Learning from Human Teachers: Issues and Challenges for ILP in Bootstrap Learning

Sriraam Natarajan[1], Gautam Kunapuli[1], Richard Maclin[3], David Page[1], Ciaran O'Reilly[2], Trevor Walker[1] and Jude Shavlik[1]

[1]University of Wisconsin-Madison
Department of Biostatistics
{natarasr, kunapg, page, walker, shavlik}@biostat.wisc.edu

[2]SRI International
Artificial Intelligence Center
ciaran.oreilly@sri.com

[3]University of Minnesota, Duluth
Department of Computer Science
rmaclin@d.umn.edu

ABSTRACT

Bootstrap Learning (BL) is a new machine learning paradigm that seeks to build an electronic student that can learn using natural instruction provided by a human teacher and by bootstrapping on previously learned concepts. In our setting, the teacher provides (very few) examples and some advice about the task at hand using a natural instruction interface. To address this task, we use our Inductive Logic Programming system called WILL to translate the natural instruction into first-order logic. We present approaches to the various challenges BL raises, namely automatic translation of domain knowledge and instruction into an ILP problem and the automation of ILP runs across different tasks and domains, which we address using a multi-layered approach. We demonstrate that our system is able to learn effectively in over fifty different lessons across three different domains without any human-performed parameter tuning between tasks.

Categories and Subject Descriptors

I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving – *logic programming.*

General Terms

Algorithms, Design, Reliability, Experimentation, Human Factors.

Keywords

inductive logic programming, human teachers, automating setup problem

1. INTRODUCTION

One of the long cherished goals of Artificial Intelligence (AI) is to design agents that learn by interacting with humans, performing actions, receiving guidance and/or feedback from the human and improving its performance[3]. Traditional supervised learning approaches treat learning as a problem where some problem-dependent criteria (such as learning error, possibly combined with other means to control the inductive bias) is optimized given labeled examples.

Bootstrap Learning (BL) is a new learning paradigm proposed by Oblinger [5] which views learning as knowledge acquisition. The electronic student assumes all relevant knowledge is possessed by the teacher who teaches through human-like *natural instruction methods* including providing domain descriptions, pedagogical examples, telling of instructions, demonstration and feedback. In addition to teacher instruction, the student learns concepts that build upon one another through a "ladder" of lessons; lower rungs of the lesson ladder teach simpler concepts which are learned first and *bootstrap* (i.e., are used to learn more complex concepts).

The electronic student, called MABLE, the Modular Architecture for Bootstrap Learning Experiments [9] addresses the aforementioned limitations of the classical learning paradigm. First, MABLE consists of several different learning algorithms, which it is able to employ depending on the concept being taught and hence can learn a diverse range of tasks across different domains. Second, by virtue of the abstracted natural instruction and its ability to bootstrap complex behaviors, MABLE can be taught by non-programmers and non-experts. Thus, while traditional learning specializes by domain, BL specializes by the various natural instruction methods.

In this paper, we focus on one particular modality of teacher input: instruction by example, including teacher *hints* about specific examples. We use a logic-based approach that creates learned models expressed in first-order logic, which is called Inductive Logic Programming (ILP) [4]. ILP is especially well-suited for the "learning from examples" component in MABLE for two reasons. First, it can use a rich knowledge base that may have been provided to the learner initially or may have been learned/augmented during earlier lessons. Second, the declarative representation of both examples and learned rules makes it easier for the teacher and student to communicate about what has been learned so far; for example, a teacher can identify and correct student mistakes from earlier lessons. Similarly, the use of logic allows for sharing lessons of learned knowledge between modules that learn from different kinds of instruction.

This paper makes *four* key contributions: First, we present an ILP based system that learns from a human teacher in the presence of a very small number of examples. Second, we present the first of its kind methodology to automatically setup ILP runs that do not require intervention by an ILP expert (or any human for that matter). Third, is our algorithm that converts human advice and feedback into sentences written in first-order logic that are then used to guide the ILP search. The final and a very important contribution is the evaluation of the system in *5* different domains with teaching lessons for over *50* different concepts and where the

Cite as: Learning from Human Teachers: Issues and Challenges in Bootstrap Learning, Sriraam Natarajan, Gautam Kunapuli, David Page, Trevor Walker, Ciaran O'Reilly and Jude Shavlik, *AAMAS 2010 Workshop on Agents Learning Interactively from Human Teachers.*

[1] 1300 University Avenue, Medical Sciences Center, Madison WI 53705
[2] 333 Ravenswood Avenue, Menlo Park CA 94025
[3] 320 Heller Hall, 1114 Kirby Drive, Duluth MN 55812

correct concepts are learned without any modification of our algorithm between the lessons. Our computerized student is scored based on several test examples for each lesson and our student achieves a near-perfect grade when given advice prepared by a third party (who is not from our institution).

2. BL Challenges

We first introduce the learning framework and then outline the challenges of ILP and BL.

2.1 Learning Framework

The learning framework consists of the teacher, the environment, and the student interacting with each other. Given a domain within which learning takes place, the concepts to be learned are organized as *lessons* within a *curriculum* created by a separate group of researchers from outside our institution and not under our control. A lesson may be taught by more than one so-called *natural instruction method*. A lesson that teaches more complex concepts is broken down into two or more simpler lessons which are learned first and the more complex lesson is bootstrapped from the simpler ones. The structure of the curriculum is analogous to a "lesson" ladder with lower rungs representing simpler concepts and the complexity of the lessons increasing as we climb higher.

The teacher interacts with the student during teaching lessons using *utterance* messages and with the simulator using *imperative* messages which are actions that change the world state. The teacher can test during testing sessions with *imperative* messages requiring MABLE to answer questions and the teacher then evaluates the student's responses by providing a *grade*.

2.2 Inductive Logic Programming

ILP combines principles of two of the most important fields of AI: machine learning and knowledge representation. An ILP system learns a logic program given background knowledge as a set of first-order logic formulae and a set of examples expressed as *facts* represented in logic. In first-order logic, *terms* represent objects in the world and comprise *constants* (e.g., **Mary**), *variables* (**x**), and *functions* (**fatherOf(John)**). *Predicates* are functions with boolean return value. *Literals* are truth-valued and represent properties of objects and relations among objects, e.g. **married(John, Mary)**. Literals can be combined into compound sentences using connectives such as *AND, OR* and *NOT*. It is common [10] to convert sets of sentences into a canonical form, producing sets of *clauses*. We are developing a Java-based ILP system called WILL.

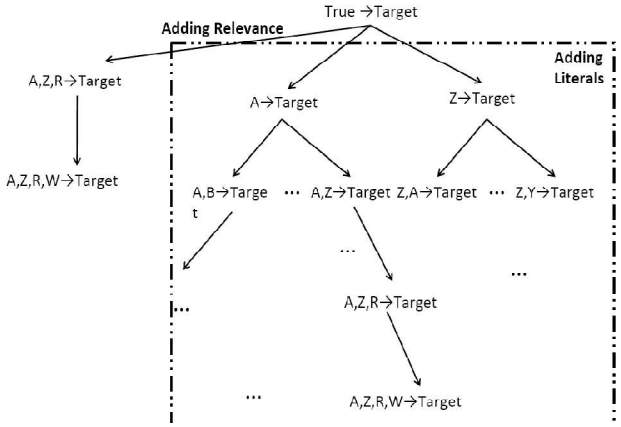

Figure 1. Sample search space to illustrate the usefulness of relevant statements to ILP.

Now, consider the ILP search space presented in **Error! Reference source not found.**, where logical variables are left out for simplicity and the possible features are denoted by a letter in *A* through *Z*. Let us assume that the true target concept is a conjunction of the predicates *A,Z,R* and *W*. ILP's search space without relevance is presented within the dashed box. Normally, ILP adds literals one after another, seeking the a short rule that covers all (or most of the) positive examples and none (or few) of the negatives. If there are *n* predicates then this can lead to a search of O(*n!*) combinations to discover the target concept. As can be seen by the portion of the search space that is outside the box, if a human teacher tells the ILP system that predicates A, Z, and R are relevant to the concept being learned, the amount of search that is needed can be greatly reduced. Such reduction can enable an ILP system to learn from a rather small number of examples. In the example, the teacher's hint specifies 3 out of 4 predicates that should appear in the target concept and hence an ILP learner needs to search over a smaller set of hypotheses to discover the correct concept.

Of course if the teacher is malicious or incompetent, then teacher-provided hints will *increase* the number of hypotheses that need to be considered since they increase the branching factor of the space being searched, but in this work we assume the human teacher has useful things to say, even if a bit imperfectly (teacher errors of omission are less harmful to WILL than errors of commission, as Figure 1 illustrates). The major BL challenge for ILP is that it has to be used, not only for different lessons within the same domain, but also across different domains; this necessitates the automation of the ILP setup problem *without the intervention of an ILP expert*.

Another important aspect requiring automated ILP runs is that the *parameter settings cannot change between different runs*. We cannot expect any human guidance regarding settings and need to find good default values that work broadly. Actually, our algorithms themselves try out a few parameter settings and use cross validation to choose good settings. However, given the large number of parameters in typical ILP systems (maximum rule length, modes, minimal acceptable accuracy of learned clauses, etc.), our algorithms cannot exhaustively try all combinations and hence must choose an appropriate set of candidate parameters that will work across dozens of learning tasks.

The goal of our ILP based agent is to translate the teacher's instructions into first-order logic. The instructions can be labels on example, as well as advice and/or feedback about these examples. We have created an interpreter that converts the advice to first-order logic by combining and generalizing the advice from individual examples and uses a cost-based search through the possible set of hypothesis to learn the target concept. BL also provides the opportunity for the student to refine its concept if it had learned an incorrect one. This setting is called *learning by feedback*, where the teacher provides explicit feedback such as providing the correct answer, pointing to important features or previously learned concept that the student should consider, etc. Our interpreter also interprets such feedback provided by the teacher and refines its learned concept.

2.3 BL Domains & Challenges

The domains of the BL project are **Unmanned Aerial Vehicle** (UAV) control, **Automated Task Force** (ATF), **International Space Station** (ISS).

UAV Domain Description: This domain involves operating a UAV and its camera to execute a reconnaissance mission. Tasks include determining if the UAV has enough fuel to accomplish a mission, achieving appropriate latitude, altitude, learning if there is a single (or multiple) stopped (or moving) truck(s) in a scenario, whether an object (say truck, building or intersection) is near another object of intersect. The idea is that the UAV is flying around and has to automatically identify scenarios that are potentially interesting from the defense perspective.

Figure 2 presents the lesson hierarchy for the domain. Each lesson is presented as an oval in the figure. An arrow between lessons indicate the bootstrapping relationship between them. For example, an arrow between *Near* and *TruckIsAtIntersection* indicates that the latter lesson requires the concept learned by former.

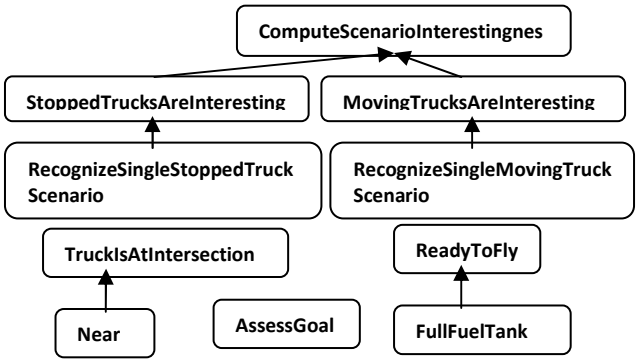

Figure 2. UAV lesson Hierarchy: A relationship A→B between lessons A and B indicates that B uses A in its concept.

UAV Challenges: The learner has to deal with complex structures such as position, which consists of attributes such as latitude, longitude, altitude, etc. Encoding these spatial attributes as part of one position literal would enable WILL to learn a smaller clause, but would increase the branching factor during search due to the additional arguments introduced by such a large-arity predicate. Representing these spatial attributes as separate predicates would decrease the branching factor at the expense of the target concept being a longer clause. In addition, the tasks involve learning the concept of "near" that can exist between any two objects of interest. In a later lesson, this concept might be used, for instance, to determine if a truck is at an intersection in which case the objects must be specialized to be of the types *truck* and *intersection*. It is a challenge for ILP systems to automatically generalize and specialize at different levels of the type hierarchy. Finally, this domain requires extensive "bootstrapping" as can be seen from Figure 2, which presents a heirarchy organizing the UAV lessons, and requires the object hierarchies to be able to generalize across different lessons.

ATF Domain Description: The goal of the ATF domain is to teach the student how to command a company of armored platoons to move from one battlefield location to another in accordance with military doctrine. The lessons are organized based on the complexity of tasks. At the lowest level are the tasks concerning individual vehicles and segments of vehicles. At a higher level are the tasks concerning platoons (sets of segments) while at the top-most level are the tasks of a company which is a set of platoons.

ATF Challenges: ATF poses at least two key challenges for application of ILP to BL. First, is the presence of a large number of numeric features. For example, there are distances and angles between vehicles, segments, platoons, and companies. For each of these objects, there are numeric attributes such as direction, location (in three dimensions), speed, etc. All these numeric features require ILP to select good thresholds or intervals, which can lead to a large number of features. The second important challenge is the deep nesting of the object structure. Each company has a list of platoons each of which has a list of segments that contain a list of vehicles. This deep nesting requires ILP to construct the predicates and features at the appropriate level of the object hierarchy. While this might not appear as a major issue with individual runs, it should be noted that the same settings have to be used across all the lessons for all the domains.

ISS Domain Description: The ISS curriculum places the student in a role of a flight controller who must detect and diagnose problems within the thermal control system of the International Space Station. The lessons include teaching the student what constitutes an emergency/alert, and *how* to post observation reports concerning actionable alerts. Examples of these include learning the conditions for alerting abnormal observations, warning, emergency and caution

ISS Challenges: This domain poses several issues that are not prominent in the other ones. The key challenge is that the number of features in the domain is very large. The fact-base of the domain consists of all the small parts, measurements, reports of the ISS and hence is significantly larger than the other domains (100's of features for a single example). A direct consequence is that the amount of time taken to construct the predicates is far greater than the other domains. This is an important issue due to the fact the learning strategies (in our case, learning by examples) have a fixed learning time. Within this time limit, the student has to interpret the teacher's statements, convert to its internal representation (in our case, first-order logic statements), learn and get evaluated on the test-example. Unlike the domains, this is not inherently relational. There are specific valves and meters that should be considered while learning the target concept. ILP, which is a powerful tool for learning first-order logic models that allow for generalization needs to consider objects at the grounded level in this domain.

3. SOLVING BL PROBLEMS

We now present the two main steps of our approach, namely, *interpreting relevance* and adopting a *multi-layered strategy* for automating ILP runs.

3.1 Interpreting Relevance

One of the key challenges in BL is learning from a very small number of examples. A human teacher will not spare the time and effort to specify thousands of examples that any common machine learning algorithm requires to learn a reasonable target concept. Instead, the human teacher provides some information (that we call *relevance statements* or *advice*) about the target concept that the student uses to accelerate its learning. For instance, when teaching the concept of a *full fuel tank*, the teacher might gesture to the fuel capacity of the tank and the current fuel level of the tank. The student might then infer the relationship between the two attributes. The main advantage of such a specification of relevant attributes is that it drastically reduces the search space (i.e., the search through the list of possible features in the target concept). Note that many possible features, such as color, length, weight, tire pressure, etc. could spuriously discriminate between positive and negative examples if the total number of examples is

very small (which is the case in the BL lessons, see Section 5). Thus, relevance statements become very critical in discovering the correct target concept.

We next outline our algorithm for interpreting the relevance statements provided by the teacher. We first illustrate the process of interpretation with an example before presenting the algorithm formally. Consider the lesson, *RecognizeSingleStoppedTruckScenario* in the UAV domain. The goal in this lesson is to identify if there is one and only one stopped truck in the scenario . We now present the teacher utterances followed by our interpretation of the statements.

```
RelevantRelationship(arg1=SameAs(arg1 = 1,
   arg2 = GetLength(arg1 = Of(arg1 =actors)
    arg2 = Scenario(actors = [Truck( name =
      Truck19, latitude = -10,longitude =
      10,moveStatus = Stopped)])))
```

Advice is provided using *Relevant* statements in BL. In the above statement, the teacher states that the length (size) of the actor list of the current scenario, should be 1. After the above relevance statement the teacher proceeds to give further instructions, here talking about a different example:

```
Gesture(atObject = Truck(name= Truck17, latitude
 =-10,longitude = 10, moveStatus = Stopped))
RelevantRelationship(arg1= InstanceOf (arg1 =
 this, arg2 = Truck))
```

In the above statements, the teacher first gestures at (points to) an object (Truck17 in this case) and explains that it being an instance of a truck is relevant to the target concept. The teacher further utters the following:

```
RelevantRelationship(arg1 =SameAs(arg1 = Of(arg1 =
 moveStatus, arg2 = Indexical(name =  this))
  arg2 = Stopped))
```

The above statement specifies that the *moveStatus* of the truck being "*Stopped*" is relevant to the target concept. The term *Indexical* is used to access the object that is being gestured at most recently by the teacher. Hence *Indexical(name = this)* here refers to the truck that has been gestured to earlier. Hence the teacher utters that the actors list of the scenario must be of size 1, that the object in that list must be of the type truck and that its move status must be equal to stopped. We will now proceed to explain how WILL interprets these statements and constructs background knowledge and partial answers correspondingly.

First, WILL identifies the interesting and relevant features from the above statements. WILL first creates the following interesting predicates:

```
isaInterestingComposite(Truck19)
isaInterestingComposite(Truck17)
isaInterestingNumber(1)
isaInterestingComposite(Scenario1)
isaInterestingSymbol(Stopped)
```

A key challenge when dealing with teacher-instruction about specific examples is "what should be generalized (i.e., to a logical variable) and what should remain constant?" The above facts provide WILL with some candidate constants that should be considered; WILL initially uses variables for all the arguments in the rules it is learning, but it also considers replacing variables with constants. WILL next creates the following relevant statements:

```
relevant: Vehicle_moveStatus
relevant: Scenario
relevant: Scenario_actors
relevant: Truck
relevant: sameAs
```

The features (attributes, objects, and relations) that are mentioned in the relevant statements are considered as relevant for the target concept. Consequentially, these features get lower scores when searching through the space of ILP rules and computing the cost of candidate rules. WILL then proceeds to construct rules corresponding to the relevance statements it receives. In the following rules, assume *S* is of type *scenario*, L is of type *list*, T is of type *truck,* and I is an *integer*. Following Prolog notation, commas denote logical AND. One rule WILL creates from teacher-provided instruction is

```
pred3(S) IF
 Scenario_actors(S,L),length(L,I),sameAs(I,1).
```

The above rule is constructed from the first relevant statement (of a positive example) that specifies that the length of the *actors* list in a scenario must be of size 1. A rule will now be constructed for the gesture that points at Truck19 in the list.

```
pred5(T,S) IF Scenario_actors(S,L),member(T,L)
```

Similarly, rules will be created for the other relevant statements corresponding to the instance of and the move status of the truck.

```
pred7(T,S) IF Truck(T, S)
```

The above rules uses the previous rule in asserting that the object that is a member of the list is of the type *truck*. Finally, the last relevance statement is interpreted as:

```
pred9(T,S) IF moveStatus(T,S),sameAs(S,stopped)
```

Once these rules are created for a particular example, WILL creates the combinations by combining the pieces of advice using the logical connective AND.

```
relevantFromPosEx1(S) IF
pred3(S),pred5(T,S),pred7(T,S), pred9(T,S)
```

WILL then proceeds to construct similar statements for the second example. Once all the individual examples are processed and the rules are created for each of the examples, WILL then proceeds to construct combinations of the rules in order to generalize across all the examples. The simplest combination is the combination of all rules from all positive examples and all the rules from all negative examples.

```
posCombo(S) IF
relevantFromPosEx1(S),...,relevantFromPosExN(S)
```

Similarly the negCombo is constructed by taking the negation of the negative relevantANDs.

```
negCombo(S) IF
~relevantFromNegEx1(S),...,~relevantFromNegExN(S)
```

We denote the negation of a concept by ~. Hence, by now our rules generalize positive and negative examples separately. Then WILL constructs the cross product across the different combinations and adds them to the background.

```
allCombo(S) IF  posCombo(S),negCombo(S)
```

All the predicates (*relevantFrom's*, *posCombo*, *negCombo*, *allCombo,* etc) are added to the background during search. and are marked as being relevant to the target concept. We also combine the advice about examples using the logical connective OR. We use both AND and OR to combine because a human teacher might be teaching the computer learning a new conjunctive concept with each example illustrating only one piece of the concept, or the teacher might be teaching a concept with several alternatives, with

each alternative illustrated via a different example. We refer to such rules as *comboRules*.

The algorithm for interpreting relevance is presented in Table 1. (Our ILP system can handle tasks that involve multiple categories by repeatedly treating them as "1 versus the others" classification problems.) WILL interprets the advice and creates relevant features corresponding to the objects and features mentioned in the relevant statements, as illustrated above.

The net result is that our algorithm has hypothesized a relatively small number of individual and 'compound' general rules that can be evaluated using the (small number of) labeled examples provided by its human teacher. Should these prove insufficient, WILL can combine and extend (by using the 'primitive' features in the domain at hand) by further searching of the hypothesis space.

Table 1. Algorithm For Interpreting Relevance.

For each category (e.g. TRUE and FALSE)
  For each example
  For each relevant statement about that example
    Construct relevant features
    Construct relevant rules for the particular example

  Combine the rules from individual examples to form "combo" rules about the current category.

Combine the rules from different examples to form "mega" rules about the concept as a whole.

3.2 Multi-Layered Strategy

One of the key issues with several machine learning methods is the periodic intervention by the domain expert to select features, tune parameters and set up runs. This is particularly true of ILP where researchers face the problem of designing new predicates, guiding ILP's search, setting additional parameters, etc. BL brings a major challenge for ILP in this area, because WILL must automatically set up training without the intervention of an ILP expert. This is needed because human teachers cannot be expected to understand the algorithmic details of a learning approach; rather they communicate with the student in and as natural and human-like dialog as is feasible [8]. This necessitates the guiding of search automatically in a *domain independent* manner. Automatic parameter selection methods such as the one proposed in [1] are not useful in our system due to the fact that we do not have access to a large number of examples. Instead we resort to a multi-layered strategy that tries several approaches to learn the target concept.

Table 2 presents the algorithm of multi-layered strategy called *Onion*. The innermost layer implements the basic strategy: invoking WILL after automated mode construction, using only the relevant combinations of features (as told by the teacher). This means that WILL initially explores a very restricted hypothesis space. If no theory is learned or if the learned theory has a poor score (based on heuristics), then the hypothesis space is expanded, say by considering features mentioned by the teacher. Continuing this way, our multi-layered approach successively expands the space of hypotheses until an acceptable theory is found. At each level, the algorithm considers different clause length and different values of coverage (#pos examples covered - #neg examples covered). Whenever a theory that fits the specified criteria is found, the algorithm returns the theory.

As a final note, while the teacher and the learner follow a fixed protocol while communicating via Interlingua, interpreting relevance amounts to more than simply implementing a rule-based parsing system. This is because of the ambiguity that is prevalent in every teacher relevance statement, in particular as to how general the advice. It is this ambiguity, of whether the teacher advice is about specific examples or applies to all examples generally, that necessitates a relevance interpreter as in Table 1.

Table 2 Multi-layered Strategy.

Procedure: Onion (*facts, background, examples*) returns *theory*

// *n* positive examples and *m* negative examples

While (*time remaining*)

1. Include only combo-rules that are generated by WILL for the search. Call WILLSEARCH. If perfect theory found, return
2. Expand search space to include all relevant features. Call WILLSEARCH. If perfect theory found, return
3. Expand search space to include all features. Call WILLSEARCH. If perfect theory found, return
4. Flip the example labels and call Onion with new examples

End-while

If no theory learned, return the largest combo-rule

Procedure: WILLSEARCH returns *theory*

- For rule length 2 to *maxRuleLength*
  - For coverage = *n* to *n/2*
    - Search for the acceptable theory. If found, return it

3.3 A Layered Approach for ILP

Having outlined the relevance interpreter and *Onion*, we now present the complete learning strategy in Table 3. We first parse all the *Interlingua* messages and create examples both positive and negative. In the case of multi-class problems, we pose the problem as one vs. others. Then the ground facts and background knowledge are constructed. The relevance interpreter then creates the comboRules. Finally, *Onion* is called with the background rules and facts to learn the target concept. Once the target concept is learned, the teacher evaluates the target on a few test examples. If the theory is unacceptable, the teacher provides more examples and/or relevant statements as a feedback, thus aiding WILL to learn a better concept.

Table 3. Learning Strategy.

Procedure: Strategy(IL Messages) returns theory

1. Construct examples(pos and neg), facts (ground truth) & background
2. Parse relevant statements, construct *comboRules* and add to background
3. Call Onion(*facts, background, examples*)
4. If acceptable theory is found, return *theory*
   Else call for the feedback lesson to obtain more examples and/or relevant statements. Go to Step 1.

4. ADDITIONAL ISSUES

Generation of Negative Examples. In general, ILP requires a large number of examples to learn a concept. While this is a challenge in all of machine learning, the need to learn complex relational concepts in first-order logic makes it even more so in ILP. In some domains, it is natural for a teacher to say that a

particular world state contains a single positive example; for example, it is natural for a teacher to point to a set of three blocks and state that they form a stack. It is a reasonable assumption that various combinations of the rest of the blocks in that scene do not form a stack and hence, WILL assumes these are (putative) negative examples. We have found that for most of the lessons provided in BL there is such a need for automatically constructing negatives because instruction contains mainly positive examples.

Another way to express negative examples is to say some world state does *not* contain any instances of the concept being taught: "the current configuration of blocks contains no stacks". Assume the teacher indicates `isaStack` takes three arguments, each of which is of type `block`. If WILL is presented with a world containing *N* blocks where there are no stacks, it can create N^3 negative examples. In general, negative examples are generated by instantiating the arguments of predicates whose types we may have been told, in all possible ways using typed constants encountered in world states; examples known to be positive are filtered out. Depending on the task, the student may have either teacher-provided negatives or induced negatives. As we do not want to treat these identically, WILL allows *costs* to be assigned to examples ensuring that the cost of covering a putative negative can be *less* than covering a teacher-provided one.

Learning the Negation of a Concept. Human teachers typically gauge the difficulty of concepts being taught by human comprehensibility, in terms of which, accurate, short, conjunctive rules are preferred. When learning concepts such as `outOfBounds` in a soccer field, the target concept might have a large set of disjunctions (since it can be out of bounds on any of four sides). It is easier to learn if the ball is *in bounds* and then *negate the learned concept*. Our learning bias here is that our benevolent teacher is teaching a concept that is simple to state, but we are not sure if the concept or its negation is simple to state, so we always consider both. For a small number of examples, it is usually hard to learn a disjunctive rule, especially if the examples are not the best ones, but rather only 'reasonable' in that they were *near* the boundaries, but not exactly *next to* them.

5. CONCLUSION

As mentioned earlier, our implemented system perfectly learned (100% accuracy) *56* lessons from a combination of training examples and teacher-provided hints. Running our ILP system without these hints - i.e., only using the training examples, for which there was an average of *7.6* labeled examples per concept taught - produced an average accuracy on held-aside examples of 63.9% (there was a 50-50 mixture of positive and negative examples, so 50% is the default accuracy of random guessing).

We have shown how the naturally provided human advice can be absorbed by ILP approach in order to learn a large number of concepts across a handful of domains. None of the advice, nor the lessons solved were created by us. Instead our task was to make effective use of the provided advice to learn the intended concepts while given only a small number of labeled examples.

The ILP approach allows learning to be applied to much richer types of data than the vast majority of machine-learning methods, due to its use of first-order logic as a representation for both data and hypotheses. However, ILP requires substantial experience to properly set up the 'hypothesis space' it searches. The natural teacher-learner interaction in our BL project is being interpreted by WILL as guidance for defining ILP hypothesis spaces, as well as biasing the search in such spaces toward the most promising areas. Finally, it should be noted that while these teacher instructions significantly influence the ILP algorithm in terms of which hypotheses it considers, the algorithm is still able to make additions to the teacher's instructions; decide which teacher instructions should be kept and which should be discarded; and choose how to integrate instructions about individual examples into a general concept. In other words, the human teacher is advising, rather than commanding, the student who still has the capability to make decisions on its own.

Human advice taking has long been explored in AI in the context of reinforcement learning [2], where the knowledge provided by the human is converted into a set of rules and knowledge-based neural networks are used to represent the utility function of the RL agent. Advice has also been incorporated in ILP systems [7] to learn constant free horn clauses. The key difference in our system is the presence of a very small number of examples.

Currently, we are focusing on our layered approach, to more robustly automate ILP in these different tasks. Also, we are currently looking at more richly exploiting teacher-provided feedback beyond statements about which features and objects are relevant. One possible future direction is to explore the possibility of refining the learned theories using teacher feedback in the lines of theory refinement for ILP [6]. Refining teacher's advice is important as it provides room for teacher mistakes.

6. ACKNOWLEDGMENTS

The authors gratefully acknowledge support of the Defense Advanced Research Projects Agency under DARPA grant HR0011-07-C-0060. Views and conclusions contained in this document are those of the authors and do not necessarily represent the official opinion or policies, either expressed or implied of the US government or of DARPA.

7. REFERENCES

[1] Kohavi, R. and John, G. Automatic parameter selection by minimizing estimated error. In *ICML*, 1995.

[2] Maclin, R. and Shavlik, J. W. Creating advice-taking reinforcement learners. *Mach. Learn.*, 22, 1996.

[3] McCarthy, J. The advice taker, a program with common sense. In *Symp. on the Mechanization of Thought Processes*, 1958.

[4] Muggleton S. and De Raedt, L. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[5] Oblinger, D. Bootstrap learning - external materials. `http:// www.sainc.com/bl-extmat`, 2006.

[6] Ourston, D. and Mooney, R. Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, 66:273–309, 1994.

[7] Pazzani, M. and Kibler D. The utility of knowledge in inductive learning. *Mach. Learn.* 9:57–94, 1992.

[8] Quinlan, J. R. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, 1986.

[9] Shen, J., Mailler, R., Bryce, D. and O'Reilly, C. MABLE: a framework for learning from natural instruction. In *AAMAS*, 2009.

[10] Russell, S. and Norvig, P. *Artificial Intelligence: A Modern Approach (Second Edition)*. Prentice Hall, 2003.