ILP: Just Do It

David Page

Dept. of Biostatistics and Medical Informatics
and Dept. of Computer Sciences
University of Wisconsin
1300 University Ave., Rm 5795 Medical Sciences
Madison, WI 53706
U.S.A.
Email: page@biostat.wisc.edu

Abstract. Inductive logic programming (ILP) is built on a foundation laid by research in other areas of computational logic. But in spite of this strong foundation, at 10 years of age ILP now faces a number of new challenges brought on by exciting application opportunities. The purpose of this paper is to interest researchers from other areas of computational logic in contributing their special skill sets to help ILP meet these challenges. The paper presents five future research directions for ILP and points to initial approaches or results where they exist. It is hoped that the paper will motivate researchers from throughout computational logic to invest some time into "doing" ILP.

1 Introduction

Inductive Logic Programming has its foundations in computational logic, including logic programming, knowledge representation and reasoning, and automated theorem proving. These foundations go well beyond the obvious basis in definite clause logic and SLD-resolution. In addition ILP has heavily utilized such theoretical results from computational logic as Lee's Subsumption Theorem [18], Gottlob's Lemma linking implication and subsumption [12], Marcinkowski and Pacholski's result on the undecidability of implication between definite clauses [22], and many others. In addition to utilizing such theoretical results, ILP depends crucially on important advances in logic programming implementations. For example, many of the applications summarized in the next brief section were possible only because of fast deductive inference based on indexing, partial compilation, etc. as embodied in the best current Prolog implementations. Furthermore, research in computational logic has yielded numerous important lessons about the art of knowledge representation in logic that have formed the basis for applications. Just as one example, definite clause grammars are central to several ILP applications within both natural language processing and bioinformatics.

ILP researchers fully appreciate the debt we owe to the rest of computational logic, and we are grateful for the foundation that computational logic has provided. Nevertheless, the goal of this paper is not merely to express gratitude, but

also to point to the present and future needs of ILP research. More specifically, the goal is to lay out future directions for ILP research and to attract researchers from the various other areas of computational logic to contribute their unique skill sets to some of the challenges that ILP now faces.[1] In order to discuss these new challenges, it is necessary to first briefly survey some of the most challenging application domains of the future. Section 2 provides such a review. Based on this review, Section 3 details five important research directions and concomitant challenges for ILP, and Section 4 tries to "close the sale" in terms of attracting new researchers.

2 A Brief Review of Some Application Areas

One of the most important application domains for machine learning in general is bioinformatics, broadly interpreted. This domain is particularly attractive for (1) its obvious importance to society, and (2) the plethora of large and growing data sets. Data sets obviously include the newly completed and available DNA sequences for *C. elegans* (nematode), *Drosophila* (fruitfly), and (depending on one's definitions of "completed" and "available") man. But other data sets include gene expression data (recording the degree to which various genes are expressed as protein in a tissue sample), bio-activity data on potential drug molecules, x-ray crystallography and NMR data on protein structure, and many others. Bioinformatics has been a particularly strong application area for ILP, dating back to the start of Stephen Muggleton's collaborations with Mike Sternberg and Ross King [29, 16]. Application areas include protein structure prediction [29, 37], mutagenicity prediction [17], and pharmacophore discovery [7] (discovery of a 3D substructure responsible for drug activity that can be used to guide the search for new drugs with similar activity). ILP is particularly well-suited for bioinformatics tasks because of its abilities to take into account background knowledge and structured data and to produce human-comprehensible results. For example, the following is a potential pharmacophore for ACE inhibition (a form of hypertension medication), where the spacial relationships are described through pairwise distances.[2]

```
Molecule A is an ACE inhibitor if:
   molecule A contains a zinc binding site B, and
   molecule A contains a hydrogen acceptor C, and
   the distance between B and C is 7.9 +/- .75 Angstroms, and
   molecule A contains a hydrogen acceptor D, and
   the distance between B and D is 8.5 +/- .75 Angstroms, and
   the distance between C and D is 2.1 +/- .75 Angstroms, and
   molecule A contains a hydrogen acceptor E, and
   the distance between B and E is 4.9 +/- .75 Angstroms, and
```

[1] Not to put too fine a point on the matter, this paper contains unapologetic proselytizing.

[2] Hydrogen acceptors are atoms with a weak negative charge. Ordinarily, zinc-binding would be irrelevant; it is relevant here because ACE is one of several proteins in the body that typically contains an associated zinc ion. This is an automatically generated translation of an ILP-generated clause.

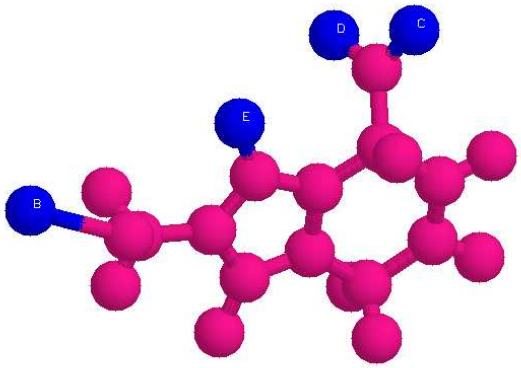

Fig. 1. ACE inhibitor number 1 with highlighted 4-point pharmacophore.

```
the distance between C and E is 3.1 +/- .75 Angstroms, and
the distance between D and E is 3.8 +/- .75 Angstroms.
```

Figures 1 and 2 show two different ACE inhibitors with the parts of pharmacophore highlighted and labeled.

A very different type of domain for machine learning is natural language processing (NLP). This domain also includes a wide variety of tasks such as part-of-speech tagging, grammar learning, information retrieval, and information extraction. Arguably, natural language translation (at least, very rough-cut translation) is now a reality—witness for example the widespread use of Altavista's Babelfish. Machine learning techniques are aiding in the construction of information extraction engines that fill database entries from document abstracts (e.g., [3]) and from web pages (e.g., WhizBang! Labs, http://www.whizbanglabs.com). NLP became a major application focus for ILP in particular with the ESPRIT project ILP^2. Indeed, as early as 1998 the majority of the application papers at the ILP conference were on NLP tasks.

A third popular and challenging application area for machine learning is knowledge discovery from large databases with rich data formats, which might contain for example satellite images, audio recordings, movie files, etc. While Dzeroski has shown how ILP applies very naturally to knowledge discovery from ordinary relational databases [6], advances are needed to deal with multimedia databases.

ILP has advantages over other machine learning techniques for all of the preceding application areas. Nevertheless, these and other potential applications also highlight the following shortcomings of present ILP technology.

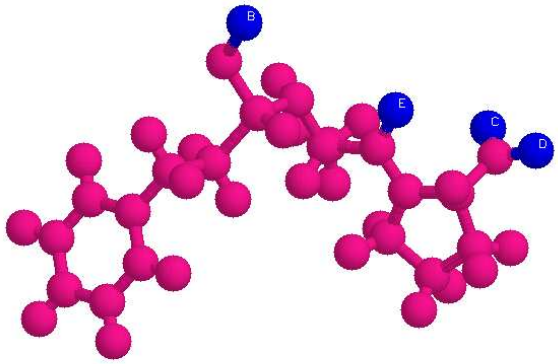

Fig. 2. ACE inhibitor number 2 with highlighted 4-point pharmacophore.

- Other techniques such as hidden Markov models, Bayes Nets and Dynamic Bayes Nets, and bigrams and trigrams can expressly represent the probabilities inherent in tasks such as part-of-speech tagging, alignment of proteins, robot maneuvering, etc. Few ILP systems are capable of representing or processing probabilities.[3]
- ILP systems have higher time and space requirements than other machine learning systems, making it difficult to apply them to large data sets. Alternative approaches such as stochastic search and parallel processing need to be explored.
- ILP works well when data and background knowledge are cleanly expressible in first-order logic. But what can be done when databases contain images, audio, movies, etc.? ILP needs to learn lessons from constraint logic programming regarding the incorporation of special-purpose techniques for handling special data formats.
- In scientific knowledge discovery, for example in the domain of bioinformatics, it would be beneficial if ILP systems could collaborate with scientists rather than merely running in batch mode. If ILP does not take this step, other forms of collaborative scientific assistants will be developed, supplanting ILP's position within these domains.

[3] It should be noted that Stephen Muggleton and James Cussens have been pushing for more attention to probabilities in ILP. Stephen Muggleton initiated this direction with an invited talk at ILP'95 and James Cussens has a recently-awarded British EPSRC project along these lines. Nevertheless, litte attention has been paid to this shortcoming by other ILP researchers, myself included.

In light of application domains and the issues they raise, the remainder of this paper discusses five directions for future research in ILP. Many of these directions require fresh insights from other areas of computational logic. The author's hope is that this discussion will prompt researchers from other areas to begin to explore ILP.[4]

3 Five Directions for ILP Research

Undoubtedly there are more than five important directions for ILP research. But five directions stand out clearly at this point in time. They stand out not only in the application areas just mentioned, but also when examining current trends in AI research generally. These areas are

- incorporating explicit probabilities into ILP
- stochastic search
- building special-purpose reasoners into ILP
- enhancing human-computer interaction to make ILP systems true *collaborators* with human experts
- parallel execution using commodity components

Each of these research directions can contribute substantially to the future widespread success of ILP. And each of these directions could benefit greatly from the expertise of researchers from other areas of computational logic. This section discusses these five research directions in greater detail.

3.1 Probabilistic Inference: ILP and Bayes Nets

Bayesian Networks have largely supplanted traditional rule-based expert systems. Why? Because in task after task we (AI practitioners) have realized that probabilities are central. For example, in medical diagnosis few universally true rules exist and few entirely accurate laboratory experiments are available. Instead, probabilities are needed to model the task's inherent uncertainty. Bayes Nets are designed specifically to model probability distributions and to reason about these distributions accurately and (in some cases) efficiently. Consequently, in many tasks including medical diagnosis [15], Bayes Nets have been found to be superior to rule-based systems. Interestingly, inductive inference, or machine learning, has turned out to be a very significant component of Bayes Net reasoning. Inductive inference from data is particularly important for developing or adjusting the conditional probability tables (CPTs) for various network nodes, but also is used in some cases even for developing or modifying the structure of the network itself.

[4] It is customary in technical papers for the author to refer to himself in the third person. But because the present paper is an invited paper expressing the author's opinions, the remainder will be much less clumsy if the author dispenses with that practice, which I now will do.

But not all is perfection and contentment in the world of Bayes Nets. A Bayes Net is less expressive than first-order logic, on a par with propositional logic instead. Consequently, while a Bayes Net is a graphical representation, it cannot represent relational structures. The only relationships captured by the graphs are conditional dependencies among probabilities. This failure to capture other relational information is particularly troublesome when using the Bayes Net representation in learning. For a concrete illustration, consider the task of pharmacophore discovery. It would be desirable to learn probabilistic predictors, e.g., what is the probability that a given structural change to the molecule fluoxetine (Prozac) will yield an equally effective anti-depressant (specifically, serotonin reuptake inhibitor)? To build such a probabilistic predictor, we might choose to learn a Bayes Net from data on serotonin reuptake inhibitors. Unfortunately, while a Bayes Net can capture the probabilistic information, it cannot capture the structural properties of a molecule that are predictive of biological activity.

The inability of Bayes Nets to capture relational structure is well known and has led to attempts to extend the Bayes Net representation [8, 9] and to study inductive learning with such an extended representation. But the resulting extended representations are complex and yet fall short of the expressivity of first-order logic. An interesting alternative for ILP researchers to examine is learning clauses with probabilities attached. It will be important in particular to examine how such representations and learning algorithms compare with the extended Bayes Net representations and learning algorithms. Several candidate clausal representations have been proposed and include probabilistic logic programs, stochastic logic programs, and probabilistic constraint logic programs; Cussens provides a nice survey of these representations [5]. Study already has begun into algorithms and applications for learning stochastic logic programs [27], and this is an exciting area for further work. In addition, the first-order representation closest to Bayes Nets is that of Ngo and Haddawy. The remainder of this subsection points to approaches for, and potential benefits of, learning these clauses in particular.

Clauses in the representation of Ngo and Haddawy may contain random variables as well as ordinary logical variables. A clause may contain at most one random variable in any one literal, and random variables may appear in body literals only if a random variable appears in the head. Finally, such a clause also has a Bayes Net fragment attached, which may be thought of as a constraint. This fragment has a very specific form. It is a directed graph of node depth two (edge depth one), with all the random variables from the clause body as parents of the random variable from the clause head.[5] Figure 3 provides an example of such a clause as might be learned in pharmacophore discovery (CPT not shown). This clause enables us to specify, through a CPT, how the probability of a molecule being active depends on the particular values assigned to the distance variables

[5] This is not exactly the definition provided by Ngo and Haddawy, but it is an equivalent one. Readers interested in deductive inference with this representation are encouraged to see [31, 30].

$D1$, $D2$, and $D3$. In general, the role of the added constraint in the form of a Bayes net fragment is to define a conditional probability distribution over the random variable in the head, conditional on the values of the random variables in the body. When multiple such clauses are chained together during inference, a larger Bayes Net is formed that defines a joint probability distribution over the random variables.

```
drug(Molecule,Activity_Level):-
    contains_hydrophobe(Molecule,Hydrophobe),
    contains_basic_nitrogen(Molecule,Nitrogen),
    contains_hydrogen_acceptor(Molecule,Acceptor),
    distance(Molecule,Hydrophobe,Nitrogen,D1),
    distance(Molecule,Hydrophobe,Acceptor,D2),
    distance(Molecule,Nitrogen,Acceptor,D3).
```

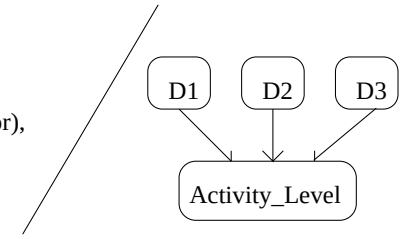

Fig. 3. A clause with a Bayes Net fragment attached (CPT not included). The random variables are *Activity_Level*, D1, D2, and D3. Rather than using a hard range in which the values of D1, D2, and D3 must fall, as the pharmacophores described earlier, this new representation allows us to describe a probability distribution over *Activity_Level* in terms of the values of D1, D2, and D3. For example, we might assign higher probabilities to high *Activity_Level* as D1 gets closer to 3 Angstroms from either above or below. The CPT itself might be a linear regression model, i.e. a linear function of D1, D2, and D3 with some fixed variance assumed, or it might be a discretized model, or other.

I conjecture that existing ILP algorithms can effectively learn clauses of this form with the following modification. For each clause constructed by the ILP algorithm, collect the positive examples covered by the clause. Each positive example provides a value for the random variable in the head of the clause, and because the example is covered, the example together with the background knowledge provides values for the random variables in the body. These values, over all the covered positive examples, can be used as the data for constructing the conditional probability table (CPT) that accompanies the attached Bayes Net fragment. When all the random variables are discrete, a simple, standard method exists for constructing CPTs from such data and is described nicely in [14]. If some or all of the random variables are continuous, then under certain assumptions again simple, standard methods exist. For example, under one set of assumptions linear regression can be used, and under another naive Bayes can be used. In fact, the work by Srinivasan and Camacho [35] on predicting levels of mutagenicity and the work by Craven and colleagues [4, 3] on information extraction can be seen as special cases of this proposed approach, employing linear regression and naive Bayes, respectively.

While the approach just outlined appears promising, of course it is not the only possible approach and may not turn out to be the best. More generally, ILP and Bayes Net learning are largely orthogonal. The former handles rela-

tional domains well, while the latter handles probabilities well. And both Bayes Nets and ILP have been applied successfully to a variety of tasks. Therefore, it is reasonable to hypothesize the existence and utility of a representation and learning algorithms that effectively capture the advantages of both Bayes net learning and ILP. The space of such representations and algorithms is large, so combining Bayes Net learning and ILP is an area of research that is not only promising but also wide open for further work.

3.2 Stochastic Search

Most ILP algorithms search a lattice of clauses ordered by subsumption. They seek a clause that maximizes some function of the size of the clause and coverage of the clause, i.e. the numbers of positive and negative examples entailed by the clause together with the background theory. Depending upon how they search this lattice, these ILP algorithms are classified as either bottom-up (based on least general generalization) or top-down (based on refinement). Algorithms are further classified by whether they perform a greedy search, beam search, admissible search, etc. In almost all existing algorithms these searches are deterministic. But for other challenging logic/AI tasks outside ILP, stochastic searches have consistently outperformed deterministic searches. This observation has been repeated for a wide variety of tasks, beginning with the 1992 work of Kautz, Selman, Levesque, Mitchell, and others on satisfiability using algorithms such as GSAT and WSAT (WalkSAT) [34, 33]. Consequently, a promising research direction within ILP is the use of stochastic search rather than deterministic search to examine the lattice of clauses. A start has been made in stochastic search for ILP and this section describes that work. Nevertheless many issues remain unexamined, and I will mention some of the most important of these at the end of this section.

ILP algorithms face not one but two difficult search problems. In addition to the search of the lattice of clauses, already described, simply testing the coverage of a clause involves repeated searches for proofs—"if I assume this clause is true, does a proof exist for that example?" The earliest work on stochastic search in ILP (to my knowledge) actually addressed this latter search problem. Sebag and Rouveirol [32] employed stochastic matching, or theorem proving, and obtained efficiency improvements over Progol in the prediction of mutagenicity, without sacrificing predictive accuracy or comprehensibility. More recently, Botta, Giordana, Saitta, and Sebag have pursued this approach further, continuing to show the benefits of replacing deterministic matching with stochastic matching [11, 2].

But at the center of ILP is the search of the clause lattice, and surprisingly until now the only stochastic search algorithms that have been tested have been genetic algorithms. Within ILP these have not yet been shown to significantly outperform deterministic search algorithms. I say it is surprising that only GAs have been attempted because for other logical tasks such as satisfiability and planning almost every other approach outperforms GAs, including simulated annealing, hill-climbing with random restarts and sideways moves (e.g. GSAT),

and directed random walks (e.g. WSAT) [33]. Therefore, a natural direction for ILP research is to use these alternative forms of stochastic search to examine the lattice of clauses. The remainder of this section discusses some of the issues involved in this research direction, based on my initial foray in this direction with Ashwin Srinivasan that includes testing variants of GSAT and WSAT tailored to ILP.

The GSAT algorithm was designed for testing the satisfiability of Boolean CNF formulas. GSAT randomly draws a truth assignment over the n propositional variables in the formula and then repeatedly modifies the current assignment by flipping a variable. At each step all possible flips are tested, and the flip that yields the largest number of satisfied clauses is selected. It may be the case that every possible flip yields a score no better (in fact, possibly even worse) than the present assignment. In such a case a flip is still chosen and is called a "sideways move" (or "downward move" if strictly worse). Such moves turn out to be quite important in GSAT's performance. If GSAT finds an assignment that satisfies the CNF formula, it halts and returns the satisfying assignment. Otherwise, it continues to flip variables until it reaches some pre-set maximum number of flips. It then repeats the process by drawing a new random truth assignment. The overall process is repeated until a satisfying assignment is found or a pre-set maximum number of iterations is reached.

Our ILP variant of this algorithm draws a random clause rather than a random truth assignment. Flips involve adding or deleting literals in this clause. Applying the GSAT methodology to ILP in this manner raises several important points. First, in GSAT scoring a given truth assignment is very fast. In contrast, scoring a clause can be much more time consuming because it involves repeated theorem proving. Therefore, it might be beneficial to combine the "ILP-GSAT" algorithm with the type of stochastic theorem proving mentioned above. Second, the number of literals that can be built from a language often is infinite, so we cannot test all possible additions of a literal. Our approach has been to base any given iteration of the algorithm on a "bottom clause" built from a "seed example," based on the manner in which the ILP system PROGOL [26] constrains its search space. But there might be other alternatives for constraining the set of possible literals to be added at any step. Or it might be preferable to consider changing literals rather than only adding or deleting them. Hence there are many alternative GSAT-like algorithms that might be built and tested.

Based on our construction of GSAT-like ILP algorithms, one can imagine analogous WSAT-like and simulated annealing ILP algorithms. Consider WSAT in particular. On every flip, with probability p (user-specified) WSAT makes an randomly-selected efficacious flip instead of a GSAT flip. An efficacious flip is a flip that satisfies some previously-unsatisfied clause in the CNF formula, even if the flip is not the highest-scoring flip as required by GSAT. WSAT outperforms GSAT for many satisfiability tasks because the random flips make it less likely to get trapped in local optima. It will be interesting to see if the benefit of WSAT over GSAT for satisfiability carries over to ILP. The same issues mentioned above for ILP- GSAT also apply to ILP-WSAT.

It is too early in the work to present concrete conclusions regarding stochastic ILP. Rather the goal of this section has been to point to a promising direction and discuss the space of design alternatives to be explored. Researchers with experience in stochastic search for constraint satisfaction and other logic/AI search tasks will almost certainly have additional insights that will be vital to the exploration of stochastic search for ILP.

3.3 Special-purpose Reasoning Mechanisms

One of the well-known success stories of computational logic is constraint logic programming. And one of the reasons for this success is the ability to integrate logic and special purpose reasoners or constraint solvers. Many ILP applications could benefit from the incorporation of special-purpose reasoning mechanisms. Indeed, the approach advocated in Section 3.1 to incorporating probabilities in ILP can be thought of as invoking special purpose reasoners to construct constraints in the form of Bayes Net fragments. The work by Srinivasan and Camacho mentioned there uses linear regression to construct a constraint, while the work by Craven and Slattery uses naive Bayes techniques to construct a constraint. The point that is crucial to notice is that ILP requires a "constraint constructor," such as linear regression, in addition to the constraint solver required during deduction. Let's now turn to consideration of tasks where other types of constraint generators might be useful.

Consider the general area of knowledge discovery from databases. Suppose we take the standard logical interpretation of a database, where each relation is a predicate, and each tuple in the relation is a ground atomic formula built from that predicate. Dzeroski and Lavrac show how ordinary ILP techniques are very naturally suited to this task, if we have an "ordinary" relational database. But now suppose the database contains some form of complex objects, such as images. Simple logical similarities may not capture the important common features across a set of images. Instead, special-purpose image processing techniques may be required, such as those described by Leung and colleagues [20, 19]. In addition to simple images, special-purpose constraint constructors might be required when applying ILP to movie (e.g. MPEG) or audio (e.g. MIDI) data, or other data forms that are becoming ever more commonplace with the growth of multimedia. For example, a fan of the Bach, Mozart, Brian Wilson, and Elton John would love to be able to enter her/his favorite pieces, have ILP with a constraint generator build rules to describe these favorites, and have the rules suggest other pieces or composers s/he should access. As multimedia data becomes more commonplace, ILP can remain applicable only if it is able to incorporate special-purpose constraint generators.

Alan Frisch and I have shown that the ordinary subsumption ordering over formulas scales up quite naturally to incorporate constraints [10]. Nevertheless, that work does not address some of the hardest issues, such as how to ensure the efficiency of inductive learning systems based on this ordering and how to design the right types of constraint generators. These questions require much further research involving real-world applications such as multimedia databases.

One final point about special purpose reasoners in ILP is worth making. Constructing a constraint may be thought of as inventing a predicate. Predicate invention within ILP has a long history [28, 39, 40, 25]. General techniques for predicate invention encounter the problem that the space of "inventable" predicates is unconstrained, and hence allowing predicate invention is roughly equivalent to removing all bias from inductive learning. While removing bias may sound at first to be a good idea, inductive learning in fact requires bias [23, 24]. Special purpose techniques for constraint construction appear to make it possible to perform predicate invention in way that is limited enough to be effective [35, 3].

3.4 Interaction with Human Experts

To discover new knowledge from data in fields such as telecommunications, molecular biology, or pharmaceuticals, it would be beneficial if a machine learning system and a human expert could act as a team, taking advantage of the computer's speed and the expert's knowledge and skills. ILP systems have three properties that make them natural candidates for collaborators with humans in knowledge discovery:

Declarative Background Knowledge ILP systems can make use of declarative background knowledge about a domain in order to construct hypotheses. Thus a collaboration can begin with a domain expert providing the learning system with general knowledge that might be useful in the construction of hypotheses. Most ILP systems also permit the expert to define the hypothesis space using additional background knowledge, in the form of a *declarative bias*.

Natural descriptions of structured examples Feature-based learning systems require the user to begin by creating features to describe the examples. Because many knowledge discovery tasks involve complex structured examples, such as molecules, users are forced to choose only composite features such as molecular weight—thereby losing information—or to invest substantial effort in building features that can capture structure (see [36] for a discussion in the context of molecules). ILP systems allow a structured example to be described naturally in terms of the objects that compose it, together with relations between those objects. The 2-dimensional structure of a molecule can be represented directly using its atoms as the objects and bonds as the relations; 3-dimensional structure can be captured by adding distance relations.

Human-Comprehensible Output ILP systems share with propositional–logic learners the ability to present a user with declarative, comprehensible rules as output. Some ILP systems can return rules in English along with visual aids. For example, the pharmacophore description and corresponding figures in Section 2 were generated automatically by PROGOL.

Despite the useful properties just outlined, ILP systems—like other machine learning systems—have a number of shortcomings as collaborators with humans

in knowledge discovery. One shortcoming is that most ILP systems return a single theory based on heuristics, thus casting away many clauses that might be interesting to a domain expert. But the only currently existing alternative is the version space approach, which has unpalatable properties that include inefficiency, poor noise tolerance, and a propensity to overwhelm users with too large a space of possible hypotheses. Second, ILP systems cannot respond to a human expert's questions in the way a human collaborator would. They operate in simple batch mode, taking a data set as input, and returning a hypothesis on a take-it-or-leave-it basis. Third, ILP systems do not question the input data in the way a human collaborator would, spotting surprising (and hence possibly erroneous) data points and raising questions about them. Some ILP systems will flag mutually inconsistent data points but to my knowledge none goes beyond this. Fourth, while a human expert can provide knowledge-rich forms of hypothesis justification, for example relating a new hypothesis to existing beliefs, ILP systems merely provide accuracy estimates as the sole justification.

To build upon ILP's strengths as a technology for human-computer collaboration in knowledge discovery, the above shortcomings should be addressed. ILP systems should be extended to display the following capabilities.

1. maintain and summarize alternative hypotheses that explain or describe the data, rather than providing a single answer based on a general-purpose heuristic;
2. propose to human experts practical sequences of experiments to refine or distinguish between competing hypotheses;
3. provide non-numerical justification for hypotheses, such as relating them to prior beliefs or illustrative examples (in addition to providing numerical accuracy estimates);
4. answer an expert's questions regarding hypotheses;
5. consult the expert regarding anomalies or surprises in the data.

Addressing such human-computer interface issues obviously requires a variety of logical and AI expertise. Thus contributions from other areas of computational logic, such as the study of logical agents, will be vital. While several projects have recently begun that investigate some of these issues,[6] developing collaborative systems is an ambitious goal with more than enough room for many more researchers. And undoubtedly other issues not mentioned here will become apparent as this work progresses.

3.5 Parallel Execution

While ILP has numerous advantages over other types of machine learning, including advantages mentioned at the start of the previous section, it has two

[6] Stephen Muggleton has a British EPSRC project on closed-loop learning, in which the human is omitted entirely. While this seems the reverse of a collaborative system, it raises similar issues, such as maintaining competing hypotheses and automatically proposing experiments. I am beginning a U.S. National Science Foundation project on collaborative systems with (not surprisingly) exactly the goals above.

particularly notable disadvantages—run time and space requirements. Fortunately for ILP, at the same time that larger applications are highlighting these disadvantages, parallel processing "on the cheap" is becoming widespread. Most notable is the widespread use of "Beowulf clusters" [1] and of "Condor pools" [21], arrangements that connect tens, hundreds, or even thousands of personal computers or workstations to permit parallel processing. Admittedly, parallel processing cannot change the order of the time or space complexity of an algorithm. But most ILP systems already use broad constraints, such as maximum clause size, to hold down exponential terms. Rather, the need is to beat back the large constants brought in by large real-world applications.

Yu Wang and David Skillicorn recently developed a parallel implementation of PROGOL under the Bulk Synchronous Parallel (BSP) model and claim superlinear speedup from this implementation [38]. Alan Wild worked with me at the University of Louisville to re-implement on a Beowulf cluster a top-down ILP search for pharmacophore discovery, and the result was a linear speedup [13]. The remainder of this section described how large-scale parallelism can be achieved very simply in a top-down complete search ILP algorithm. This was the approach taken in [13]. From this discussion, one can imagine more interesting approaches for other types of top-down searches such as greedy search.

The ideal in parallel processing is a decrease in processing time that is a linear function, with a slope near 1, of the number of processors used. (In some rare cases it is possible to achieve superlinear speed-up.) The barriers to achieving the ideal are (1) overhead in communication between processes and (2) competition for resources between processes. Therefore, a good parallel scheme is one where the processes are relatively independent of one another and hence require little communication or resource sharing. The key observation in the design of the parallel ILP scheme is that two competing hypotheses can be tested against the data completely independently of one another. Therefore the approach advocated here is to distribute the hypothesis space among different processors for testing against the data. These processors need not communicate with one another during testing, and they need not write to a shared memory space.

In more detail, for complete search a parallel ILP scheme can employ a master-worker design, where the master assigns different segments of the hypothesis space to workers that then test hypotheses against the data. Workers communicate back to the master all hypotheses achieving a pre-selected minimum valuation score (e.g. 95 % accuracy) on the data. As workers become free, the master continues to assign new segments of the space until the entire space has been explored. The only architectural requirements for this approach are (1) a mechanism for communication between the master and each worker and (2) read access for each worker to the data. Because data do not change during a run, this scheme can easily operate under either a shared memory or message passing architecture; in the latter, we incur a one-time overhead cost of initially communicating the data to each worker. The only remaining overhead, on either architecture, consists of the time spent by the master and time for master-worker communication. In "needle in a haystack" domains, which are the motivation

for complete search, one expects very few hypotheses to be communicated from workers to the master, so overhead for the communication of results will be low. If it also is possible for the master to rapidly segment the hypothesis space in such a way that the segments can be communicated to the workers succinctly, then overall overhead will be low and the ideal of linear speed-up can be realized.

4 Conclusions

ILP has attracted great interest within the machine learning and AI communities at large because of its logical foundations, its ability to utilize background knowledge and structured data representations, and its comprehensible results. But most of all, the interest has come from ILP's application successes. Nevertheless, ILP needs further advances to maintain this record of success, and these advances require further contributions from other areas of computational logic. System builders and parallel implementation experts are needed if the ILP systems of the next decade are to scale up to the next generation of data sets, such as those being produced by Affymetrix's (TM) gene expression microarrays and Celera's (TM) shotgun approach to DNA sequencing. Researchers on probability and logic are required if ILP is to avoid being supplanted by the next generation of extended Bayes Net learning systems. Experts on constraint satisfaction and constraint logic programming have the skills necessary to bring successful stochastic search techniques to ILP and to allow ILP techniques to extend to multimedia databases. The success of ILP in the next decade (notice I avoided the strong temptation to say "next millennium") depends on the kinds of interactions being fostered at Computational Logic 2000.

Acknowledgements

This work was supported in part by NSF grant 9987841 and by grants from the University of Wisconsin Graduate School and Medical School.

References

1. D. Becker, T. Sterling, D. Savarese, E. Dorband, U. Ranawake, and C. Packer. Beowulf: A parallel workstation for scientific computation. In *Proceedings of the 1995 International Conference on Parallel Processing (ICPP)*, pages 11–14, 1995.
2. M. Botta, A. Giordana, L. Saiita, and M. Sebag. Relational learning: Hard problems and phase transitions. 2000.
3. M. Craven and J. Kumlien. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, Heidelberg, Germany, 1999. AAAI Press.
4. M. Craven and S. Slattery. Combining statistical and relational methods for learning in hypertext domains. In *Proceedings of the Eighth International Conference on Inductive Logic Programming (ILP-98)*, pages 38–52. Springer Verlag, 1998.

5. J. Cussens. Loglinear models for first-order probabilistic reasoning. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*. Stockholm, Sweden, 1999.
6. S. Dzeroski. Inductive logic programming and knowledge discovery in databases. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. 1996.
7. P. Finn, S. Muggleton, D. Page, and A. Srinivasan. Discovery of pharmacophores using Inductive Logic Programming. *Machine Learning*, 30:241–270, 1998.
8. N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. Stockholm, Sweden, 1999.
9. N. Friedman, D. Koller, and A. Pfeffer. Structured representation of complex stochastic systems. In *Proceedings of the 15th National Conference on Artificial Intelligence*. AAAI Press, 1999.
10. A. M. Frisch and C. D. Page. Building theories into instantiation. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1995.
11. A. Giordana and L. Saitta. Phase transitions in learning with fol languages. Technical Report 97, 1998.
12. G. Gottlob. Subsumption and implication. *Information Processing Letters*, 24(2):109–111, 1987.
13. J. Graham, D. Page, and A. Wild. Parallel inductive logic programming. In *Proceedings of the Systems, Man, and Cybernetics Conference (SMC-2000)*, page To appear. IEEE, 2000.
14. D. Heckerman. A tutorial on learning with bayesian networks. Microsoft Technical Report MSR-TR-95-06, 1995.
15. D. Heckerman, E. Horvitz, and B. Nathwani. Toward normative expert systems: Part i the pathfinder project. *Methods of Information in Medicine*, 31:90–105, 1992.
16. R. King, S. Muggleton, R. Lewis, and M. Sternberg. Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proceedings of the National Academy of Sciences*, 89(23):11322–11326, 1992.
17. R. King, S. Muggleton, A. Srinivasan, and M. Sternberg. Structure-activity relationships derived by machine learning: the use of atoms and their bond connectives to predict mutagenicity by inductive logic programming. *Proceedings of the National Academy of Sciences*, 93:438–442, 1996.
18. C. Lee. *A completeness theorem and a computer program for finding theorems derivable from given axioms*. PhD thesis, University of California, Berkeley, 1967.
19. T. Leung, M. Burl, and P. Perona. Probabilistic affine invariants for recognition. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 1998.
20. T. Leung and J. Malik. Detecting, localizing and grouping repeated scene elements from images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page To appear, 2000.
21. M. Litzkow, M. Livny, and M. Mutka. Condor—a hunter of idle workstations. In *Proceedings of the International Conference on Distributed Computing Systems*, pages 104–111, 1988.
22. J. Marcinkowski and L. Pacholski. Undecidability of the horn-clause implication problem. In *Proceedings of the 33rd IEEE Annual Symposium on Foundations of Computer Science*, pages 354–362. IEEE, 1992.

23. T.M. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Department of Computer Science, Rutgers University, 1980.
24. T.M. Mitchell. Generalisation as search. *Artificial Intelligence*, 18:203–226, 1982.
25. S. Muggleton. Predicate invention and utilization. *Journal of Experimental and Theoretical Artificial Intelligence*, 6(1):127–130, 1994.
26. S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
27. S. Muggleton. Learning stochastic logic programs. In *Proceedings of the AAAI2000 Workshop on Learning Statistical Models from Relational Data*. AAAI, 2000.
28. S. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 339–352. Kaufmann, 1988.
29. S. Muggleton, R. King, and M. Sternberg. Protein secondary structure prediction using logic-based machine learning. *Protein Engineering*, 5(7):647–657, 1992.
30. L. Ngo and P. Haddawy. Probabilistic logic programming and bayesian networks. *Algorithms, Concurrency, and Knowledge: LNCS 1023*, pages 286–300, 1995.
31. L. Ngo and P. Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171:147–177, 1997.
32. M. Sebag and C. Rouveirol. Tractable induction and classification in fol. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pages 888–892. Nagoya, Japan, 1997.
33. B. Selman, H. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*. AAAI Press, 1994.
34. B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446. AAAI Press, 1992.
35. A. Srinivasan and R.C. Camacho. Numerical reasoning with an ILP system capable of lazy evaluation and customised search. *Journal of Logic Programming*, 40:185–214, 1999.
36. A. Srinivasan, S. Muggleton, R. King, and M. Sternberg. Theories for mutagenicity: a study of first-order and feature based induction. *Artificial Intelligence*, 85(1,2):277–299, 1996.
37. M. Turcotte, S. Muggleton, and M. Sternberg. Application of inductive logic programming to discover rules governing the three-dimensional topology of protein structures. In *Proceedings of the Eighth International Conference on Inductive Logic Programming (ILP-98)*, pages 53–64. Springer Verlag, 1998.
38. Y. Wang and D. Skillicorn. Parallel inductive logic for data mining. http://www.cs.queensu.ca/home/skill/papers.html#datamining, 2000.
39. R. Wirth and P. O'Rorke. Constraints on predicate invention. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 457–461. Kaufmann, 1991.
40. J. Zelle and R. Mooney. Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 817–822, San Mateo, CA, 1993. Morgan Kaufmann.