Theory Restructuring: Coarse-grained Integration of Strategies for Induction & Maintenance of Knowledge Bases

Edgar Sommer
Artificial Intelligence Research Division (FIT.KI)
GMD — German National Research Center for Information Technology
Schloss Birlinghoven, D–53757 Sankt Augustin, Germany
Fax +49(2241)14-2072, eddi@gmd.de

Abstract

Even after the publication of (Michalski & Tecuci 1994), the debate over what precisely constitutes *multistrategy* learning continues. While some authors have advocated a very fine-grained view of cooperation between relatively small learning operators, (Emde *et al.* 1993) argued that significant practical benefits can be obtained from a more coarse-grained interaction between different learning modules, each of which is complete by itself. I subscribe to this view, but add non-inductive functionality to establish a multistrategy context for the design and maintenance of knowledge bases.

This paper describes (the implementation of) a coarse-grained, multistrategy architecture for theory induction and maintenance, or *theory restructuring*, called RRT. The aim of theory restructuring, and RRT, is to close the gaps between knowledge engineering, knowledge acquisition and machine learning, in order to better support the design, and maintenance over time, of knowledge bases. To this end, a number of subtasks in the KB-maintenance process have been identified, namely

- **induction**: offering access to existing implementations of approaches to machine learning in the hope of automatically generating useful rules from exemplary data,
- **analysis**: offering insight into the status quo of a KB,
- **reorganization**: offering transformations that change the form and structure of a KB, whilst retaining the set of computed answers, and
- **evaluation**: offering a collection of simple, computable criteria that may help in assessing the suitability of different forms of a KB to different tasks.

Implementations of approaches to these tasks have been incorporated in RRT, but can be and are being augmented by new tools as these become available, so I prefer to view the current toolchest as an initial proof-of-concept collection — even though it works quite well already.

This paper gives an overview of the setup, describes the current tools in some detail, shows how third-party tools fit seamlessly into the coarse-grained, loosely coupled architecture, and then reports on empirical results in two non-toy problem domains.

Theory Restructuring: a Multistrategy Perspective on Maintenance of Knowledge Bases

A knowledge base (KB) is a formal model of some problem domain and consists, at the core and very generally, of facts and rules. The facts describe the problem domain, and the task of a knowledge based system (KBS) is to infer new facts on the basis of these given ones and the rules, and/or to answer queries about the validity of a new fact with yes or no.

Practical experience with designing such KBs (Sommer *et al.* 1994) has shown a need for addressing the task of *maintenance*, as well as the tasks addressed in the established fields of Knowledge Acquisition, Representation & Engineering, Machine Learning and Logic Programming. More succinctly, *reorganization and correcting knowledge bases is the most time-consuming phase during KBS development* (Carbonell 1991).

At various points in the life and design cycle of a knowledge base, the rules may be redundant, inconsistent and only partially cover the concepts [1] that represent the inferential goal, and it may be very difficult to gain insight into this status quo. Naturally, which rules are desirable and which are not depends on the application, but even on an abstract level, an analysis of a KB's status quo can be variously motivated:

- "Semantically" motivated: How good are the rules at solving their prime objective, covering the goal concept(s), that is, inferring new instances and/or answering queries? If coverage is not complete, can this failure be pinpointed and characterized? Are some rules redundant, in the basic sense that omitting them has no effect on the set of computable answers? Can this redundancy be characterized? Is the rule set inconsistent and can reasons for inconsistency be pinpointed?
- Pragmatically motivated: Is the model of the real problem domain as simple as it could be? Is it coherent and homogeneous not only in the sense of performance, but in

[1] I use the word *concept* in loose synonymy with the word *predicate* throughout this paper. Likewise, clause and rule are used more or less interchangeably.

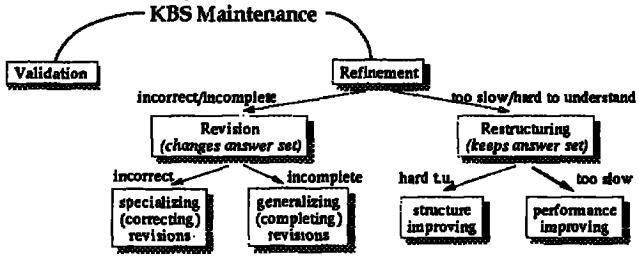

Figure 1: KBS maintenance (adapted from (Wrobel 1996))

the sense that similarities in the "real world" are reflected in similar structures in the model? More specifically, are valid and useful relationships in the real world *explicit* as concepts in the model? Are these concepts put to use consistently and throughout? Is the inferential goal reached by way of a chain of inferences with intermediate concepts (deep theory) or in a single, complex step (flat theory; see figures in the appendix)? The latter may be desirable in view of run-time optimization,[2] while the former may be easier to understand and modify.

- Externally motivated: Is the model adequate, i.e. correct with respect to the aspect of the world it is meant to represent?

From this perspective, the maintenance task can be divided into three interrelated topics: validation, revision, and restructuring (Figure 1). *Validation* is concerned with determining whether the knowledge base, as a formal model of reality, does in fact fulfill the purpose. (Meseguer 1992) gives a concise overview of this issue. *Revision* is concerned with changing the computed answer set of the KBS to fix some problem found in validation, for instance. (Wrobel 1994) gives a detailed treatment. *Restructuring* is concerned with changes to the knowledge base that do not alter the computed answers, but rather improve other criteria, such as understandability or execution time.

Theory maintenance: Insight, clean-up & reorganization

My main interest lies in the investigation and development of methods for maintaining knowledge bases of logical knowledge based systems, i.e. knowledge bases that can be interpreted as restricted first order logical theories.

Services associated with *insight* are meant to provide information on various aspects of the current state of the theory, ranging from simple statistical information and graphical presentations to such criteria as redundancy, utility and coverage of the rules in the theory, explanations of why specific inferences do or do not occur, and computation of criteria intended to help evaluate theory quality.

[2] As the "utility problem" (Minton 1988) in explanation-based learning and subsequent research on "speed-up learning" (Boström 1992; Subramanian & Hunter 1992; Clark & Holte 1992) have shown, this is only a rule of thumb, and often not the case in practice.

Clean-up involves acting on the prior analysis, such as elimination of unnecessary rules in theory and unnecessary conditions in individual rules, fixing non-generative rules, eliminating unused concepts and renaming concepts to better reflect their intended use.

Reorganization concerns the modification of a given theory's inferential structure without changing the answer set. Some examples of such modification are:

- Introduction of new concepts into the theory that allow a more concise re-expression of the rules. Motivation is twofold: finding meaningful and useful new concepts in a theory, and providing a deeper inferential structure of the theory, making it more modular, easier to understand and modify.
- Flattening of inferential structure by replacing intermediate concepts with their definitions where they occur in rules. This is the complement of the previous, and minimizes the number of inferences the KBS must perform at runtime.

Embedding Machine learning & Knowledge Acquisition

Restructuring is embedded in a context that combines Machine Learning, Knowledge Acquisition and Knowledge Engineering. Machine learning is concerned with developing algorithms capable of discovering new relationships — expressed as rules — between concepts in a given KB. In the context of KBS, these activities can also be viewed as forms of automatic analysis of existing knowledge, resulting in novel interpretations of existing data and/or the discovery of new relationships and structures in initially unstructured data (Sommer 1995c). More simply, ML offers an alternative to manual elicitation and formalization of rules from experts in cases where data rather than human expertise is more readily available.

Early work in ML was not concerned with this embedded aspect of learning algorithms, but rather with "pure" algorithms that take input and produce a result (a concept or rule) not necessarily expressed in the same language as the input, and not interpreted in any way by the algorithm itself (one-shot learning). More recent work (Morik *et al.* 1993) is taking ML in the direction of open systems, where learning results may function as input for subsequent learning passes (closed-loop learning), and where a uniform knowledge representation allows knowledge sharing between different components for differing purposes (such as knowledge browser, inference engine, explanation, revision and truth maintenance modules).

Work along these lines — using ML as a provider of rules for a KBS — has shown a need for more explicit and elaborate forms of evaluation and post-processing than either the areas of ML and Knowledge Acquisition have been concerned with. In ML, the main success criterion has been accuracy: what percentage of the given examples are covered by the induced theory, and in some cases, what percentage of probable future examples will be covered? This does not solve the task of combining the result of several algorithms. Other aspects of utility, such as understandabil-

ity, modularity, maintainability have not been in the main focus of ML research.

In Knowledge Acquisition, emphasis has been on manual construction of rules via elicitation, often in the absence of significant numbers of concrete examples. Here, the success criterion is based on experts judgment: does the expert agree with the rules that have been formulated by the knowledge engineer on the basis of the expert's utterances?

Theory restructuring aims at filling the gap between these two

- by allowing for various forms of inspection and evaluation of a KB,
- by offering ways of reorganizing a KB to make it more understandable, maintainable and more modular,
- by offering means of characterizing alternative, empirically equivalent forms of KBs,
- and by offering means of sifting a large number of rules produced by a number of competing ML algorithms (and knowledge engineers) for a most concise set of necessary rules.

RRT: an architecture for multistrategy theory induction & maintenance

This section describes the coarse-grained integration of diverse strategies aimed at offering the knowledge engineer a broad palette of services she may wish to take advantage of at different times in the prolonged process of designing and maintaining a knowledge base. RRT is implemented as one of a number of tools in the MOBAL system (Morik *et al.* 1993; Sommer *et al.* 1996). The following subsections briefly introduce the modules that currently make up the RRT architecture.[3] The selection of modules described here is intended to be representative only — a number of third-party induction algorithms can already be used from with RRT, and we are adding new ones, as well as new forms of restructuring as they become available. A final caveat: the analytical services that round off my perspective on KB maintenance (recall the introduction) are also omitted here, but described in (Sommer 1996b; 1995d; Sommer *et al.* 1996).

LINK: selective, incremental, flexible, "anytime" induction

LINK (Sommer 1994a) is a relational induction algorithm based on the idea of *describing* examples in terms of what is known about them (or, more precisely, about the objects that appear as arguments in the given examples), rather than defining a border between positive and negative examples, and making assumptions about what is not known.

LINK's attempt at being descriptive is accomplished, first, by viewing an incrementally extendable, but comparatively small, excerpt of a given knowledge base as a graph, and using heuristics to select a subgraph and treat it as a hypothesis. Second, a hypothesis is evaluated by computing the number of positive, negative, predicted and uncovered examples, and testing these values against two criteria that are parameters of the system. This allows LINK to accept hypotheses that cover some negative examples or predict some new ones (this method of testing hypotheses is due in large part to (Kietz & Wrobel 1991) and was first implemented in RDT).

Third, since the hypotheses are always created around individual examples, it is a simple matter to achieve a basic but effective type of incrementality: LINK can be told to use only uncovered examples in constructing hypotheses, so if it does find acceptable rules, these are guaranteed to provide a measurable contribution toward the goal of constructing a complete definition of the goal concept.

Outside of the context provided by RRT, LINK can be viewed as a multistrategy induction algorithm, in the sense that it combines aspects of three major approaches to induction:

- In the manner of bottom-up, least general generalization (LGG (Plotkin 1970)) algorithms, LINK generates sets of candidate premise literals by collecting facts about the terms that appear in goal examples — i.e. constructs *fact chains* to be used as starting clauses.
- Rather than using maximal fact chains as starting clauses, however, it treats each fact chain as a reservoir of potential premise literals, using a set of built-in, heuristic criteria to select amongst these candidates, in order to form hypothesis clauses (conjunctions of the selected literals, with the current example playing the role of head or conclusion literal). In other words, the decision about which literals to add to a clause is heuristically guided (although LINK does not climb hills).
- It uses additional, parameterized, structural criteria to select among possible hypotheses in the manner of declarative bias-based systems.

This may serve to illustrate the distinction between fine- and coarse-grained integration of strategies: LINK is an instance of fine-grained integration, where several strategies work symbiotically within one implementation,[4] while RRT is an instance of coarse-grained integration, where several autonomous implementations of strategies cooperate in a very loosely coupled manner — the advantage being that they need not be designed for a multistrategy context. In sum, this combination makes LINK especially suited to the restructuring context I am stipulating in this paper, by allowing it

[3] Since I wish to stress the cumulative effect of these tools in this paper, they are not described in detail. As it happens, all of them are described in (Sommer 1996b), along with the individual references given.

[4] LINK's predecessor INCY (Sommer 1993) took this tight integration one step further by treating the induced rules as rule models and feeding them into RDT as declarative bias. INCY's rules functioned as example rules, in a manner of speaking, which RDT then used for a more elaborate exploration of the hypothesis space. (Lübbe 1995; Kietz & Lübbe 1994) later implemented a much more intricate version of this approach to cooperation between inducers.

- to produce results without relying on complex declarative bias or negative examples,
- to work with sparse data,
- to produce generative rules,
- to produce more characteristic rather than discriminant definitions,
- to work incrementally, and
- to be flexible about which acceptance criteria for hypotheses it applies (e.g., allowing some negatives to be covered, or some new instances to be predicted).

XRA: pragmatic reduction of (sub-) theories

The topic of redundancy is a difficult one, mainly because no general-purpose definition exists. I have adopted the viewpoint defended in (Buntine 1988): "In the final analysis, a rule or some condition in the rule is redundant for a particular KBS if, on its removal, the system would continue to perform as required and still do so after any feasible extensions to the systems knowledge have been made." Nevertheless, most approaches to detecting and eliminating redundancy outside of ML have been of a more theoretical nature, including Buntine's own *generalized subsumption*, an extension of Plotkin's seminal subsumption algorithm (Plotkin 1970). By "theoretical" I mean that rather than analyzing (and taking advantage of) the target KBS's inferential capabilities, these are simulated by some idealized logical inference mechanism, such as SLD-resolution and subsumption. I call this approach **intensional**, because it detects redundancies on the basis of concepts' definitions, i.e. their intensions, as opposed to their *extensions*, the set of their instantiations.

In contrast, the **extensional** approach, which should be familiar to ML researchers since it is often included as a post-processing step in induction algorithms, makes decisions based on the known examples of a clause: GOLEM (Muggleton & Feng 1990), for instance, reduces an acceptable hypothesis by successively dropping literals from the premise and computing the coverage of the shorter clause. If it is still within bounds, the shorter clause replaces the original hypothesis, and this process continues until the shorter clause is deemed over-general. Conceptually similar post-processing functionality can be found in most TDIDT programs, such as the tree pruning offered by C4.5 (Quinlan 1992).

A second, orthogonal dichotomy is that of **clause** vs. **literal** redundancy: deciding whether an entire rule can be dropped without penalty, and deciding whether some condition(s) within a rule can go, turn out to be two different tasks, contrary to what one might expect. This said, XRA can be characterized as a collection of services offering extensional clause and literal reductions for restricted first order theories (no function symbols, a restriction inherited from MOBAL).[5] I believe in offering this functionality *outside* of individual algorithms for reasons which may be obvious if you have followed me this far:

- Not all induction programs perform this service on the clauses they induce.
- More importantly, an individual program is partially blindfolded in a number of ways:
  - It is not, in general, aware of the rules it may have induced in previous passes.
  - It is unaware of bigger picture, consisting of rules induced by other programs, as well as rules entered by hand.

In combination, this has bearing on the longer-term goal, which I stipulate in this paper, of *maintaining* a KB. We cannot expect arbitrary programs to take into account the status quo of the KB when proposing new rules, and certainly very few ML programs go back and re-evaluate the rules they produced last week on the basis of the current, changed state of affairs. So we require the services XRA offers in order to maintain a coherent, reduced version of the KB.

FENDER: reorganization through predicate invention

In a complex theory consisting of many long rules, we may ask if the theory can be re-expressed in a more understandable way, without losing the desired complete coverage. In the concept definition discussed in following results section, consisting of 30 rules with an average length of seven premise literal making reference to ten different concepts, there must be a lot of repetition in the conditions — such recurring conditions could be used to define new concepts, which, "folded"[6] into the originals, would reduce the size of the theory and add structure to the inferences of the target KBS. Moreover, the rules could be made simpler by "hiding" some of the variables that appear in the body, but not in the conclusion (non-head variables).

In (Sommer 1994b; 1995b) the notion of stratifying a given theory was introduced, essentially separating the inductive properties of inverse resolution (Muggleton & Buntine 1988) from its (re-) structuring properties. Following

[5] XRA offers a number of other services aimed at the knowledge engineer wishing to gain insight into this state of affairs: focussed excerpts of the KB in text and graphics windows, performance statistics, distinction of different types of concept instances (input, derived, contradictory, redundantly covered, uncovered), explanations of (un-) coveredness & redundancy, unused concepts, minimum required inputs for the inference of a specified goal (Sommer 1995d; Sommer *et al.* 1996).

[6] Folding and unfolding are basic transformations of logic programs investigated in the field of Logic Programming (not surprisingly), see e.g. (Bruynooghe *et al.* 1994). Unfolding should be familiar to most ML researchers, as it is the formal variant of what is done in explanation-based {learning|generalization}. Folding is the inverse operation, replacing a conjunction of literals in a rule premise with a single literal, that elsewhere in theory, is defined precisely as the consequence of that conjunction. From a Logic Programming perspective, constructive induction systems, as well as FENDER, perform "constructive folding", because the fold literal is being defined on-the-fly, rather than already existing in the theory.

a specific strategy, FENDER performs a number of inter- and intraconstruction steps on a given ruleset, restructuring by introducing new intermediate concepts, *retaining* the set of computed answers of the theory. In other words, FENDER performs the constructive part of constructive induction, leaving the induction part to some other algorithm. So, while the input to a (constructive) induction program consists of facts, FENDER's consists of rules already induced by another module (or entered by the knowledge engineer). The result is a new inferential structure that is deeper and more modular, and possibly easier to understand and maintain. The new intermediate concepts are intensionally defined and put to immediate use; they make implied relationships explicit by exploiting similarities and differences between original clauses of the theory. Figures 6 and 7 show the inferential structure of a flat theory, and corresponding deep theory using concepts invented by FENDER; see also the section on access policies below.

EVAL: criteria for communication about theories

(Sommer 1995a) (Sommer 1996a) motivated and defined some simple, computable criteria for the evaluation of theories. The primary purpose was to look beyond the established ML criteria concerning accuracy, and attempt to capture the more elusive quality of understandability; as such, the proposed criteria are similar in spirit to the intuitions expressed in (Bergadano *et al.* 1988), although quite different in detail. The criteria are summarized in Figure 2, and Figures 3, 4 and 5 make use of some in comparing the theories described in the next section.

Empirical results: Access Policies & Machine Shop Scheduling

Access policy in a telecom network

This subsection gives brief account of RRT being used in concert with the learning algorithms RDT and FOIL, applied to the SPEED domain (Sommer *et al.* 1994). The task is to find rules that correctly explain or derive the known instances of `may-operate(<User>, <Component>, <Operation>`, an instance of which specifies that `<User>` may apply `<Operation>` to `<Component>` in the distributed network of telecommunication routing components (examples below). The available background knowledge concerns various attributes of the employees, companies, switches and operations involved, and relationships between them, such as who owns or rents which components, who works for whom and in which department, etc. There are 26 different concepts in all and 1215 instances of the goal,

Induction We apply two different ML algorithms in the hope of "discovering" a policy in the distribution of permitted accesses, represented by the known instances of `may-operate`. The results of FOIL (Quinlan 1990) on this dataset are not entirely satisfactory: it produces a theory consisting of two rules that cover only 60% of the given instances, one of which is recursive (i.e. makes reference to `may-operate` in defining `may-operate`, which, at least, is not a very intuitive way of implementing an access policy).

Next, we apply RDT (Kietz & Wrobel 1991) to the same data. Here we get a large number of rules (30) that do however cover all instances[7]. One example of the 30 original non-redundant goal-concept rules:

```
   owner(X, Y)
& has-dept(Y, Z)
& manages(Z, X)
& works-in(U, Z)
& operator(U)
& optype(V, threshold-read)
→  may-operate(U, X, V).
```

It is to be understood as follows: "If an employee works in the department that manages a switch, and that department belongs to the company that owns the switch, and the employee has operator status, then the employee may perform operations of type `threshold-read` on that switch".

Analysis The theory induced by RDT provides two important types of insight into the underlying data. First, the rules explain how the known access rights can be derived from the other data (the basic description of the problem domain, which might be sourced from a GIS or DBMS). Second, of the 26 relations present in the data, 16 are not referred to in the body of any rule. This has obvious implications for data analysis and database design: if the target relations represent the goal of the database, i.e. the queries that are to answered by the system, then those relations found irrelevant need not be included in the data. In the present state of the theory induced by RDT, this means information about geographical locations, and about the employee relationship between users and companies (as opposed to departments) is irrelevant for determining who may do what to which component.

Understandability & Reorganization At this point we may ask the question posed in the section on FENDER above: can the current theory, consisting of 30 clauses with average length of seven premise literals, be re-expressed in a more understandable way, without losing the desired complete coverage? FENDER suggests one way of achieving this by introducing four new concepts into the theory. In the following, constructed predicates have been renamed for clarity. Note also that if they are deemed relevant and their definitions have been understood by the user, these new concepts and their user-supplied names make explicit valid relationships between objects of the problem domain; they add structure and useful terminology to the theory on the "knowledge level" as well as on the implementational level.

[7] RDT uses an entirely different approach than FOIL, based on additional declarative bias in the form of rule models, so that this should not be taken as a general indication of the relative merits of the two techniques; see (Sommer *et al.* 1994) and the original sources for details.

The first concept found by FENDER makes a relationship between components, departments and users explicit that is implicit in all of the original rules:

```
    has-dept(C, D)
& works-in(U, D)
& manages(D, S)
→ cu_manages(S, C, U)
```

Rewriting the original rules by using this new concept (folding) does not reduce the number of rules, but makes them simpler, both by shortening them and by eliminating any reference to departments, i.e. eliminating a non-head variable from the top-level rules. The second new concept makes use of the first, and is disjunctively defined:

```
  rented-by(S,C)
& works-for(U,C)
& cu_manages(S,C,U)
→ responsible(S,U)
```

```
  owner(S,C)
& works-for(U,C)
& cu_manages(S,C,U)
→ responsible(S,U)
```

Folding this into the rules reduces their number, since any occurrence of either the one or the other definition body is replaced, so that several pairs of rules may "melt" into one. Note also that the top-level rules are again shortened and any reference to companies is eliminated. The third and fourth concepts introduced, `manager_op(OP)` and `operator_op(OP)`, group the constants found in the original rules according to the context they appear in. This eliminates constants from the top-level rules and further reduces their number, so that finally, the 30 are subsumed by these two:

```
  responsible(S,U)
& manager_op(OP)
& manager(U)
→ may-operate(U,S,OP).
```

```
  responsible(S,U)
& operator_op(OP)
& operator(U)
→ may-operate(U,S,OP).
```

Figures 6 and 7 show the inferential structure of, respectively, the flat theory induced by RDT and corresponding deep theory as reorganized by FENDER.

Evaluation This definition of the goal concept is arguably easier to understand than the original one, as it contains far less rules than the original, and the individual rules are shorter and use fewer variables. A more detailed comparison and discussion of understandability issues is given in (Sommer 1996a); see also the figures in the appendix.

Next to providing insight into the data and making the induced theory more understandable, the concepts proposed during restructuring may also be useful in a database design context (Sommer 1995c).[8] Suppose the original database with 26 relations were an initial rough draft. After induction and restructuring, one might consider replacing those 26 with the 4 found by FENDER, i.e. modify the input format of the database to include only required information in a concise format. Notably, the fact that FENDER was able to fold both `rented-by` and `owner` into a single intermediate concept (`responsible`) means that this distinction need not be made in the future (although it may be required for purposes other than determining access privileges).

[8] Thanks to Luc deRaedt for pointing out this possibility.

Reducing individual rules The 30 rules induced by RDT are not clause redundant (recall the discussion in section on XRA above), in fact no instance is covered by more than one rule — even though RDT does not guarantee such results per se. They may however be literal redundant, and this may serve as brief illustration of extensional reduction's value in practice: As the rules have been induced from facts, no background theory relating concepts to one another is available, so that simple subsumption is the only applicable form of intensional analysis. No rule is subset of another, so intensional analysis offers no form of reducing the ruleset.

Extensional analysis, on the other hand, finds one or more premise literals in each rule that can be dropped without changing the rule's coverage, so that the average number of premise literals is reduced from seven to 5.5, and the size of the theory is reduced from 750 to 615 symbols. Moreover, one concept has been eliminated entirely from the list of required minimum inputs.

Allowing generalizations — accepting reduced versions of the rule that cover more examples than the original, as long as this remains within the confines of the overall set of known examples — has an even more marked effect: the average number of premise literals is reduced from seven to four, and the size of the theory is reduced from 750 to 240 symbols. The list of required minimum inputs is reduced from nine to five. Most significantly, the number of rules has been halved, from 30 to fifteen, because the deletion of literals has rendered pairs of rules equivalent. These results are summarized in Figure 3.

Reducing a set of rules As a brief illustration of the distinction between intensional and extensional reduction in practice, consider a version of SPEED containing a large number of rules from a variety of sources (RDT, GOLEM, LINK, FENDER, user input and a procedure that produces random permutations of rules). The rules are both literal-redundant and, since several algorithms have been applied independently, clause redundant.

Intensional reduction here is able to spot the rules that are random permutations of others, but not, for instance, rules that make use of synonymous concepts, such as those produced by repeated calls to FENDER. Extensional reduction is able to weed out all rules up to the two constructed by FENDER (above), thus attaining the best possible result. These results are summarized in Figure 4.

Relearning In a final set of experiments, we return to FOIL. Using the new concepts formed by FENDER, it is able to produce two rules that correctly cover all instances (they are almost identical to FENDER's reformulations above). Furthermore, in a larger dataset containing more operations and more than 7000 instances of `may-operate`, FOIL is also able to induce a correct theory using the new concepts, while it fails without them. This provides a second indication of the new concepts' utility. Note this could be a general-purpose way of proceeding: first using a excerpt of data together with an exhaustive-search algorithm like RDT, then letting FENDER invent highly informative

(in the information-gain sense) new concepts, and finally applying a faster, heuristically guided algorithm such as FOIL to the larger dataset, including the new concepts. (Wnek 1993) reports on successful applications of a similar strategy, "hypothesis-driven constructive induction", in attribute/value domains.

Machine Shop Scheduling

Third-party work This is work in progress conducted principally by Wolfgang Ewert and colleagues at the Technical University of Chemnitz, Germany (Ewert, Dürr, & Oley 1994; Dürr, Ewert, & Leidhold 1995).

Induction In work towards her Master' thesis (Zöller 1995), Gina Zoeller describes the use of RRT in restructuring a theory governing the sequence in which form elements are processed in the construction of workpieces. In prior steps, definitions were induced by RDT for three concepts stating whether elements are to be processed in direct succession, simultaneously (an important concept in scheduling, as it means the elements must mounted on the vise together) or in general succession (allowing other elements to be handled in between).

Reduction XRA was able to significantly reduce the size of these (sub)theories, e.g. from 183 to 14 and 267 to 16 rules in different versions of the domain (Zöller 1995, p. 50). Dropping redundant literals from the remaining rules reduced their average premise length from 3 literals to 2.

Reorganization FENDER was applied to these reduced theories and was able to suggest several intermediate concepts reflecting recurring conditions in the original rules. As some of these where disjunctively defined (and deemed useful by the domain expert), this led to further reductions in the number of rules needed. It also "discovered" a concept describing situations in which several pairs of elements are to be processed in one production step.

In summary, RRT was judged useful both in (Zöller 1995) and when applied to previous work in the machine shop scheduling domain (Oley 1994).

Continuing work The machine shop scheduling problem looks to be a very promising application for the coarse-grained integration of strategies in RRT. Although the work is very much in progress at the time of this writing, I'd like to list some the results attained more recently:

Incremental induction In a domain with some 2500 instances of a binary ordering relation specifying which form element should by processed before which in the construction of a complex machine tool, LINK was able to induce a partial definition, covering approx. 90% with no false classifications, but some predictions we are not sure about at this time — i.e. the theory may be slightly over-general in practice. This definition initially consisted of well over 300 clauses, and the induction time was considerable: all in all, over 24 hours processor time on a Sparc IPX was necessary. LINK "built-in incrementality" came in very handy here, as we were forced to abort several learning passes — when we restarted, LINK was able to take into account the clauses induced in previous passes, and since it can be made to use only uncovered instances as seeds for the construction of new hypotheses, the definition grew in a truly useful incremental manner.

Wolfgang Ewert had previously applied FOIL, GOLEM and RDT to the same domain. Even with hand-crafted negative examples (with hundreds of form elements, a blind closed-world assumption would produce prohibitive megabytes of negative examples) neither of the first two were able to produce good results — note that LINK does not need negative examples in general, and did not use any here. RDT did produce some promising results, but only with the help of some hand-crafted rule models (RDT's primary form of declarative bias, higher order constructs in which predicate symbols are variables and must be instantiated with predicate names from the KB at hand) produced by Wolfgang Ewert in the course of a year of trail and error, whereas LINK produced results immediately, from the "bare facts" only.

Reduction Next, the 300-plus clause definition was reduced down to approximately 100 clauses using XRA's extensional clause and literal reduction facilities. RRT offers a complementary module implementing some intensional reductions (Plotkin and Buntine in the section on XRA above), but this offers little help in practice: in the presence of representative examples, the "redundancies" discovered intensionally are always a subset of the extensional ones. There is a basic tradeoff between the two — extensional analysis may determine "too many" redundancies, which may become invalidated over time in unfavorable cases, while intensional analysis may discover too few, that, however, are guaranteed to be valid across monotonic growth of the KB. This is due to an *intensional completeness assumption* that all valid relationships between concepts in a KB be explicitly note in the form of rules — without such rules, redundancy can only be detected extensionally. This dependency is especially relevant when rules are learned from facts only: since the concepts are defined by enumeration only, intensional analysis has nothing to work with (except the simplest subsumption relation between the induced clauses themselves) (Sommer 1996b, Ch. 3).

Accordingly, only a handful of rules were found to be intensionally redundant here, but a number of rules with relatively low coverage were made unnecessary by rules LINK induced in one of its subsequent passes (obviously, this is *a posteriori* analysis, and another indication of the power of extensional reductions in the task of consolidating the results of several passes and/or several induction algorithms). Also, some of the premise literals were found to be extensionally redundant (dropping them had no adverse effect on rules' coverage), so the average length of rules was also reduced, along with the number of rules.

Reorganization Finally, the reduced theory was fed into FENDER, which came up with about ten new intermediate concepts. Folded into the reduced theory, they reduced the size by another significant factor. Wolfgang Ewert is currently trying to talk the local engineering experts in Chemnitz into judging these intermediate concepts for real-world relevance. Since real-world expertise is a well-known bottle-neck in our game, we are also rerunning the experiments with FOIL, GOLEM and LINK, to see if the new concepts provide additional help in the induction process, as they did in the SPEED domain above.

Conclusion

I hope to have illustrated the benefits of adopting my perspective on KB maintenance, that of *theory restructuring*. By offering the services of induction, analysis, evaluation and reorganization under a coherent roof, the overall functionality of the resulting multistrategy system exceeds the net profit of using the tools individually, in a number of ways:

- Induction can be used as a form of data analysis for unfamiliar domains, both in the obvious vein of proposing new or alternate definitions for goal concepts, and in the less obvious one of offering insights into relevance and irrelevance of specific concepts.
- Incremental induction can be used to complete incomplete definitions, and update the theory as new cases become known.
- Predicate invention on the basis of induced and expert-elicited theories
  - offers automatic delivery of novel perspectives on the current KB by producing alternatives that are more concise, modular, arguably easier to understand, make recurring conditions explicit, and yet are empirically equivalent to the original;
  - adds further insight into the possibilities of designing more concise representations or database schemas;
  - may even provide added help for subsequent induction (recall "Relearning" in the section on access policy above).
- Stand–alone reduction, as opposed to post-processing steps provided by individual inducers, can produce a concise, coherent theory from large collections of rules from a variety of sources, and can detect redundancies that occur over time, and between competing sub-theories, which individual induces are blind to.
- Simple, computable evaluation criteria allow comparison of alternative, empirically equivalent theories, and may allow cooperation and competition between inducers working on the same set of tasks.

The loose coupling of diverse strategies exemplified by LINK (induction), XRA (reduction), FENDER (reorganization through predicate invention) and EVAL (evaluation) put the power of introspection, incremental reduction, modularity and dynamic restructuring at the knowledge engineer's fingertips.

Current Work Although I am sceptical about the ultimate goals of so-call "hard AI", promising experiences with the present system have motivated us to look into the possibilities of adding more autonomy to the services provided by RRT. We are currently working on a process envelope to RRT that would result in a more autonomous, agent-like system able to accept high-level goal specifications and perform corresponding experiments on its own. The central idea is a simplified stock market metaphor, in which brokers mediate between an arbitrary number of autonomous agent processes that either compete on one and the same subtask or cooperate by addressing different aspects of a larger conglomerate task. The brokers would communicate in a language based on the criteria implemented in EVAL,[9] coordinating their clients' (e.g. FOIL, GOLEM, LINK, XRA, FENDER) induction and maintenance activities via a traditional blackboard scheme, the advantage being that these clients need not be designed for a distributed setting — from their point of view, they are performing precisely the services they were designed for.

While RRT represents a coarse-grained integration of strategies, this new system would be a more finely integrated multistrategy system from the user's point of view. In RRT, the functionality of the tools is put at her fingertips, but she retains control. In the new system, her intuitions about which tool might yield interesting results in the next step would have to be replaced by explicitly implemented control strategies based on the current status quo in the theory, as reflected in the numbers computed by EVAL. In the vein of empowering rather than automating (Winograd & Flores 1986), it seems to me one cannot hope to replace domain experts' intuitions with hard-coded strategies, although one may hope to compensate with computational brute force, in the manner of chess playing systems. However, while in chess success is determined by whether or not the program wins, when relying on a KBS in practice, we will be more apt to understand and trust a theory whose evolution we ourselves have guided than one issued from a black box.

Acknowledgments This work was partially funded by ESPRIT project "Inductive Logic Programming" (ILP). I thank Werner Emde, Stefan Wrobel, Dietrich Wettscherek, Jörg-Uwe Kietz, and the other members of the ML groups at GMD and Dortmund University for valuable discussions. Thanks is due to Ross Quinlan and colleagues (FOIL5) and Stephen Muggleton (GOLEM) for allowing their programs to be incorporated in MOBAL, which is available free of charge and liability for academic purposes via `http://nathan.gmd.de/projects/ml/home.html`.

[9] Thanks to Donato Malerba of Bari University, Italy, for suggesting something like this.

Criterion	Rationale
$\|R_{goal}\|$	# rules about goal concept; less is more
$Ratio = \frac{\|Pos_{goal}\|}{\|R_{goal}\|}$	relation between # instances & # rules; more is better
$Av.Coverage = \frac{\sum_{i=1}^{n} \|cov(r_i)\|}{\|R_{goal}\|}$	av. # instances covered by one rule; more is better
$RedundancyIndex = 1 - \frac{Ratio}{Av.Coverage}$	normalized ratio between the two above; a value > 0 indicates instances are covered by more than one rule
$Size(R_{goal})$	# symbols required to represent theory
$Compression = \frac{Size(R_{goal})}{Size(Pos_{goal})}$	rel. between above & # symbols required to represent the instances; a value > 1 indicates remembering the instances is "cheaper" than remembering the theory
Inference Depth	max. # of inference steps required to compute an answer;
Min. Req. Inputs	# concepts used in rules of the theory; restructuring tranformations such as un/folding & reduction influence this; rules may reference concepts "unnecessarily"
Av. Length of clauses	av. # of premise literals in the rules of the theory; a rough approximation of the complexity of indiviual rules
Av. #Vars per clause	# variables in rules' premises; another rough approximation of the complexity of indiviual rules
Av. #NonHead Vars per clause	# variables appearing in premise, but not in conclusion; these "clutter" rules unnecessarily & might be avoided by better representation design (intermediate concepts that "suppress" non-head variables)
Av. #Constants per clause	constants in rules might be considered an indication of over-specific rules and/or imperfect representation design (could be avoided with intermediate concepts that "suppress" constants)

Figure 2: Summary of EVAL's theory evaluation criteria (Sommer 1996a)

	Original		Reduce		Allow Generalizations
#Rules	30	⇝	30	⇝	15
Inst/RulesRatio	40.5	⇝	40.5	⇝	81
TheorySize	750	⇝	615	⇝	240
Compression	0.154	⇝	0.126	⇝	0.049
Av.Length	7	⇝	5.5	⇝	4
Av.Vars	5	⇝	5	⇝	4
Av.NonHead	2	⇝	2	⇝	1
Req.InputConcepts	9	⇝	8	⇝	5

Figure 3: Reducing individual rules (Literal redundancy). See "Reducing individual rules" in the section on access policies.

	Original		Intensional		Extensional
#Rules	260	⇝	109	⇝	2
Inst/RulesRatio	4.67	⇝	11.14	⇝	607.5
Av.Coverage	60.75	⇝	86.2	⇝	607.5
RedundancyIndex	0.92	⇝	0.87	⇝	0.0
TheorySize	5605	⇝	1844	⇝	22
Compression	1.153	⇝	0.379	⇝	0.004
Av.Length	5.76	⇝	4.1	⇝	3
Av.Vars	4.67	⇝	4.23	⇝	3
Av.NonHead	1.67	⇝	1.23	⇝	0
Av.Constants	0.98	⇝	0.96	⇝	0
Req.InputConcepts	9	⇝	9	⇝	5

Figure 4: Intensional and extensional reduction (Clause redundancy). See "Reducing a set of rules" in the section on access policies.

Evaluation Criteria	**Original**		FENDER
$\|R_{goal}\|$ (# rules about goal concept)	30	⇝	2 (20)
$Ratio = \frac{\|Pos_{goal}\|}{\|R_{goal}\|}$	40.5	⇝	607.5
$Av.Coverage = \frac{\sum_{i=1}^{n} \|cov(r_i)\|}{\|R_{goal}\|}$	40.5	⇝	607.5
$RedundancyIndex = 1 - \frac{Ratio}{Av.Coverage}$	0.0	⇝	0.0
Syntactic Size of R_{goal} : $Size(R_{goal})$	750	⇝	22 (151)
$Compression = \frac{Size(R_{goal})}{Size(Pos_{goal})}$	0.154	⇝	0.004 (0.031)
Inference Depth	1	⇝	3
Av. Length of clauses	7.0	⇝	3 (1.50)
Av. #Vars per clause	5.0	⇝	3 (1.55)
Av. #NonHead Vars per clause	2.0	⇝	0 (0.15)
Av. #Constants per clause	1.0	⇝	0 (0.75)

Figure 5: Comparison of original (induced by RDT) and restructured (FENDER) versions of SPEED with 1215 example instances of `may-operate`. The criteria are motivated and defined in (Sommer 1996a). Briefly, however:

- Values for the *required theory* are in parentheses: in a deep theory using intermediate concepts in inferences, it makes sense to investigate the values both for the *top-level* theory (the set of rules defining the goal concept) and the *required theory*, which is the union of the former and the set of rules defining the intermediate concepts. In a flat theory, the two are identical, since there are no intermediate concepts.
- The syntactic size of formulae is determined by counting the number of symbols (as in (Muggleton & Buntine 1988), for example.
- $cov(r_i)$ is the set of instances covered by rule (clause) $r_i \in R_{goal}$. Via $Av.Coverage$, this is used in $RedundancyIndex$, which represents as simple approximation of a given (sub-) theory's redundancy. The closer this value is to 1, the more "redundant" the theory is, in the following sense: ideally, each instance should be covered by only one rule in the theory; if this is the case, then $Ratio = AvCov$ and consequently $RedundancyIndex = 0$. The more instances are multiply covered, the larger $Av.Coverage$'s value, while $Ratio$'s remains constant, so that $Red \rightarrow 1$. Note that in the example above, the original ruleset is not redundant, so the restructured one isn't either.

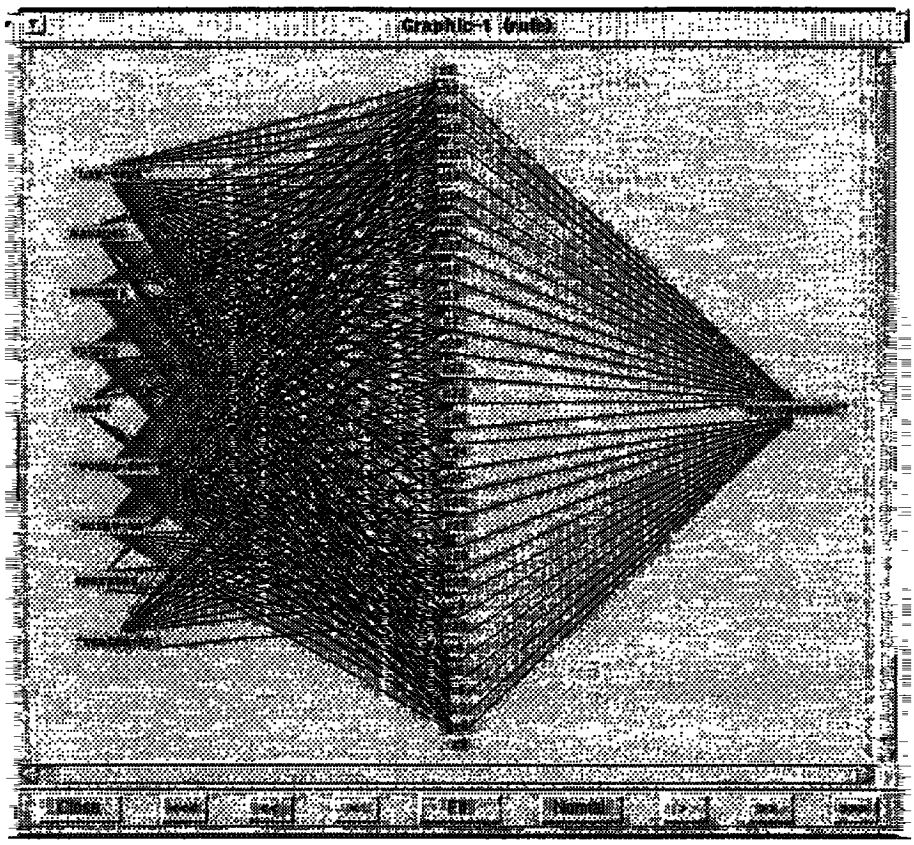

Figure 6: Inferential structure of (flat) SPEED theory induced by RDT, consisting of 30 rules (see also Figure 5). Left to right: **input concepts** used in rule premises → top-level **rule layer** → **goal concept**.

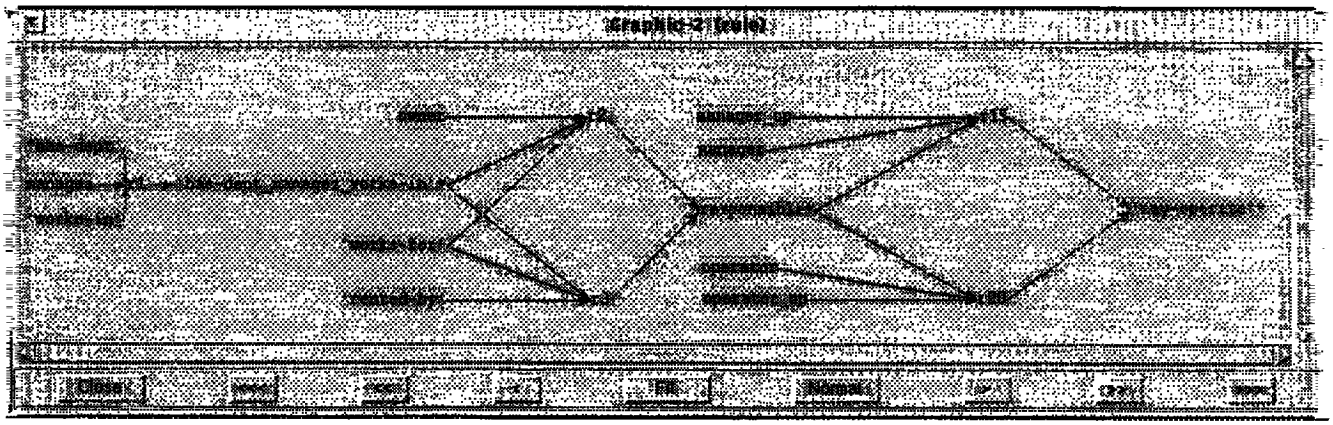

Figure 7: Inferential structure after reorganization by FENDER (inference depth 3, see also Figure 5). Left to right: input concepts → intermediate-level 1 rules → intermediate concepts 1 → intermediate-level rules 2 → intermediate concepts 2 → top-level rules → goal concept)

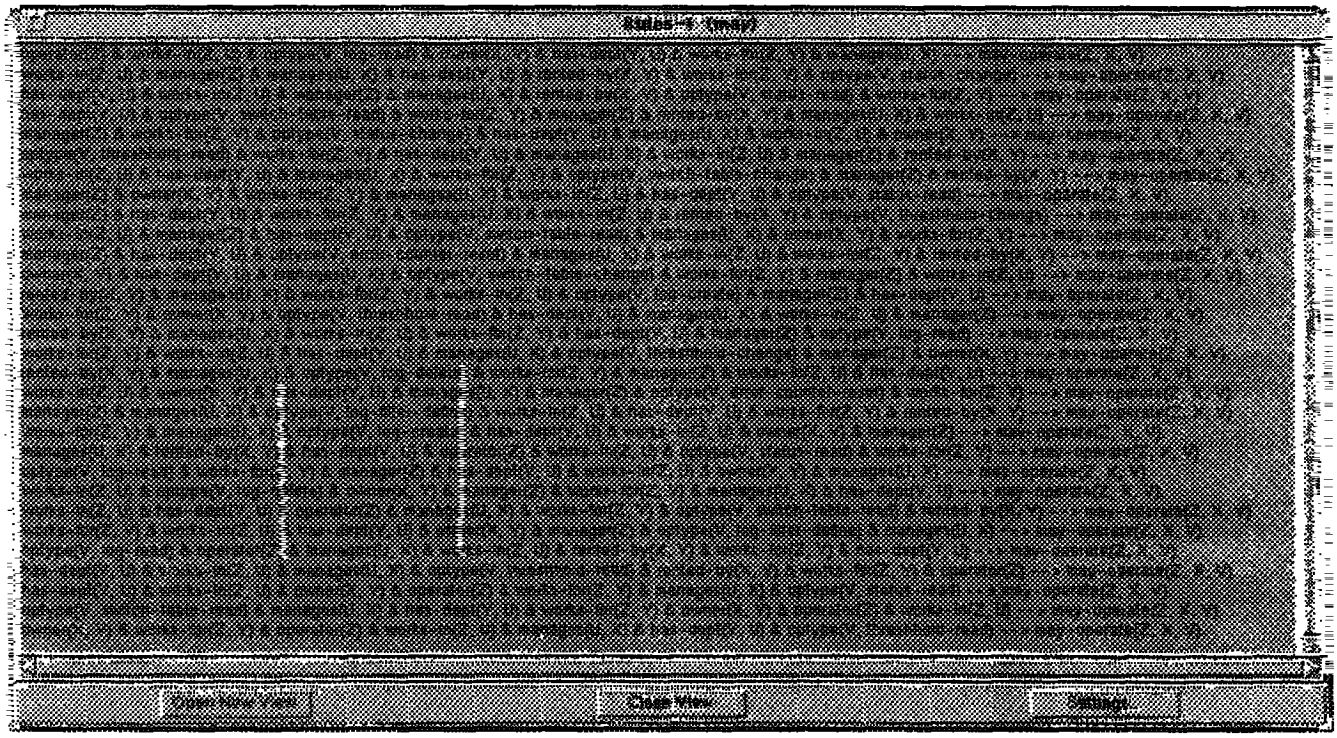

Figure 8: Flat SPEED theory induced by RDT. The corresponding reorganized version consists of only two rules plus definitions of the concepts invented by FENDER (see "Understandability & Reorganization" in the section on access policies).

References

Bergadano, F.; Matwin, S.; Michalski, R.; and Zhang, J. 1988. Measuring quality of concept descriptions. In Sleeman, D., ed., *Third European Working Session on Learning, Glasgow*. London: Pitman Publishing. Congressional Quarterly Almanac, Volume XLII, 1986.

Boström, H. 1992. Eliminating redundancy in explanation-based learning. In Sleeman, D., and Edwards, P., eds., *ML92 Proc. 9th Intl. Conf. on Machine Learning*, 37–42. Morgan Kaufman.

Bruynooghe, M.; Debray, S.; Hermeneglio, M.; and Maher, M., eds. 1994. *Journal of Logic Programming: Special Issue Ten Years of Logic Programming*, volume 19/20. New York: Elsevier Science.

Buntine, W. 1988. Generalized subsumption and its applications to induction and redundancy. *Artificial Intelligence* 36:149–176.

Carbonell, J. G. 1991. Scaling up kbs via machine learning. In Kodratoff, Y., ed., *Proc. Fifth European Working Session on Learning (EWSL-91)*. Springer.

Clark, P., and Holte, R. 1992. Lazy partial evaluation: An integration of explanation-based generalization and partial evaluation. In *Proc. Ninth Intern. Workshop on Machine Learning*. Morgan Kaufman.

Dürr, H.; Ewert, W.; and Leidhold, F. 1995. Machine learning in generative process planning. In Morik, K., and Herrmann, J., eds., *Proceedings FGML–95 (German ML Workshop)*, number 580 in FB Infortmatik Research Reports.

Emde, W.; Kietz, J. U.; Sommer, E.; and Wrobel, S. 1993. Cooperation between internal and external learning modules in mobal: different facets of multistrategy learning. In Saitta, L., ed., *Proc. of the First MLNet Workshop on Multi-Strategy Learning*. Available via WWW `http://nathan.gmd.de/projects/-ml/lit/mlpublist.html`.

Ewert, W.; Dürr, H.; and Oley, G. 1994. Learning in first order logic in a mechanical engineering planning domain. Unpublished.

Kietz, J.-U., and Lübbe, M. 1994. An efficient subsumption algorithm for inductive logic programming. In *Proc. Eleventh International Conference on Machine Learning (ML-94)*.

Kietz, J.-U., and Wrobel, S. 1991. Controlling the complexity of learning in logic through syntactic and task-oriented models. In Muggleton, S., ed., *Proc. 1st Int. Workshop on ILP*, 107 – 126. Also in S.Muggleton (ed.), Inductive Logic Programming, Academic Press, 1992.

Lübbe, M. 1995. Datengesteuertes lernen von syntaktischen einschränkungen des hypothesenraumes für modellbasiertes lernen. Master's thesis, Fachbereich Informatik der Universität Dortmund, Dortmund, Germany. (in German; available as Tech. Report LS-8/15).

Meseguer, P. 1992. Towards a conceptual framework for expert system validation. *AI Communications* 5(3):119–135.

Michalski, R., and Tecuci, G., eds. 1994. *Machine Learning: A Multistrategy Approach, Vol. IV*. San Mateo, CA: Morgan Kaufmann.

Minton, S. 1988. Quantitative results concerning the utility of explanation-based learning. In *Proc. 7th National Conference on Artificial Intelligence*, 564–569. Morgan Kaufman.

Morik, K.; Wrobel, S.; Kietz, J.-U.; and Emde, W. 1993. *Knowledge Acquisition and Machine Learning*. London: Academic Press.

Muggleton, S., and Buntine, W. 1988. Machine invention of first-order predicates by inverting resolution. In *Proc. Fifth Intern. Conf. on Machine Learning*. San Mateo, CA: Morgan Kaufman.

Muggleton, S., and Feng, C. 1990. Efficient induction of logic programs. In *Proc. First Conf. on Algorithmic Learning Theory*. Tokyo: Ohmsha Publishers.

Oley, G. 1994. Akquisistion von strategie- und planungswissen mit dem system mobal. Master's thesis, Fakultät Informatik, Technische Universität Chemnitz. (in German).

Plotkin, G. D. 1970. A note on inductive generalization. In Meltzer, B., and Michie, D., eds., *Machine Intelligence*, volume 5. American Elsevier. chapter 8, 153–163.

Quinlan, J. R. 1990. Learning logical definitions from relations. *Machine Learning* 5(3):239 – 266.

Quinlan, J. R. 1992. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Sommer, E.; Morik, K.; Andre, J.; and Uszynski, M. 1994. What on-line learning can do for knowledge acquisition. *Knowledge Acquisition* 6:435–460.

Sommer, E.; Emde, W.; Kietz, J.-U.; and Wrobel, S. 1996. Mobal 42 user guide ((always) draft). Arbeitspapiere der gmd, GMD. Available via WWW `http://nathan.gmd.de/-projects/ml/lit/mlpublist.html`.

Sommer, E. 1993. Cooperation of data-driven and model-based induction methods for relational learning. In Michalski, R., and Tecuci, G., eds., *Second International Workshop on Multistrategy Learning*, 180–187. Available via WWW `http://nathan.-gmd.de/projects/ml/lit/mlpublist.html`.

Sommer, E. 1994a. Learning relations without closing the world. In *Proc. of the European Conference on Machine Learning (ECML-94)*. Berlin: Springer-Verlag.

Sommer, E. 1994b. Rulebase stratification: an approach to theory restructuring. In *Proc. 4th Intl. Workshop on Inductive Logic Programming (ILP-94)*. Available via WWW `http://nathan.-gmd.de/projects/ml/lit/mlpublist.html`.

Sommer, E. 1995a. An approach to quantifying the quality of induced theories. In Nedellec, C., ed., *Proc. IJCAI Workshop on Machine Learning and Comprehensibility*. Available via WWW `http://nathan.gmd.de/projects/-ml/lit/mlpublist.html`.

Sommer, E. 1995b. Fender: An approach to theory restructuring. In Wrobel, S., and Lavrac, N., eds., *Proc. of the European Conference on Machine Learning (ECML-95)*, volume 912 of *Lecture Notes in Artificial Intelligence*. Berlin: Springer-Verlag.

Sommer, E. 1995c. Induction, evaluation, restructuring: Data analysis as a machine learning loop. In Lasker, G. E., ed., *Proc. of the Conference on Intelligent Data Analysis (IDA-95)*.

Sommer, E. 1995d. Mobal's theory restructuring tool rrt. Technical report, ESPRIT Project ILP (6020). ILP Deliverable GMD 2.2.

Sommer, E. 1996a. An approach to measuring theory quality. In Shadbolt, N., ed., *Proc. European Workshop on Knowledge Acquisition (EKAW-96)*. Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag.

Sommer, E. 1996b. *Theory Restructuring*. NN. (submitted).

Subramanian, D., and Hunter, S. 1992. Measuring utility and the design of provably good ebl algorithms. In *Proc. Ninth Intern. Workshop on Machine Learning*. Morgan Kaufman.

Winograd, T., and Flores, F. 1986. *Understanding computers and cognition: A new foundation of design*. Norwood, NJ: Ablex.

Wnek, J. 1993. *Hypothesis-driven Constructive Induction*. Ph.D. Dissertation, George Mason University, Fairfax VA.

Wrobel, S. 1994. *Concept Formation and Knowledge Revision*. Dordrecht, Netherlands: Kluwer Academic Publishers.

Wrobel, S. 1996. First order theory refinement. In Raedt, L. D., ed., *Advances in ILP*. Amsterdam: IOS Press. ftp://ftp.gmd.de/ml-archive/GMD/papers/ML73.ps.gz.

Zöller, G. 1995. Strukturierung and systematisierung einer wissensbasis f r das lernen von planungswissen mit dem system mobal. Master's thesis, Fakultät Informatik, Technische Universität Chemnitz. (in German).