Efficiently Constructing Relational Features from Background Knowledge for Inductive Machine Learning

John M. Aronis and Foster J. Provost
Computer Science Department
University of Pittsburgh, Pittsburgh, PA 15260
aronis@cs.pitt.edu foster@cs.pitt.edu

Abstract

Most existing inductive learning systems form concept descriptions in propositional languages from vectors of basic features. However, many concepts are characterized by the relationships of individual examples to general domain knowledge. We describe a system that constructs relational terms efficiently to augment the description language of standard inductive systems. In our approach, examples and domain knowledge are combined into an inheritance network, and a form of spreading activation is used to find relevant relational terms. Since there is an equivalence between inheritance networks and relational databases, this yields a method for exploring tables in the database and finding relevant relationships among data to characterize concepts. We also describe the implementation of a prototype system on the CM-2 parallel computer and some experiments with large data sets.

1. Introduction.

Typical inductive learning systems, such as decision-tree learners [10] and rule learners [2], [6], form concept descriptions in propositional languages based on the similarities and differences between vectors of features. The set of features is static and completely determined beforehand. Furthermore, these learners do not take into account relationships among the examples, or relationships to general domain knowledge.

The goal of the work described in this paper is to use existing domain knowledge to create new terms, not already present in the features originally provided, for inductive learners to use. Our approach is to represent examples and background knowledge in the form of an inheritance hierarchy with roles (equivalent to a multi-table relational database). We then use parallel formula propagation techniques [1] to suggest relevant terms efficiently. Parallel formula propagation can find relationships in large knowledge bases, including relationships that span multiple functional links and relate multiple examples (see example below). The new terms are used to augment the description language of a standard machine learning program.

The use of domain knowledge is necessary when the features attached to individual examples do not capture abstractions and general distinctions that relate many examples of a concept. In some domains, typical inductive learning with only the basic features creates many small disjuncts that are inherently error-prone [4] [3], because of a lack of statistical confidence in a disjunct that covers very few examples. Creating higher-level features

can be useful for coalescing many related small disjuncts into a larger rule, with more statistical confidence [12]. As the examples below show, these larger rules can also be more understandable than the corresponding collection of small disjuncts.

In order for a system that combines data with prior domain knowledge to be useful, it must not require that the user know *a priori* what knowledge will be relevant to the learning problem. Thus, such systems must be able to deal with large knowledge bases that contain mostly irrelevant knowledge. When large knowledge bases are combined with large example sets, it is vital that efficient techniques are available–both for the representation of the prior knowledge and for the search for relevant prior knowledge. Previous work has shown how parallelism can help to scale up feature-based inductive learning to very large data sets [9]. In the present work we show how parallelism can be used to scale up learning with domain knowledge to large knowledge bases (perhaps in combination with large data sets).

2. Parallelism and Domain Knowledge.

We will use a simple 'blocks world' example to illustrate our basic ideas; later we will describe how the system performs on real-world learning problems.

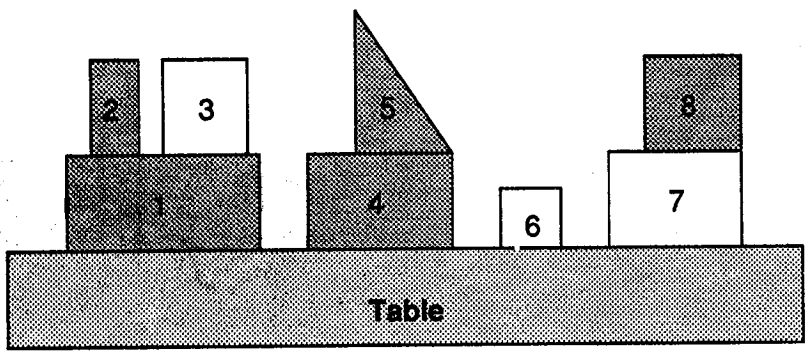

Figure 1: A Simple Example.

Consider the example in Figure 1. It shows blocks of various shapes, sizes, and colors. Additionally, some of the blocks are stacked on top of each other. A simple feature-based learner cannot learn a description of the concept set consisting of blocks 2, 3, and 5 since it is limited to reasoning only about their intrinsic features such as shape and color. To learn a simple description of the concept (it consists of blocks that are on other gray blocks) requires representing and reasoning with relational information.

We can represent the relevant information for this concept with the simple inheritance network in Figure 2. We can see that the blocks of the concept are connected by paths consisting of on and color links to the node Gray. The problem is that we do not know beforehand that this combination of links characterizes the set. In a more complex example there would be many links coming from each node, and even after looking at the diagram it might not be clear what relationships were common to all the nodes of the concept.

Aronis [1] developed a method, called *parallel formula propagation* to explore all the

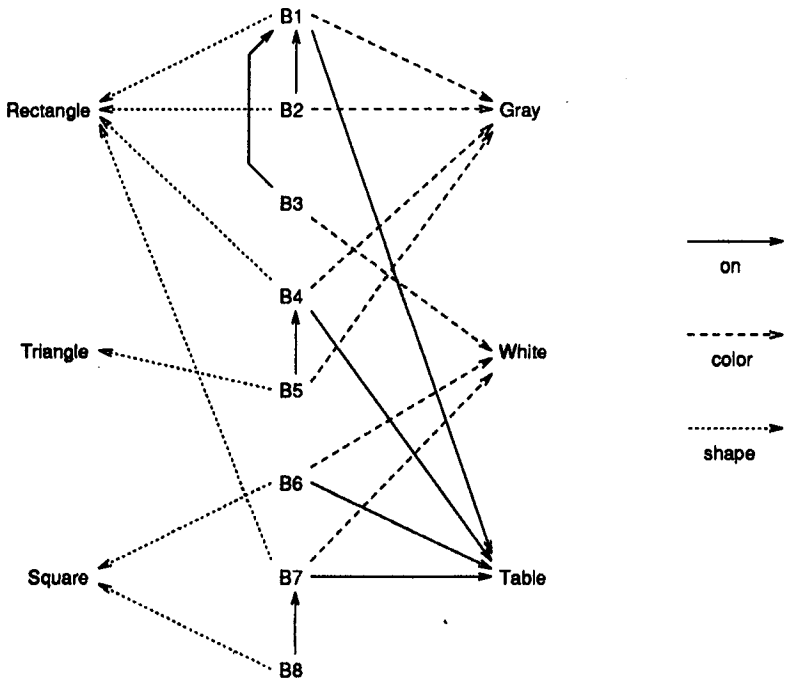

Figure 2: Network for Example.

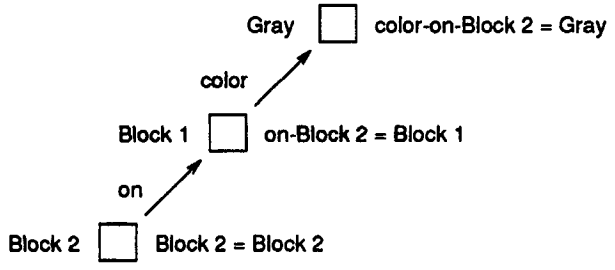

Figure 3: A Simple Example of Formula Propagation.

possibilities simultaneously. If we attach *initial formulas* to one or more nodes in the network, we can propagate them through the network according to a small set of rules to show the relationship of the original nodes to the other nodes in the network. The idea is illustrated in Figure 3. The initial formula Block 2 = Block 2 is attached to node Block 2. It crossed the Block 2 $\stackrel{\text{on}}{\rightarrow}$ Block 1 link to become the formula on-Block 2 = Block 1. Notice the new formula is true because the link tells us that Block 2 is on Block 1. Then, this new formula crosses the Block 1 $\stackrel{\text{color}}{\rightarrow}$ Gray link to become the formula color-on-Block 2 = Gray. That is, the Block 2 is on a block whose color is Gray.

This model can be extended to count occurrences of formulas, and hence record the frequency of relationships. Suppose that in Figure 2 we mark all the blocks in the concept (Block 2, Block 3, and Block 5) with the initial formula X = X, and the blocks in the complement with the initial formula Y = Y. As these formulas propagate through the network, certain formulas show up frequently on certain nodes. This reflects the prevalence

of particular relationships from nodes in the concept (or complement) to that node. Nodes can count the number of times each formula occurs to keep track of these relationships. In the example here, the node Gray will eventually have three occurrences of the formula color-on-X = Gray, and no occurrences of the formula color-on-Y = Gray. This shows that all three of the blocks in the concept have this relationship, and none of the blocks in the complement do, so we know that the formula color-on-X = Gray characterizes the concept. In other words, the concept consists of exactly those blocks that are on a gray block.

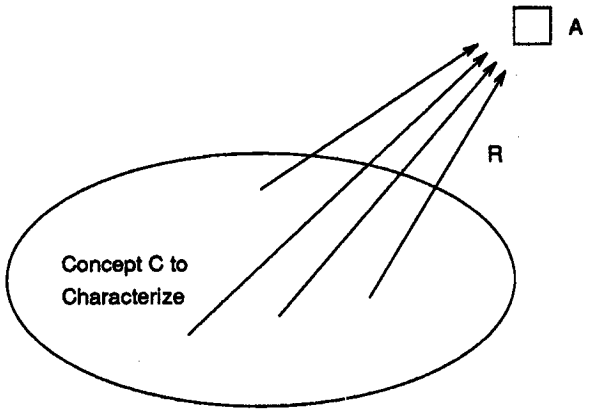

Figure 4: Characterizing a Class.

This method will work well if nearly all of the nodes of the concept are related to a single node in the same way, and few if any nodes of the complement have that same relationship. Figure 4 illustrates a set, C, that can be characterized by the rule:

$$[R(X) = A] \rightarrow C(X)$$

That is, the items in C (at least *most* of them) are connected to A by a sequence of links R. If we place the initial formula X = X on each item in C, they will propagate across the network and collect on the node A as (several copies of) the formula R(X) = A. The node A will be distinguished because it has so many formulas initially from the concept C. In fact, any relationship and node that characterizes the concept will have an accumulation of formulas.

Although potential characterizing relationships can be found in parallel, we still may need to search through combinations of them to find suitable rules. For instance, in the situation above the rule may also be true of many objects outside of the concept C. In that case we will need to search through additional conditions to exclude these negative examples. Alternatively, the relationship might not be true of all the concept, and we might need to search for additional formulas to characterize the remaining items of the concept. Consider the situation in Figure 5. It could be the case that neither R(x) = A nor S(x) = B characterizes the set, but their conjunction does.

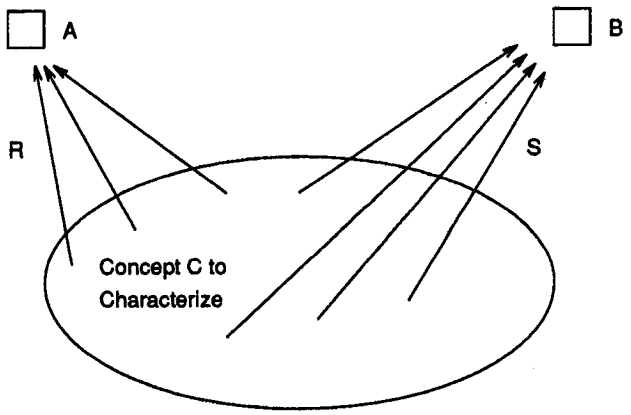

Figure 5: Characterizing a Class with a Conjunction.

3. A Prototype System.

We have built a prototype system based on the ideas presented above. The basic operation of our system is as follows:

1. Load the learning set and a knowledge base into an inheritance network. Then mark the nodes of the concept with *positive* initial formulas ($X = X$), mark the nodes of its complement with *negative* initial formulas ($Y = Y$), and allow them to propagate up through the network keeping track of accumulations.

2. Select significant formulas based on the proportion of positive and negative formulas that have accumulated. Briefly, if a formula has a large number of positive occurrences and few negative occurrences, it is a candidate to characterize some or all of the concept but little of the complement.

3. Use the selected formulas to create new features for the learning set (this can be done in parallel as described below), then run a feature-based learning system on the enhanced database (this can also be done in parallel, see [9]).

The basic ideas have been explained above; the rest of this section will describe some of the details of our implementation.

3.1. Parallel Implementation.

The CM-2 Connection Machine is a SIMD computer with thousands of processors that is ideally suited for implementing the formula-propagation model. Each node can be assigned to a single processor and instructions for propagating formulas across links can be issued by the front-end to all processors simultaneously.

The process of propagating formulas from the concept and its complement up through the network has been described above. One important problem surfaces when this is implemented on the CM-2: inheritance networks have irregular structure with branching factors

that can range from a few links to several hundred or thousand, but the CM-2 uses a regular interconnection scheme for data communications between processors. A bottleneck will result if many nodes try to send formulas to a single node at once (each node has at most 32 direct communication lines, but may have hundreds of links). Furthermore, each processor is a sequential processor and must process each incoming formula individually. We have solved this problem by using *auxiliary nodes* to spread a node's links, and hence its communication and processing load, across several processors. By building a tree of processors rooted at a node with many links, each leaf processor can process a portion of the original node's links, then combine the results through the tree. If the original node had n connections, this scheme introduces a $\log n$ factor, which is much better than the order n time required to process all links directly. This scheme also reduces the amount of memory required by nodes to store all of its links.

Once the formulas have been propagated, potentially useful ones need to be identified. We say a node is *significant* if it has some formula attached to it that has a higher proportion of positive formulas than the proportion of negative formulas (proportions computed relative to the total number of items in the concept or complement). These nodes can be identified and marked in parallel. More complex measures of significance are possible and further work is required to explore these.

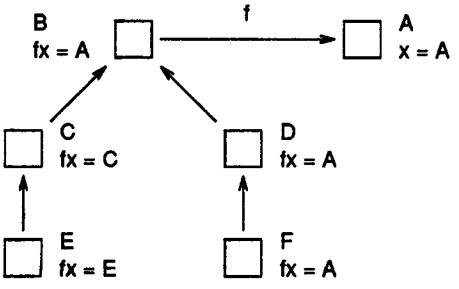

Figure 6: A Simple Downscan.

Given a significant formula FX = A on a node A we want to attach the new *feature* FX = A to every node of the learning set whose F is A, that is, every node whose initial formula propagated through a sequence of links F and isa links to end on the node A. This can be done in parallel by first attaching an initial formula to the node A and allowing it to propagate backwards across links. Nodes of the original learning set related to node A (both in the concept and its complement) will have the formula FX = A attached to them, and this can be used as a new feature. This is illustrated in Figure 6.

The new features are derived from nodes and relationships that seemed to be significant, according to a rough indicator based on the proportion of concept and complement nodes related to it. While there is no guarantee these new features will enhance the learning process, they will generally augment the features an inductive learner can operate with. In our prototype, after the new features are identified and attached to the learning set we use the RL sytem [2] to learn rules to characterize the concept.

3.2. Results on Scientific Data.

The system was run on a database of several hundred stellar spectra to learn classification rules for luminosity. A stellar spectrum is marked by several *lines* corresponding to ionized substances in the star's atmosphere. These lines have varying intensity and can be used to predict the star's luminosity.

When the RL system was given data on the stars and their spectral lines (whether they were *present* or *absent*) it produced the following four rules:

TiO line → High Luminosity

C_2 line → High Luminosity

OH line → High Luminosity

Ca^+ line → High Luminosity

That is, if a star's spectrum has a line for any of the molecules TiO, C_2, OH, or the ion Ca^+ it has High Luminosity.

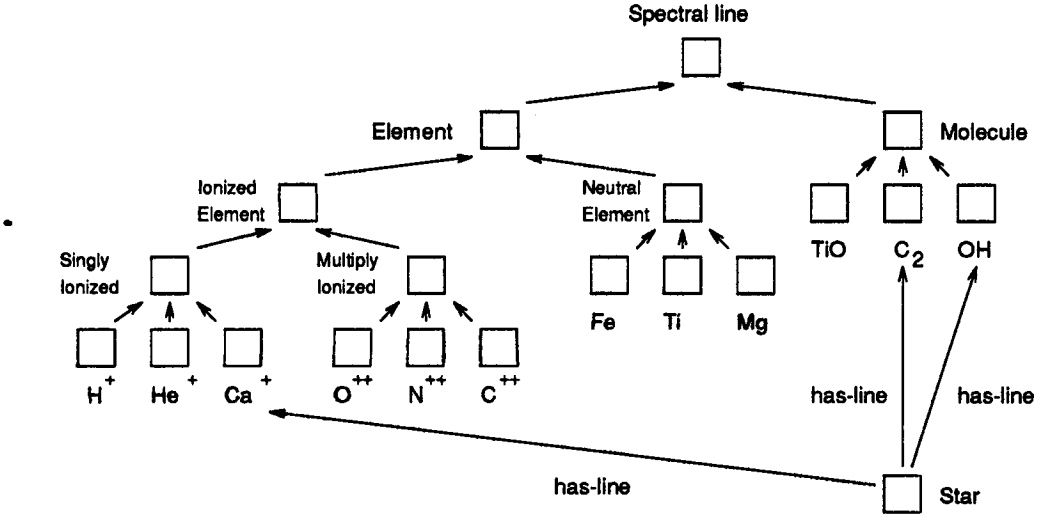

Figure 7: Spectral Lines Classification.

These rules are useful, and they match experts' classification of stars, but they do not capture connections and generalizations in the data and features being reasoned about. To capture and represent generalizations in learned rules requires representing and using basic scientific knowledge. We linked the stellar spectra to the knowledge base of elements and their characteristics shown in Figure 7. (In our experiments, data for several thousand stars were used, but only one is shown here for clarity.) By learning with both data and the general knowledge available in the inheritance network, the system learned two rules (instead of the previous four):

Molecule(has-line(x)) → High Luminosity(x)

Ca^+(has-line(x)) → High Luminosity(x)

The first rule states that if an item ("x") has a spectral line that satisfies the predicate Molecule, then that item satisfies the predicate High Luminosity. The second rule says that if an item has a line that satisfies the predicate Ca^+ then is satisfies the predicate High Luminosity.

These two rules can, in fact, be combined into a single rule if more knowledge is added to the knowledge base that represents an exceptional property of the element Calcium that makes it similar to molecules. But this would require representing and using nonmonotonic reasoning, which is currently beyong the capabilities of the system.

The system was also run on a database of 3.5 million infant births and deaths from the U.S. Department of Health. In the original data, births were categorized according to county, state, and region of the country where they occurred. These regions included areas such as New England, Middle Atlantic, Southeast, etc. In addition to the categories found in the original data from the Department of Health, we linked the records to other geographic categories including the new region "East Coast." When run on the combination of original data and the new domain knowledge the system was able to find a new rule that Asians living on the East Coast have an extremely low incidence of infant mortality. Note that a simple attribute-value hierarchy would not have discovered this rule, because East Coast is not one of the regions that would have been specified, and there was no *a priori* way to know that 'East Coast' would be a relevant category. However, by putting in several categories that might be relevant (into an efficient representation), this rule could be learned.

4. Capabilities and Limitations.

Since formulas are propagated along all paths in parallel, the complexity of finding significant nodes, and new terms for learning, is linear in the depth of the network. Because of this efficiency, and the fact that all paths can be explored simultaneously, the system does not have to choose which paths seem most promising. Other systems, such as FOIL, have to perform a heuristic search through a large space of terms. But such a system often cannot do sufficient look-ahead to know which branches of the search tree are most promising.

For instance, [11] gives the following example of a concept that FOIL cannot learn. The FOIL system is given the relations:

$$P = \{\langle 1\rangle, \langle 2\rangle\}$$

$$A = \{\langle 1, t\rangle, \langle 2, f\rangle, \langle 3, t\rangle, \langle 4, f\rangle\}$$

$$B = \{\langle 1, t\rangle, \langle 2, t\rangle, \langle 3, f\rangle, \langle 4, f\rangle\}$$

$$C = \{\langle 1, f\rangle, \langle 2, t\rangle, \langle 3, f\rangle, \langle 4, t\rangle\}$$

$$Q = \{\langle t \rangle\}$$

When attempting to learn the concept P, it must first select one of the relations A, B, or C. But with limited search and lookahead they all look equally promising, and FOIL chooses the wrong one, thus pruning the branch of the search tree leading to the correct solution. Using our system, (with relations represented in graph format) formulas are propagated across all links in parallel and the correct characterization of the concept is noticed immediately. [1]

The efficiency of the system stems from its limited description language based on links of an inheritance network. The inheritance system can find new terms of the form $f_1 \ldots f_3 x = y$ where x and y are nodes, and each f_i is a role link in the network. (There can also be isa links interspersed with the role links, but they will not appear in the final formula). Then, using RL, the system can assemble the new terms into Boolean combinations to characterize a concept.

Although this language is more restricted than some other systems, for example FOIL and GOLEM, it takes advantage of a close correspondence between databases and inheritance networks. Records (rows of relations) correspond to individuals, and columns correspond to relational links. For instance, the simple database with the relations shown in Figure 8 can be represented as the inheritance network shown in Figure 9. This simplified example does not show all the structure associated with the various attributes. For instance, BMW's may themselves have structure and be classified with additional isa links.

People		
Name	*Occupation*	*Automobile*
Bob	Accountant	BMW
John	Programmer	Spectrum
Sam	Bartender	Corvette
Suzie	Lawyer	BMW
Tim	Lawyer	BMW

Occupations	
Occupation	*Salary*
Accountant	High
Programmer	Medium
Bartender	Low
Lawyer	High

Figure 8: A Simple Relational Database.

Reasoning with only the disjoint tables of the original database, a learning program could only find that accountants and lawyers own BMW's. But a learning program based on formula propagation can use the inheritance network to learn that people with high salaries own BMW's. This would be done by first marking the node BMW, and then propagating formulas to mark those people who own BMW's. From there, the new formulas would be propagated through the network to find characteristics of those people. In this case, it would be those people who have jobs with high salaries. The rule that people with high salaries own BMW's is equivalent in this database to the rules that accountants and lawyers own BMW's. But the single rule intuitively is more explanatory, and in fact is more useful. Since inheritance networks represent generalizations (knowledge) as well as individual facts, using

[1] The concept is characterized by the predicate Q(B(x)), but since we represent the unary relation Q as the single-valued role link $\xrightarrow{Q}$ Q, we find the formula Q(B(x)) = Q

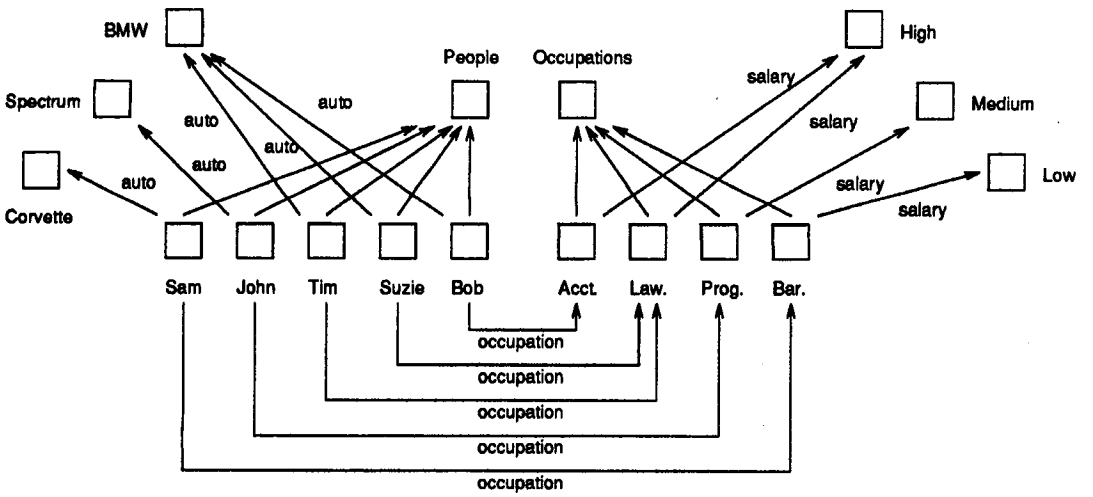

Figure 9: An Equivalent Inheritance Network.

them it is possible to generalize rules. Thus, you will get rules that are more robust as data is added.

This correspondence between databases and inheritance networks will allow our system to learn from the relations implicit in the structure of many databases. It will also open the possibility of integrating databases into knowledge bases to exploit generalities that are expressed as *knowledge*, rather than just data.

5. Related Work.

This work is closely related to other work on relational inductive learning, which has come to be known as "inductive logic programming" [11] [7] [8]. Our method differs from these methods in that we take advantage of the efficiencies offered by representing background knowledge as an inheritance hierarchy (with roles). Standard relational approaches (*e.g.*, FOIL [11] and GOLEM [7]) assume that background knowledge is expressed as a set of ground facts. For large amounts of background knowledge combined with large data sets, the number of ground facts needed can be immense. Therefore, these approaches will not scale up to very large problems. Inheritancy hierarchies give a compact and efficiently searchable representation for background knowledge.

We use techniques for finding relationships in inheritance hierarchies to suggest new terms to a propositional learner. Thus, our work is related to other work on feature construction (or "constructive induction"). Matheus gives a good overview of techniques for feature construction [5]. Most of this work does not address the use of a large body of domain knowledge for suggesting relational terms. One exception is the inductive logic programming system LINUS [8]. LINUS uses relational background knowledge to suggest new terms to standard feature-based inductive learners. However, LINUS enumerates all possible relational terms, based on syntactic and semantic constraints (such as type constraints). As with the relational approaches discussed above, such an exhaustive enumeration will not scale up to very

large problems (*e.g.*, problems with many examples and lots of irrelevant domain knowledge). We address this problem by using formula propagation techniques to find only terms that appear to be useful for characterizing the concept to be learned; these terms can be compositions of relationships from the hierarchy.

6. Discussion and conclusions.

The two real-world applications to which we have applied our prototype system do not take full advantage of the system's capability for learning relational terms. In each case, a system that allowed learning with attribute-value hierarchies alone (and not relational knowledge) could also have learned the more general rules. However, as the blocks-world example illustrates, more complicated relationships can be discovered. In addition, the computational effort involved scales to the complexity of the knowledge represented in the inheritance hierarchy. At the least-complex end of the spectrum (no background knowledge), the system degenerates into a standard feature-based learner. If attribute-value hierarchies are provided, the system can take advantage of them to learn more general rules. At the most-complex end, the system can take advantage of background knowledge that includes functional roles and relationships between examples.

In conclusion, we believe that inheritance hierarchies can represent background knowledge that can be useful for real-world learning problems. We have shown that inductive learning can take advantage of the efficiency of the inheritance hierarchy representation for augmenting the description language of a propositional learner. This is important when there is a large amount of (mostly irrelevant) background knowledge in addition to a large number of examples. In such domains, blindly compiling all background knowledge into propositional form (or the set of all possible ground facts) is infeasible. The inheritance hierarchy representation also facilitates parallelization. On a massively parallel machine, this greatly increases the space of relations that can be searched for relevant knowledge.

Most of the work on this project is future work. We hope to link large databases with large knowledge bases in such a way that learning is feasible. For example, infant mortality data could be linked with knowledge about industrial and environmental factors for different regions of the country. Considering that we are already dealing with over three million live births per year, efficiency is a major concern. In addition, we would like to improve the parallel inheritance hierarchy search to deal with more complex inheritance representations (including non-monotonicity).

References

[1] Aronis, J. (1993). *Implementing a theory of nonmonotonic inheritance on the connection machine.* Ph.D. Thesis, Intelligent systems Program, University of Pittsburgh.

[2] Clearwater, S., and Provost, F. (1990). RL4: A Tool for Knowledge-Based Induction. In *Proceedings of the Second International IEEE Conference on Tools for Artificial*

Intelligence, pp. 24-30. IEEE C.S. Press.

[3] Danyluk, A. P., and Provost, F. J. (1993). Small Disjuncts in Action: Learning to Diagnose Errors in the Telephone Network Local Loop. In *Proceedings of the Tenth International Conference on Machine Learning (ML-93)*. Morgan Kaufmann.

[4] Holte, R. C., Acker, L. E., and Porter, B. W. (1989). Concept Learning and the Problem of Small Disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 813-818. Morgan Kaufmann.

[5] Matheus, C. (1989). *Feature Construction: An Analytic Framework and an Application to Decision Trees*, Ph.D. Thesis. Department of Computer Science, University of Illinois at Urbana-Champaign.

[6] Michalski, R., Mozetic, I., Hong, J., and Lavrač, N. (1986). The Multi-purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 1041-1045. AAAI-Press.

[7] Muggleton, S., and Feng, C. (1990). Efficient Induction of Logic Programs. In *Proceedings of the First International Conference on Algorithmic Learning Theory*, pp. 369-381. Tokyo, Japan: Japanese Society for Artificial Intelligence.

[8] Lavrač N., and Džeroski S. (1994). *Inductive Logic Programming*. Ellis Horwood.

[9] Provost, F.J. and Aronis, J.M. (1994). Scaling Up Inductive Learning with Massive Parallelism. Intelligent Systems Laboratory Report ISL-94-3, Computer Science Department, University of Pittsburgh

[10] Quinlan, J. (1986). Induction of Decision Trees. *Machine Learning*, 1, pp. 81-106.

[11] Quinlan, J. R. (1990). Learning Logical Definitions from Relations. *Machine Learning*, 5(3), pp. 239-266.

[12] Rendell, L. and Seshu, R. (1993). Learning Hard Concepts Through Constructive Induction: Framework and Rationale. To appear in Computational Intelligence.