demonstrated, partly because of the lack of a declarative bias to delimit the hypothesis space. Failure to address these issues has, until recently, led to limited progress being made in this important topic and many well-known ILP systems such as ALEPH [39] and FOIL [32] have no predicate invention. In MIL, predicate invention is conducted via construction of substitutions for meta-rules applied by a meta-interpreter. The use of the meta-rules clarifies the declarative bias being employed. New predicate names are introduced as higher-order skolem constants, a finite number of which are added during every iterative deepening of the search.

### 2.4 Logical mechanisms in concept formation and representation

Mechanisms from logical reasoning are found to play crucial roles in human understanding and conceptualization. *Induction* has long been shown to be highly related to concept attainment and information processing [17], and *abduction* also is considered a key mechanism in this context [11]. *Recursion* plays a similarly prominent role during the process of concept acquisition and meaning making, and has been argued to be a key human ability regarding language and understanding in general [10]. Additionally, the capacity to apply recursion is strictly necessary for the representation of infinite concepts (such as, e.g., the concept of an ancestor, or the notion of ordinal numbers). All three mechanisms are also present in MIL. There, induction and abduction, together with predicate invention, are all achieved by way of (higher-order) meta-rules. Owing to the existentially quantified variables in the meta-rules, the resulting first-order theories are strictly logical generalisation of the meta-rules.

Learning recursive programs is a technically difficult task in ILP and is not fully supported by general-purpose ILP systems such as Foil [33], Golem [27] and Progol [25]. Still, different techniques allow for the induction of recursive programs. For instance CRUSTACEAN [1], CLAM [34], TIM [12] and MRI [8] use inverse entailment based on structural analysis. SMART [23] and FILP [3] use top-down induction of arbitrary Horne clauses, including recursive definitions. However, the search remains incomplete due to restrictions regarding the use of (intensional) background knowledge, as well as pruning techniques. FILP can only induce functional predicates and SMART cannot learn mutually inter-depending clauses. Regarding functional and inductive programming, for example the system Igor1 relies on explanation-based generalization over program traces [16]. The successor Igor2 [15] can induce recursive functions which depend on additional, invented functions based on the abduction of input-output pairs for some function call (e.g., in modelling the inductive generalization of rules for domains such as Tower of Hanoi or blocksworld [36]). However, Igor requires the first $k$ examples of a target theory to generalise over a whole class. Esher [2] is a generic and efficient algorithm that interacts with the user via input-output examples, and synthesizes recursive programs implementing intended behaviour. Hence, Esher needs to query an oracle each time a recursive call is encountered to ask for examples.

## 3 Framework

### 3.1 General Setting

We assume sets of constants, predicate symbols and first-order variables are denoted $\mathcal{C}, \mathcal{P}, \mathcal{V}$. We assume definite clause programs to be defined in the usual way. Furthermore we assume a human as possessing background knowledge $B$ expressed as a definite program. We now define the distinction between private and public predicate symbols.

**Definition 1 [Public and private predicate symbols].** We say that a predicate symbol $p \in \mathcal{P}$ found in definite program $P$ is *public* with respect to a human population $S$ in the case that $p$ forms part of the background knowledge of each human $s \in S$. Otherwise $p$ is private.

Now we define Predicate Invention as follows.

**Definition 2 [Predicate Invention].** In the case background knowledge $B$ of an ILP is extended to $B \cup H$, where $H$ is a definite program we call predicate symbol $p \in \mathcal{P}$ an Invention iff $p$ is defined in $H$ but not in $B$.

### 3.2 Comprehensibility

Next we provide our operational definition of comprehensibility.

**Definition 3 [Comprehensibility, $C(S, P)$].** The comprehensibility of a definition (or program) $P$ with respect to a human population $S$ is the mean accuracy with which a human $s$ from population $S$ after brief study and without further sight can use $P$ to classify new material sampled randomly from the definition's domain.

Note that this definition allows us to define comprehensibility in a way which allows its experimental determination given a set of human participants. However, in order to clarify the term "after brief study" we next define the notion of inspection time.

**Definition 4 [Inspection time $T(S, P)$].** The inspection time $T$ of a definition (or program) $P$ with respect to a human population $S$ is the mean time that a human $s$ from $S$ spends studying $P$ before applying $P$ to new material.

Since, in the previous subsection, we assume humans as having background knowledge which is equivalent to a definite program, we next define the idea of humans mapping privately defined predicate symbols to ones found in their own background knowledge.

**Definition 5 [Predicate recognition $R(S, p)$].** Predicate recognition $R$ is the mean proportion of times that a human $s$ from population $S$ gives the correct public name to a predicate symbol $p$ presented as a privately named definition $q$.

For each of these mappings from privately defined predicate symbols to elements from the background knowledge we can now experimentally determine the required naming time.

**Definition 6 [Naming time $N(S, p)$].** For a predicate symbol $p$ presented as a privately named definition $q$ in definite program $P$ the naming time $N$ with respect to a human population $S$ is the mean time that a human $s$ from $S$ spends studying $P$ before giving a public name to $p$.

Lastly we provide a simple definition of the textual complexity of a definite program.

**Definition 7 [Textual complexity, $Sz(P)$].** The textual complexity $Sz$ of a definition of definite program $P$ is the sum of the occurrences of predicate symbols, functions symbols and variables found in $P$.

### 3.3 Ultra-strong Machine Learning

The following definitions extend those above by describing measures for estimating the degree to which humans can be aided by inspection of the output of a symbolic machine learning algorithm. Firstly we define the output of symbolic machine learning.

**Definition 8 [Machine Learned Program, $M(E)$].** The program learned from examples $E$ using machine learning algorithm $M$ which outputs a symbolic hypothesis in the form of a definition of program.

Unaided human learning from training examples can now be defined as follows.

**Definition 9 [Unaided Human Comprehension of Examples, $C(S, E)$].** The comprehensibility of a definition (or program) $P$ with respect to a human population $S$ is the mean accuracy with which a human $s$ from population $S$ after brief study of an example set $E$ of a hidden target definition can classify new material sampled randomly from the target definition's domain.

Lastly we define machine-aided human learning from training examples.

**Definition 10 [Machine-aided Human Comprehension of Examples, $C(S, M(E))$].** The machine-aided comprehensibility of a definition (or program) $P$ with respect to a human population $S$ is the mean accuracy with which a human $s$ from population $S$ after brief study of a program $M(E)$, learned by a symbolic Machine Learning algorithm $M$ from examples $E$, can classify new material sampled randomly from the target definition's domain.

### 3.4 Experimental Hypotheses

We are now in a position to define and explain the motivations for the experimental hypotheses to be tested in Section 4. Below $C(S, P)$, $T(S, P)$, $R(S, p)$, $N(S, p)$, $Sz(P)$, $C(S, E)$, $C(S, M(E))$ are denoted by $C, T, R$, $N$, $Sz$, $C_{\mathrm{H}}$, $C_{\mathrm{HM}}$ respectively. Note that $C_{\mathrm{H}}$ and $C_{\mathrm{HM}}$ indicate respectively Comprehension of a Human given data as opposed to Comprehension of a Human given data and a Machine Learning system.

**Hypothesis H1,** $C \propto \frac{1}{T}$. This hypothesis relates to the idea of using inspection time as a proxy for incomprehension. That is, we might expect that humans take a long time to commit to an answer in the case they find the program hard to understand. As a proxy, inspection time is easier to measure than comprehension.

**Hypothesis H2,** $C \propto R$. This hypothesis is related to the idea that humans understand private predicate symbols, such as *p1/2*, generated during predicate invention, by mapping them to public ones in their own background knowledge.

Table 1: Mapping defined properties from this section and independent variables in the experiments.

| Defined property | Experimental variable |
|---|---|
| Comprehensibility $C$ | Score |
| Inspection time $T$ | Time |
| hline Recognition $R$ | CorrectNaming |
| Naming Time $N$ | NamingTime |

**Hypothesis H3,** $C \propto \frac{1}{Sz}$. This hypothesis is motivated by the idea that a key property of predicate invention is its ability to compress a description by introducing new predicates which are used multiply within the definition. We are interested in whether the resultant compression of the description leads to increased comprehensibility.

**Hypothesis H4,** $R \propto \frac{1}{N}$. This hypothesis relates to the idea that if humans take a long time to recognise and publicly name a privately named predicate they are unlikely to correctly identify it. Analogous to H1, this allows naming time to be used as a proxy for recognition of an invented predicate.

**Hypothesis H5,** $C_{\text{withM}} \geq C_{\text{withoutM}}$. This hypothesis relates to the idea of Ultra-Strong Machine Learning. That is, we might expect that humans perform better on unseen data after having seen a symbolic machine learned definition compared with simply inspecting the training data.

In the next section we describe experiments which test these four hypotheses. Table 1 shows the mapping between the measurable properties defined in this section and the independent variables used in the experiments.

## 4 Experiments

To investigate the hypotheses concerning comprehensibility of concept description in a logical representation we conducted two experiments with human participants. In a first experiment, our main interest was whether making a concept definition more compact by introducing additional predicates in the body of Prolog rules positively impacts comprehensibility and we explored several aspects related with the use of invented predicates. In a second experiment, focus was on ultra-strong learning. To examine whether a machine learned hypothesis is operationally effective, we compared performance of participants when they had to classify unseen objects of a domain on their own in contrast to being offered explicit classification rules learned with an ILP approach. In the following, we will present the details of the experiments, introducing the materials, describing the experimental methods, and giving the results.

### 4.1 Experiment 1 - Comprehensibility and Predicate Invention

We gave a closer look on whether classification rules using additional predicates are helpful per se or whether their helpfulness is dependent on the ability of a person

to assign a specific meaning to the predicate. Furthermore, we were interested in possible interactions between predicate use and complexity of rules. For this reason, the first experiment involved quite a number of variations in material and procedure which are introduced in the following.

### *4.1.1 Materials*

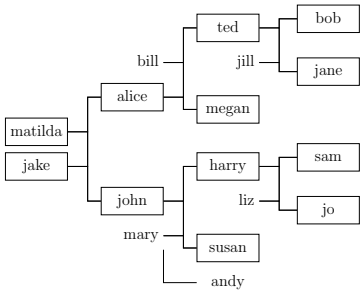
Material construction is based on the well-known family tree examples used to teach Prolog [41] and also used in the context of ILP [29]. Based on the *grandparent/2* predicate, three additional problems were defined: *grandfather/2* which is more specific than *grandparent/2*, *greatgrandparent/2* which needs the double amount of rules if defined without an additional (invented) predicate, that is, which has a high textual complexity, and the recursive predicate *ancestor/2* which has small textual but high cognitive complexity [13]. Instead of these meaningful names, target predicates are called *p/2*. Given facts are identical to the family tree presented in [29].In the rule bodies, either public names (*mother*, *father*)—that is, names which relate to the well-known semantics of family relations—or private names (*q1/2*, *q2/2*) were used. Furthermore, programs were either presented with or without the inclusion of an additional (invented) predicate for *parent/2* which was named *p1/2*. The tree and the predicate definitions for the public name space are given in Figure 3.

- What is the result of `p(bill,bob)`?
  □ true □ false □ don't know
- What is the result of `p(jake,harry)`?
  □ true □ false □ don't know
- What is the result of `p(bob,bill)`?
  □ true □ false □ don't know
- What is the result of `p(mary,jo)`?
  □ true □ false □ don't know
- What is the result of `p(john,sam)`?
  □ true □ false □ don't know
- What is the result of `p(X,bob)`?
  □ false □ $X$ = bill □ $X$ = alice
  □ $X$ = bill; alice □ don't know
- What is the result of `p(john,X)`?
  □ false □ $X$ = sam □ $X$ = jo
  □ $X$ = sam; jo □ don't know

Fig. 4: Questions for the *grandparent/2* problem with public names.

In Section 3 we defined comprehensibility of a program as the accuracy with which a human can classify new material sampled from the domain. To assess comprehensibility, we defined seven questions for each of the four predicates (see Fig. 4). For five questions, it has to be determined whether a relation for two given objects is true. For two further questions, it has to be determined for which variable bindings the relation can be fulfilled. In addition, an open question was included, where a meaningful name had to be given to predicate *p/2* for each of the four problems and—if applicable—also to the additional predicate *p1/2*.

To evaluate the material, we ran a pilot study (March 2016) at Imperial College London with 16 students of computer science with a strong background in programming, Prolog, and logic. The pilot study was conducted as a paper-and-pencil experiment where for each problem first the seven questions had to be answered and afterwards a meaningful name had to be given to the program. 13 out of the 16 students solved all questions correctly and most students were able to give the correct public names to all of the programs, regardless whether they had to work with the public or with the private names. Participants needed about 20 minutes for the four problems. Thus, the instructions and the material are understandable and coherent. A very interesting outcome of the study was that about a third of the students made notes on the questionnaires. Some of the notes showed that students first named the target predicates and the invented predicate (see Fig. 5) and then




**Background Knowledge (Observations):**

```
father(jake,alice).     mother(matilda,alice).
father(jake,john).      mother(matilda,john).
father(bill,ted).       mother(alice,ted).
father(bill,megan).     mother(alice,megan).
father(john,harry).     mother(mary,harry).
father(john,susan).     mother(mary,susan).
                        mother(mary,andy).
father(ted,bob).        mother(jill,bob).
father(ted,jane).       mother(jill,jane).
father(harry,sam).      mother(liz,sam).
father(harry,jo).       mother(liz,jo).
```

**Target Concepts (Rules):**

```
% grandfather without invented pred.
p(X,Y) :- father(X,Z), father(Z,Y).
p(X,Y) :- father(X,Z), mother(Z,Y).

% grandfather with invented predicate
p(X,Y) :- p1(X,Z), father(Z,Y).
p1(X,Y) :- father(X,Y).
p1(X,Y) :- mother(X,Y).

% grandparent without invented pred.
p(X,Y) :- father(X,Z), father(Z,Y).
p(X,Y) :- father(X,Z), mother(Z,Y).
p(X,Y) :- mother(X,Z), mother(Z,Y).
p(X,Y) :- mother(X,Z), father(Z,Y).

% grandparent with invented predicate
p(X,Y) :- p1(X,Z), p1(Z,Y).
p1(X,Y) :- father(X,Y).
p1(X,Y) :- mother(X,Y).

% ancestor without invented predicate
p(X,Y) :- father(X,Y).
p(X,Y) :- mother(X,Y).
p(X,Y) :- father(X,Z), p(Z,Y).
p(X,Y) :- mother(X,Z), p(Z,Y).

% greatgrandparent without invented predicate
p(X,Y) :- father(X,U), father(U,Z), father(Z,Y).
p(X,Y) :- father(X,U), father(U,Z), mother(Z,Y).
p(X,Y) :- father(X,U), mother(U,Z), father(Z,Y).
p(X,Y) :- father(X,U), mother(U,Z), mother(Z,Y).
p(X,Y) :- mother(X,U), father(U,Z), mother(Z,Y).
p(X,Y) :- mother(X,U), father(U,Z), father(Z,Y).
p(X,Y) :- mother(X,U), mother(U,Z), mother(Z,Y).
p(X,Y) :- mother(X,U), mother(U,Z), father(Z,Y).

% greatgrandparent with invented predicate
p(X,Y) :- p1(X,U), p1(U,Z), p1(Z,Y).
p1(X,Y) :- father(X,Y).
p1(X,Y) :- mother(X,Y).

% ancestor with invented predicate
p(X,Y) :- p1(X,Y).
p(X,Y) :- p1(X,Z), p(Z,Y).
p1(X,Y) :- father(X,Y).
p1(X,Y) :- mother(X,Y).
```

Fig. 3: Public tree and the Prolog programs for *grandfather/2*, *grandparent/2*, *greatgrandparent/2*, and *ancestor/2* with and without use of an additional (invented) predicate *parent/2*. In the corresponding programs for the private name space, *father/2* is replaced by *q1/2*, *mother/2* is replaced by *q2/2*, and given names are replaced by two letter strings as shown in *Observations* in Figure 8.

```
p(X,Y)  :- p1(X,U), p1(U,Z), p1(Z,Y).
p1(X,Y) :- father(X,Y).
p1(X,Y) :- mother(X,Y).
```

parent

grandgrand parent

Fig. 5: Example of student giving meaningful names to predicate symbols.

answered the questions. That is, students gave a meaningful name without being instructed to do so and one can assume that they used this strategy because it made answering the questions easier.

### *4.1.2 Method*

*Variables.* To assess the influence of meaningful names and of predicate invention on comprehensibility, we introduced the following three independent variables:

NameSpace: The name space in which context the problems is presented is either **public** or **private**.

PredicateInvention: The problems are given either **with** or **without** an additional (invented) predicate *p1/2* which represents the *parent/2* relation.

NamingInstruction: The open question to give a meaningful name to predicate *p/2* and—if applicable— also to the additional predicate *p1/2* is either given **before** or **after** the seven questions given in Figure 4 had to be answered.

The variation of the independent variables results in a $2 \times 2 \times 2$ factor design which was realised between-participants for factors NameSpace and NamingInstruction and within-participants for factor PredicateInvention. Problem presentation with PredicateInvention was either given for the first and the third or the second and the fourth problem.

The **textual complexity** varies over problems and in dependence of the introduction of the additional predicate *p1/2*. The textually most complex program is *greatgrandparent/2* without the use of *p1/2*. The least complex program is *grandfather/2* without the use of *p1/2* as can be seen in Figure 3.

The following dependent variables were assessed:

Score: For each problem, the score is calculated as the sum of correctly answered questions (see Fig. 4). That is, score has minimal value 0 and maximal value 7 for each problem.

Time: The time to inspect a problem is measured from presenting the problem until answering the seven questions.

CorrectNaming: The correctness of the given public name for a predicate definition *p/2* was judged by two raters. In addition, it was discriminated between clearly incorrect answers and responses where participants wrote nothing or stated that they do not know the correct meaning.

NamingTime: The time for naming is measured from presenting the question until indication that the question is answered by going to the next page. For condition PredicateInvention/with both *p/2* and *p1/2* had to be named.

*Empirical Hypotheses.* Given the independent and dependent variables, hypotheses can now be formulated with respect to these variables:

H1: Score is inverse proportional to Time, that is, participants who comprehend a program, give more correct answers in less time than such participants who do not comprehend the program.

H2: CorrectNaming is proportional to Score, that is, participants who can give the intended public—that is, meaningful—name to a program have higher scores than participants who do not get the meaning of the program.

H3: Score is inverse proportional to textual complexity, that is, for problem *greatgrandparent/2* the differences of score should be greatest between the PredicateInvention/with and PredicateInvention/without condition because here the difference in textual complexity is highest.

H4: CorrectNaming is inverse proportional to NamingTime, that is, if participants need a long time to come up with a meaningful name for a program, they probably will get it wrong.

*Participants and Procedure.* The experiment was conducted in April 2016 with cognitive science students of the University of Osnabrück. All students had passed at least one previous one-semester course on Prolog programming and all have a background in logic. That is, their background in Prolog is less strong than for the Imperial College sample but they are no novices. From the originally 87 participants,

three did not finish the experiment and six students were excluded because they answered "don't know" for more than 50% of the questions. All analyses were done with the remaining 78 participants (43 male, 35 female; *mean age* = 23.55 years, $sd = 2.47$).[1]

The experiment was realised with the `soscisurvey.de` system and was conducted online during class. After a general introduction, students worked through an example problem ("sibling") to get acquainted with the domain and with the types of questions they needed to answer. Afterwards, the four test problems were presented in one of the experimental conditions. For each problem, on the first page the facts and the tree and the predicate definition was presented. On the next page, this information was given again together with the first question or the naming instruction. If the "next"-button was pressed, it was not possible to go back to a previous page.

Working through the problems was self-paced. The four problems were presented in the sequence *grandfather/2*, *grandparent/2*, *greatgrandparent/2*, *ancestor/2* for all participants. That is, we cannot control for sequence effects, such as performance gain due to getting acquainted with the style of the problems and questions or performance loss due to decrease in motivation or fatigue. However, since problem type is not used as an experimental condition, possible sequence effects do not affect statistical analyses of the effects of the independent variables introduced above.

#### *4.1.3 Results*

*Scores and Times.* When considering time for question answering and naming together, participants needed about 5 minutes for the first problem and got faster over the problems. One reason for this speed-up effect might be, that participants needed less time to inspect the tree or the facts for later problems. There is no speed-accuracy trade-off, that is, there is no systematic relation between (low) number of correct answers and (low) solution time for question answering. In the following, time is given in seconds and for statistical analyses time was logarithmically transformed.
*Giving meaningful names.* In the public name condition, the names the participants gave to the programs were typically the standard names, sometimes their inverse, such as "grandchildren", "child of child", or "parent of parent" for the *grandparent/2* problem. In the condition with private names, the standard names describing family relations were also used by most participants, however, some participants gave more abstract descriptions, such as "X and Y are connected via an internode" for *grandparent/2*. Among the incorrect answers for the *grandparent/2* problem often were over-specific interpretations such as "grandson" or "grandfather". The same was the case for *greatgrandparent/2* with incorrect answers such as "greatgrandson". Some participants restricted the description to the given tree, for example, "parent of parent with 2 children" for *grandparent/2*. Incorrect answers for the *ancestor/2* problem typically were overly general, such as "related".
*Impact of NameSpace, PredicateInvention, and NamingInstruction on Score and Time.* An overview of the impact of all factors on score is given in Figure 6. There it can be seen that NameSpace/public results in higher scores for all four problems. The effects of PredicateInvention and NamingInstruction are less obvious. It is not the case that having to think about the meaning of a predicate before question answering has a

[1] A comprehensive description of all analyses and results can be found at `http://www.cogsys.wiai.uni-bamberg.de/publications/comprAnalysesDoc.pdf`.

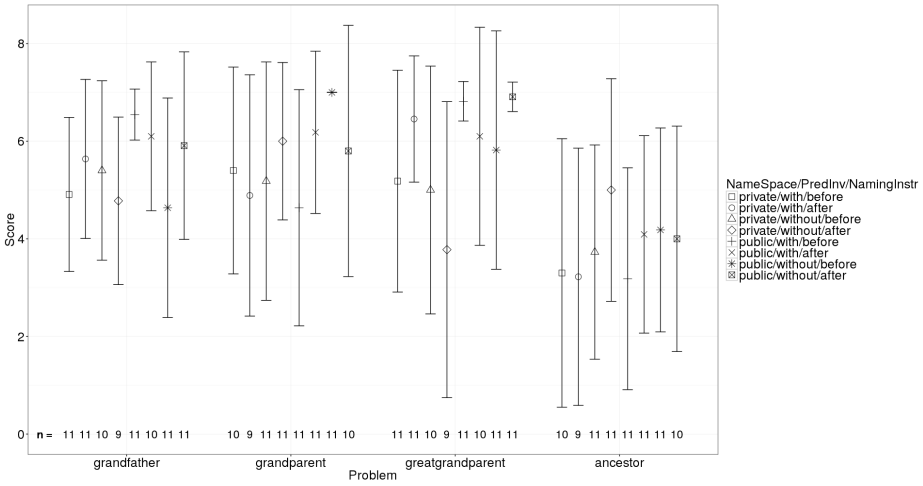



Fig. 6: Scores distributed over NameSpace, PredicateInvention, and NamingInstruction (arithmetic means and standard deviations are given; for significant differences, see text).

general positive effect on Score. PredicateInvention is helpful for some problems, for others not. We will give a closer look on the effect of PredicateInvention for the textually most complex problem *greatgrandparent/2* below (H3). Statistical analyses were done with general linear models with NameSpace, PredicateInvention, and NamingInstruction as predictor variables and Score as criterion variable. Predictor variables were dummy coded as contrasts. The effect of NameSpace/public is significant for *grandfather/2* ($b = 1.55$, $p = 0.03$) and marginally significant for *greatgrandparent/2* ($b = 1.12$, $p = 0.069$). In addition, for *grandfather/2* the interaction of NameSpace and PredicateInvention is significant ($b = -2.52$, $p = 0.017$).

*Inverse proportional relation between Score and Time (H1).* There is a significant negative Pearsons product-moment correlation between Time and Score over all problems ($r = -.38$, $p \leq 0.001$).

*Effect of CorrectNaming on Score (H2).* To assess the impact of being able to give a meaningful name to a problem (CorrectNaming) on comprehensibility (Score), answers were classified as "correct", "incorrect" and "no answer" which covers answers where participants either did not answer or explicitly stated that they do not know the answer. Participants who were able to give meaningful names to the programs answered significantly more questions correctly. Statistical analyses were again performed with general linear models with dummy coding (contrast) for the predictor variable CorrectNaming. The results are given in Table 2.

*Impact of textual complexity on the effect of PredicateInvention on Score (H3).* For the *greatgrandparent/2* problem, there is a marginally significant effect of PredicateInvention for NameSpace/private and NamingInstruction/after with a higher score for the PredicateInvention/with condition ($b = -1.59$, $p = 0.09$).

Table 2: Means and standard deviations of Score in dependence of CorrectNaming, where "no answer" covers answers where participants either did not answer or explicitly stated that they do not know the answer. Results for linear models are given as b-estimates and p-values for the contrast between correct and incorrect naming.

| | Correct | Incorrect | No answer | Test |
|---|---|---|---|---|
| Grandfather | n = 28 | n = 46 | n = 4 | |
| Score | Mean 6.68 (sd = 0.61) | 5.15 (1.81) | 4.75 (1.71) | -1.53, $p < 0.001$ |
| Grandparent | 50 | 23 | 5 | |
| Score | 6.56 (1.23) | 5.04 (2.12) | 3.4 (1.82) | -1.52, $p < 0.001$ |
| Greatgrandparent | 54 | 18 | 6 | |
| Score | 6.76 (0.66) | 5.78 (1.66) | 3 (1.67) | -1, $p < 0.001$ |
| Ancestor | 32 | 39 | 7 | |
| Score | 5.75 (1.44) | 3.08 (1.8) | 2.86 (1.57) | -2.67, $p < 0.001$ |

*Relation of CorrectNaming and NamingTime (H4).* Participants who give a correct meaningful name to a problem do need less time to do so than participants who end up giving an incorrect name for all problems except *ancestor/2*. This relation is given in Figure 7 accumulated over all factors per problem. Statistical analyses were done separately for conditions PredicateInvention/with and PredicateInvention/without because in the first case two names—for target predicate *p/2* and for the additional predicate *p1/2*—had to be given. Differences between correct and incorrect are significant for *grandfather/2* in the condition PredicateInvention/without ($b = 0.31$, $p = 0.007$) and marginally significant for *grandparent/2* in the condition PredicateInvention/with ($b = 0.2$, $p = 0.084$). For *ancestor/2* in the condition PredicateInvention/with there is a significant difference between correct naming and "no answer" ($b = -0.49$, $p = 0.039$).

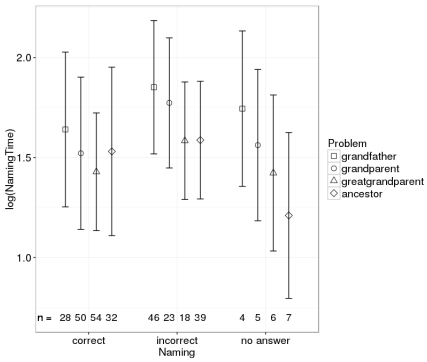



Fig. 7: Relation between time needed for giving a meaningful name and correctness of naming, where "no answer" covers answers where participants either did not answer or explicitly stated that they do not know the answer (averaged over PredicateInvention with/without).

## 4.2 Experiment 2 - Ultra-Strong Learning

After exploring the impact of predicate invention on comprehensibility, we conducted a further experiment to test the hypothesis that ILP-learned relational concepts can support humans making sense from observations in complex relational domains. To follow Michie's characterisation of ultra-strong learning, we aim to show that ILP learned classification rules can result in operational effectiveness for humans. That is, given a set of observations in a domain, we need to show that humans are not

able to induce a classification rule but an ILP system can and additionally that the ILP learned rules can be understood by humans and successfully applied to new observations.[2]

### *4.2.1 Material*

We focused on the *grandparent/2* problem investigated in the first experiment (see Fig. 3) and constructed an isomorphic fictitious chemistry domain shown in Figure 8. The **Observations** correspond to the ones of the private version of the family tree used in Experiment 1. The **Test Results** are four positive and negative examples which can be used as training examples for an ILP system such as Metagol. The **Rules** are the classification rules induced by Metagol. For one group of participants these rules initially are not given and the participants were required to induce them by themselves from the same test results. To assess comprehensibility, an isomorphic questionnaire to the one used in the first experiment has been used (see Fig. 4).

Imagine you work in a chemical laboratory. Over the last days you tested several substances (named `aa`, `ab`, and so on) for two reactions `q1(X,Y)` and `q2(X,Y)`. For example, `q1(aa,ab)` means that `aa` is a substrate and `ab` is a product of reaction `q1`. A list of all observations is given below.

**Observations:**

```
q1(ab,ac).   q2(aa,ac).
q1(ab,ae).   q2(aa,ae).
q1(ad,ag).   q2(ac,ag).
q1(ad,ai).   q2(ac,ai).
q1(ae,aj).   q2(af,aj).
q1(ae,al).   q2(af,al).
q1(ag,an).   q2(af,am).
q1(ag,ao).   q2(ah,an).
q1(aj,ap).   q2(ah,ao).
q1(aj,aq).   q2(ak,ap).
             q2(ak,aq).
```

Today you tested whether a pair of substances are related to an `exothermic` reaction (a chemical reaction that releases energy by light or heat). For example, `exothermic(ac,an)` means that `ac` and `an` are, respectively, substrate and product of a (chain of) reaction(s) which is exothermic. You observed the following test results:

**Test Results:**

```
exothermic(ac,an).   not exothermic(aa,ab).
exothermic(aa,al).   not exothermic(ad,ai).
exothermic(ab,ag).   not exothermic(ab,aq).
exothermic(ae,ap).   not exothermic(aj,ap).
exothermic(aa,ag).   not exothermic(an,ac).
```

You have a new computer program which can support you in finding rules to characterize substances. When you presented your observations to the program, it returned the following rules:

**Rules:**

```
exothermic(X,Y) :- q1(X,Z), q1(Z,Y).
exothermic(X,Y) :- q1(X,Z), q2(Z,Y).
exothermic(X,Y) :- q2(X,Z), q2(Z,Y).
exothermic(X,Y) :- q2(X,Z), q1(Z,Y).
```

Fig. 8: Fictitious chemistry domain.

### *4.2.2 Method*

*Design and Variables.* To control for possible effects of previous involvement with the problem, we used a pre-test post-test design as shown in Table 3. In a between-participants design, participants of one group were asked to induce a classification rule by themselves from examples (Rule Acquisition and Application, **RAA**), another got immediately presented with the classification rules analogous to the first experiment (Rule Application, **RA**). Comprehensibility scores (dependent variable Score) were assessed for both groups after the classification rules were presented (O2). For condition RAA, comprehensibility additionally has been assessed after rule induction O1). For this group, participants additionally were asked to define the classification rule in Prolog or natural language (dependent variable Rule Generation).

[2] A detailed description of the material and the results is given in `http://www.cogsys.wiai.uni-bamberg.de/publications/UltraStrExpAnalyses.pdf`.

Table 3: Experimental design with conditions Rule Acquisition and Application (RAA) and Rule Application (RA), R = randomised, O1 and O2 are measurements of comprehensibility, X is the presentation of the ILP-learned rule, D is an unrelated distractor task.

| | | | | |
|---|---|---|---|---|
| RAA | R | O1 | X | O2 |
| RA | R | D | X | O2 |

*Empirical Hypothesis.* We assume that for this unfamiliar chemistry domain, human problem solvers are not able to come up with the correct classification rules. However, an ILP approach such as Metagol can generate rules which are comprehensible to humans. Consequently, our operational hypothesis is:

H5: Score after simply inspecting training data (O1) is significantly lower than after having seen a symbolic machine learned definition (O2) regardless whether participants had first to try to induce the rules themselves or not (no difference of O2 scores between groups RAA and RA).

That is, measurement O1 addresses unaided human comprehension of examples and O2 addresses machine-aided human comprehension as introduced in Section 3. Additionally, we assume that participants are not able to formulate correct classification rules in Prolog or natural language.

*Participants and Procedure.* The experiment has been conducted in December 2016 at University of Osnabrück. Participants were 43 undergraduate students of cognitive science (20 female, 23 male, *mean age* = 22.12 years, *sd* = 2.51) with a good background in Prolog and in logic but no background in inductive logic programming.

The participants were randomly assigned to one of the following two conditions: Rule Acquisition and Application (RAA, $n = 22$, 12 male, 10 female, *mean age* = 22.09 years, *sd* = 2.56) or Rule Application (RA, $n = 21$, 11 male, 10 female, *mean age* = 22.14 years, *sd* = 2.52). For both conditions, participants had to solve comprehensibility problems (O1 only for RAA, O2 for both RAA and RA, see Figure 4). The participants were tested in class as part of a lecture. The experiment again was realised with `soscisurvey.de` and participants used their own computers.

After a general introduction, for the RA condition an unrelated task (D in Table 3) was presented to control for length of time and mental effort. Both experimental groups first received an example problem (a blocks-world domain concept) to get acquainted with the experimental task. Then, the participants of the RAA condition were presented with the examples of the chemical domain—but *not* with the four rules giving the relational concept. Instead, they were asked to describe the concept either in natural language or as Prolog rules. Next, they had to solve the comprehensibility test (O1 in Table 3). From there on, the procedure for the RAA and RA group was identical: Participants were presented with the ILP-learned rules which characterise the searched-for concept and had to solve the second comprehensibility test (O2 in Table 3) which consists of tasks isomorphic to the first test. Afterwards, demographic data were obtained. The experiment took about 20 minutes.

*4.2.3 Results*

*Rule Generation.* The 22 participants of the RAA condition had to formulate the rules which characterise the target concept *exothermic*. 13 participants tried to formulate the rules. Of these, 11 wrote Prolog code, 2 gave a natural language description. Only one participant gave the correct rules (in Prolog). All other participants gave erroneous rules, often either too specific (not covering all of the given positive examples) or too general (covering some negative examples). Some example solutions are given in Figure 9. The results support our assumption that the fictitious chemistry domain is too complex for humans to be able to acquire the correct relational concept from examples.

Participant 327: too specific

```
exothermic if the substrate appears as a
substrate and the product appears as a
product in the same type of q.  if they
 are both substrates or both products,
or if they appear like that but in
 different q's, then it's not exothermic
```

Participant 314: too specific

```
exothermic(X,Y) :- q2(X,Z), q1(Z,Y).
```

Participant 295: too general

```
not_exothermic(X,Y) :- q2(X,Z), q1(Y,Z).
not_exothermic(X,Y) :- q1(X,Y).
exothermic(X,Y) :- not(not_exothermic(X,Y)).
```

Fig. 9: Examples for erroneous rules.

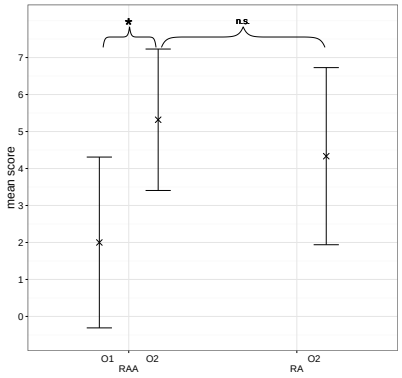



Fig. 10: Mean comprehensibility scores for rule acquisition and application (RAA) vs. rule application (RA) condition (details see text).

*Scores.* To evaluate the comprehensibility scores, we excluded the one participant who could formulate the correct relational concept. This participant also had maximum score values for both comprehensibility tests. Participants of the RAA condition had very low comprehensibility scores at the first testing time ($n = 21$, $mean = 1.76$, $sd = 2.07$). However, their scores significantly improved for the second testing time (t-test for dependent samples, $t(21) = 7.63$, $p < 0.001$), that is, after they were presented with the ILP-learned rules ($n = 21$, $mean = 5.24$, $sd = 1.92$). Participants of the RA condition who immediately were presented the ILP-learned rules performed slightly worse ($n = 21$, $mean = 4.33$, $sd = 2.39$), but not significantly so (Wilcoxon rank sum test with continuity correction, $W = 267$, $p = 0.119$). The results are summarised in Figure 10. They clearly support our hypothesis that white box machine learning approaches such as ILP can support humans to identify relational concepts in complex domains.

## 4.3 Discussion

Our findings show that presenting programs in relation to a public name space facilitates comprehension. Contrary to our expectations, being instructed to first think about a meaningful name for a program before answering questions in general does not facilitate generation of answers. We would have expected that having a (denotational) semantic interpretation for a predicate supports working on classification and variable

Table 4: Hypotheses concerning comprehensibility, meaningful names, and predicate invention. Conf. stands for Confrimation, C means confirmed, P partially confirmed.

| Hypothesis | | Conf. |
|---|---|---|
| H1 | Comprehensibility manifests itself in high scores and fast solution times. | C |
| H2 | Comprehensibility means to be able to give a meaningful name to a program. | C |
| H3 | Predicate invention helps comprehensibility if it reduces textual complexity of the program. | P |
| H4 | If coming up with a meaningful name needs a long time, it will probably be the false concept. | P |
| H5 | ILP can generate classification rules which fulfil the ultra-strong learning criterion. | C |

bindings of new material from a given domain because mental evaluation of a program can be—at least partially—avoided. Furthermore, as expected, the use of additional (invented) predicates does not facilitate program comprehension in general but only under specific conditions which are discussed below (H3).

Results concerning our hypotheses are summarised in Table 4. Hypothesis H1 is confirmed by our empirical data: if a person comprehends a program, she or he can come up with correct answers in short time. Hypothesis H2 is also confirmed: participants who can give a meaningful name to a program give more correct answers than participants who give incorrect answers or state that they do not know the answer. In addition, participants who give a correct name give answers faster. As hypothesis H3 we assumed that predicate invention supports comprehensibility if it reduces the textual complexity of a program. For the four problems we investigated, the reduction in complexity is greatest for *greatgrandparent/2*. Here we get a partial confirmation: predicate invention results in more correct answers for the private name space and if the instruction for naming was given after question answering. This experimental condition is the most challenging, because comprehensibility is not supported by public names and because participants were not encouraged to think about the meaning of the presented predicate before they had to answer questions about it.

Finally, we assumed that persons who have problems to come up with a meaningful name for a predicate spend a longer amount of time to come up with an (incorrect or no) answer (H4). Results show that this is the case—with the exception of the *ancestor/2* problem. However, the differences are only significant under specific conditions. The observation that long answering time can indicate a problem with comprehensibility could be exploited for the design of the interaction of a person with an ILP system: if a person does not come up quickly with a name for a predicate, the system could offer examples of the predicates behaviour. For example, for the *ancestor/2* problem, pairs for which this predicate is true could be highlighted in the given tree.

In a second experiment we demonstrated that there are complex domains where humans are not able to induce the underlying relational concept but that they can correctly apply an explicit, rule-based representation of the relational concept (H5). All together, our empirical results indicate that inductive logic programming can be used to help humans to make sense of complex data.

## 5 Conclusions and Further Work

In this paper we provide an operational definition of comprehensibility of a logic program (Section 3) and use this within within two experiments.

In the first experiment we identify factors which affect comprehension. These factors include the time required to inspect the program, the accuracy with which a partcipant can recognise a predicate to one already known and the textual complexity of the program. As expected, the four problems presented in the first differ with respect to comprehensibility. The problem most participants had difficulty with is the recursive *ancestor/2*. For this problem less than half of the participants (32) gave the correct meaningful name and for this problem participants have the lowest scores. However, since this problem was positioned last in the sequence, the results might also be due to loss of motivation or exhaustion. Interestingly, *ancestor/2* is also the only one of the four problems where participants reached the highest score in the private naming condition without predicate invention. The kinship predicates presented to human participants in the first experiment are all ones which could be expected to be equivalent to ones already known to the participants. In the second experiment we studied te effects of human users being presented with definitions of predicates which are novel to the user.

The second experiment tested whether humans can improve their performance on unseen data when shown a program genearated by a symbolic machine learning system compared with their predictions based only on unaided study of the training data. The experimental support of hypothesis $H5$ represents to a world first demonstration of the existence of Ultra-strong Machine Learning in the sense introduced by Michie [21]. However, for further work we note that H5 will only hold in the case that the Machine Learning is effective.

In our opinion, explanation is a signature not of intelligence, but of being human. It is possible to imagine the neural network of a snail being trained to play expert level Go, but such a snail will never be able to explain to itself or other snails how to play well. Human explanation represents a novel evolutionary development which allows signalling between individuals using abstract languages. Human science can be viewed as a collection of powerful and predictive explanations of the world. Black box learning is an acceptable approach for forming intelligent individual agents, but not for forming intelligent societies which incorporate human beings. This paper demonstrates that good explanations learned by machines can have the power to mibue individuals with improved performance which would fail to achieve even when provided with large numbers of examples.

In closing we believe the operational definition of comprehensibility has enormous potential to both clarify one of the central concepts of AI research as well as to provide a bridge to the study of factors affecting the design of AI systems which improve human understanding.

## References

1. D. W. Aha, S. Lapointe, C. X. Ling, and S. Matwin. Inverting implication with small training sets. In *European Conference on Machine Learning*, pages 29–48. Springer, 1994.
2. A. Albarghouthi, S. Gulwani, and Z. Kincaid. Recursive program synthesis. In *International Conference on Computer Aided Verification*, pages 934–950. Springer, 2013.

3. F. Bergadano and D. Gunetti. *Inductive Logic Programming: from machine learning to software engineering*. MIT Press, 1996.
4. B. Chandrasekaran, M. C. Tanner, and J. R. Josephson. Explaining control strategies in problem solving. *IEEE expert*, 4(1):9–15, 1989.
5. W. J. Clancey. The epistemology of a rule-based expert system: A framework for explanation. *Artificial intelligence*, 20(3):215–251, 1983.
6. K. D. Forbus. Software social organisms: Implications for measuring AI progress. *AI Magazine*, 37(1), 2016.
7. A. A. Freitas. Comprehensible classification models: A position paper. *SIGKDD Explor. Newsl.*, 15(1):1–10, March 2014.
8. M. Furusawa, N. Inuzuka, H. Seki, and H. Itoh. Induction of logic programs with more than one recursive clause by analyzing saturations. In *International Conference on Inductive Logic Programming*, pages 165–172. Springer, 1997.
9. B. R. Gaines. Transforming rules and trees into comprehensible knowledge structures. *Advances in Knowledge discovery and Data mining*, pages 205–226, 1996.
10. M. D. Hauser, N. Chomsky, and W. T. Fitch. The faculty of language: What is it, who has it, and how did it evolve? *science*, 298(5598):1569–1579, 2002.
11. J. R. Hobbs. Abduction in natural language understanding. *Handbook of pragmatics*, pages 724–741, 2004.
12. P. Idestam-Almquist. Efficient induction of recursive definitions by structural analysis of saturations. *Advances in inductive logic programming*, pages 192–205, 1996.
13. H. Kahney. What do novice programmers know about recursion? In E. Soloway and J. C. Spohrer, editors, *Studying the Novice Programmer*, pages 209–228. Lawrence Erlbaum, 1989.
14. S.T. Kedar-Cabelli and L.T. McCarty. Explanation-based generalization as resolution theorem proving. In P. Langley, editor, *Proceedings of the Fourth International Workshop on Machine Learning*, pages 383–389, Los Altos, 1987. Morgan Kaufmann.
15. E. Kitzelmann. Analytical inductive functional programming. In *International Symposium on Logic-Based Program Synthesis and Transformation*, pages 87–102. Springer, 2008.
16. E. Kitzelmann and U. Schmid. Inductive synthesis of functional programs: An explanation based generalization approach. *Journal of Machine Learning Research*, 7(Feb):429–454, 2006.
17. E. A. Lemke, H. J. Klausmeier, and C. W. Harris. Relationship of selected cognitive abilities to concept attainment and information processing. *Journal of Educational Psychology*, 58(1):27, 1967.
18. B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 9(3):1350–1371, September 2015.
19. Z. C. Lipton. The mythos of model interpretability. *CoRR*, abs/1606.03490, 2016.
20. D. Michie. Machine learning in the next five years. In *Proceedings of the Third European Working Session on Learning*, pages 107–122. Pitman, 1988.
21. D. Michie. Machine learning in the next five years. In *Proceedings of the Third European Working Session on Learning*, pages 107–122. Pitman, 1988.
22. T.M. Mitchell, R.M. Keller, and S.T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.
23. C. R. Mofizur and M. Numao. Top-down induction of recursive programs from small number of sparse examples. In *Advances in Inductive Logic Programming*. IOS Press, 1996.
24. M. Mozina, J. Zabkar, and I. Bratko. Argument based machine learning. *Artificial Intelligence*, 171(10–15):922 – 937, 2007.
25. S.H. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
26. S.H. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In *Proceedings of the 5th International Conference on Machine Learning*, pages 339–352. Kaufmann, 1988.
27. S.H. Muggleton and C. Feng. Efficient induction of logic programs. In S.H. Muggleton, editor, *Inductive Logic Programming*, pages 281–298. Academic Press, London, 1992.
28. S.H. Muggleton, D. Lin, N. Pahlavi, and A. Tamaddoni-Nezhad. Meta-interpretive learning: application to grammatical inference. *Machine Learning*, 94:25–49, 2014.
29. S.H. Muggleton, D. Lin, and A. Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. *Machine Learning*, 100(1):49–73, 2015.

30. S.H. Muggleton, L. De Raedt, D. Poole, I. Bratko, P. Flach, and K. Inoue. ILP turns 20: biography and future challenges. *Machine Learning*, 86(1):3–23, 2011.
31. G. L. Murphy and M. E. Lassaline. Hierarchical structure in concepts and the basic level of categorization. *Knowledge, concepts, and categories*, pages 93–131, 1997.
32. J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.
33. J.R. Quinlan and R.M. Cameron. Induction of logic programs: FOIL and related systems. *New Generation Computing*, 13:287–312, 1995.
34. R. Rios and S. Matwin. Efficient induction of recursive prolog definitions. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 240–248. Springer, 1996.
35. C. Rouveirol and J-F. Puget. A simple and general solution for inverting resolution. In *EWSL-89*, pages 201–210, London, 1989. Pitman.
36. U. Schmid and E. Kitzelmann. Inductive rule learning on the knowledge level. *Cognitive Systems Research*, 12(3):237–248, 2011.
37. U. Schmid, C. Zeller, T. Besold, A. Tamaddoni-Nezhad, and S.H. Muggleton. How does predicate invention affect human comprehensibility? In A. Russo and J. Cussens, editors, *Proceedings of the 26th International Conference on Inductive Logic Programming (ILP 2016, Sept. 4th-6th, London)*. Springer, 2017.
38. E. H. Shortliffe. A rule-based computer program for advising physicians regarding antimicrobial therapy selection. In *Proceedings of the 1974 annual ACM conference-Volume 2*, pages 739–739. ACM, 1974.
39. A. Srinivasan. The ALEPH manual. *Machine Learning at the Computing Laboratory, Oxford University*, 2001.
40. I. Stahl. Constructive induction in inductive logic programming: an overview. Technical report, Fakultat Informatik, Universitat Stuttgart, 1992.
41. L. Sterling and E. Y. Shapiro. *The art of Prolog: advanced programming techniques*. MIT Press, 1994.
42. A. M Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.
43. M. R. Wick and W. B. Thompson. Reconstructive expert system explanation. *Artificial Intelligence*, 54(1-2):33–70, 1992.