Prague University of Economics

Faculty of informatics and statistics

Learning from observational data with prior knowledge

Doctoral dissertation

Author: **Vojtěch Svátek**
Supervisor: **Jiří Iványek**

Praha, 1997

æ

Preface

This thesis is the main result of research carried out during 1991-1996, at the Department of Information and Knowledge Engineering, Prague University of Economics.

The aim of the work was twofold. First, I wanted to analyze the general idea of learning (or, knowledge base construction) using two sources of information - *prior knowledge* (of various types) and *data*, and to outline a (necessarily subjective) framework for describing different approaches to solving this task, which have rarely been examined together before. Second, I wanted to apply some of the outcomes of this methodological work in the design of new learning algorithms. At my current workplace, there is a long tradition of research on learning from observational data, from the first, theoretical, outlines of the ESOD method, up to the newest variations of the KEX knowledge engineering toolbox. So far, however, these methods have been oriented to processing categorial data without recourse to other sources of knowledge. It seemed therefore to be an attractive task, to couple the existing learning paradigm of ESOD with the state-of-the-art of *learning with prior knowledge*, including some new ingredients proposed by the author of the thesis himself.

My background is that of machine learning and knowledge engineering, which naturally influenced the way how the thesis was written, in the sense of topic selection, preferred terminology and stylistic form. I soon realized that the topic of the work intersects with a wider range of disciplines and research areas than a single person can be expert at. Many specialists will certainly argue that my treatment of problems belonging to e.g. uncertainty processing, statistical hypothesis testing or formal logic is rather superficial and biased; I had to take this risk, due to limited time.

I have been aided in my work by several colleagues from the department, to whom I owe my thanks. Among them, I would like to mention my supervisor Jiří Ivánek, Petr Jirků, Radim Jiroušek and Petr Berka, with whom I discussed most research problems and who also were careful readers of my papers which later formed the backbone of the thesis. I am also indebted to Miroslav Kubát for offering me a visit to the Graz Technical University in 1993; during this visit, I had the opportunity to study the newest machine learning literature, which I urgently needed for my work to progress. Another person who greatly influenced my project was Pavel Brazdil from the Porto University; he gave me useful advice (especially in the context of the International Diploma Course on AI, in 1993 in Brno) and sent me several relevant research papers. I have also benefited from literature from and discussions with Claire Nédellec, Jerôme Thomas, Stefan Wrobel, Foster Provost, Hussein Almuallim and many others. A further impetus for my work was my participation at European Conferences on Machine Learning in 1994, 1995 and 1997, where I could discuss the topics of interest with a large number of renowned European researchers. These and some other research activities of mine were partially sponsored by grants:

- *Building intelligent systems* - grant no.201/93/0781 of the Grant agency of the Czech Republic;

- *Using modern mathematical methods for analysis of economic information* - grant no.201/94/1327 of the Grant agency of the Czech Republic;
- *Integrated knowledge-based systems on workstations* - grant no.94/1097 of the Higher Education Development Fund;
- *Laboratory of intelligent systems* - grant no.VS96008 of the Czech Ministry of Education.

I am also indebted to Ivan Bruha from the McMaster University, Canada, for a large number of inspiring comments and suggestions concerning, in particular, the formal structure of the thesis, and to Vilém Sklenák, the present Head of my department, for general comprehension and also for his help with typographic matters.

Finally, my thanks belong to my parents and to my fiancee, who did their best (in a difficult situation for the family, due to my mother's serious illness) to create an environment in which I could concentrate on work, especially in the last months of completing the thesis.

The thesis consists of six sections, which differ both in size and in form.

Section 1 maps the broader context of the work, at a general level; among other, all terms from the title of the thesis are thereby introduced. Subsection 1.1 attempts to grasp the meaning of the terms *knowledge* (1.1.1) and *data* (1.1.3), as they are used in the discipline of knowledge engineering, and in particular in this thesis; also, the basic problems of knowledge *acquisition* and *modelling* (1.1.2), and the categorization of forms of data (1.1.4) are presented. Subsection 1.2 introduces the notion of machine learning (1.2.1) and outlines the relation between *learning* and *problem solving* (1.2.2); a modest contribution of the author himself (in this "descriptive" section) is the informal distinction of various problem-solving tasks falling under the common notion of *classification* (1.2.3). Finally, subsection 1.3 declares the possibility to use prior knowledge as input to the learning process, and thus serves as a bridge to the next section.

Section 2 maps existing research in the area of learning with prior knowledge. Subsection 2.1 is devoted to learning with prior *problem-solving knowledge*. First, we mention incremental learning (2.1.1), which starts with an empty body of problem-solving knowledge and updates it gradually; input knowledge thus always corresponds to target knowledge from the previous learning step. Then, we proceed with the closely related task of knowledge revision (2.1.2); there, input knowledge usually has different origin than data. Finally, we briefly mention the task of knowledge integration (2.1.3), which, by itself, does not have the character of learning from data, but (as we show in section 4) can be viewed as a part of a "bypass" of knowledge revision. Subsection 2.2, in contrast, concentrates on the task of learning with prior *static domain knowledge*. First, we categorize the ways how such knowledge (sometimes also called "background knowledge") can provide *bias* (i.e. additional information) to the process of empirical learning (2.2.1). Then we describe

two possibilities in more detail: constrained learning (2.2.2) and constructive induction (2.2.3), including examples of learning systems. Finally, we treat separately the use of two types of static domain knowledge - value hierarchies and integrity constraints (2.2.4) - which is the specific subject of original work in section 5.

Section 3 "interleaves", in the thesis, the descriptive part from the research part. It describes the ESOD method for learning and classification, as formulated by Ivánek, and mentions also its newer implementations made by Berka. The reason for devoting a whole section to ESOD is that it is a "background" project which motivated a large part of the original research described in sections 4 and 5; furthermore, as basic ESOD does not take prior knowledge into account, we could not include it satisfactorily into section 2.

Sections 4 and 5 are both devoted to original research, namely to exploitation of prior *problem-solving knowledge* and *static domain knowledge*, respectively. Each starts with an "objectives" subsection (4.1, 5.1). However, as the topics have been approached differently, the internal structure of the sections is dissimilar; the former is mostly theoretical, including proofs of given assertions, while the latter describes implemented algorithms and their empirical testing.

Section 4 presents mainly theoretical work in the field of learning with prior *problem-solving knowledge* in the form of rules. It consists of three main subsections, with the following topics:

1. The "bypass" model of learning from expert knowledge and data, which replaces knowledge revision with empirical induction and *knowledge integration* (section 4.2). The model has been conceived only as an informal scheme, and a few hypotheses have been formulated; it serves merely as a starting point for the next two topics.

2. The algebraic knowledge integration method for rules *without weight* (section 4.3). First, some formal notions are introduced (4.3.1); then the method for selection of an integrated ruleset is presented (4.3.2) and some properties proven (4.3.3); finally, open problems and future perspectives are outlined (4.3.4).

3. The technique for integration of *weighted* categorial rules from expert with rules discovered in observational data, based on (heuristic use of) confidence intervals (section 4.4). First, the task is presented from the viewpoint of knowledge revision (4.4.1); then, properties to be satisfied by an interpolation function are suggested (4.4.2) and an instance of such function constructed (4.4.3); finally, open problems and future perspectives are again outlined (4.4.4).

Section 5 presents methodologies as well as implementation and experiments in the field of learning with prior *static domain knowledge* in the form of abstraction (value) hierarchies and integrity constraints. It consists of four main subsections, of which the first two are devoted each to one form of prior knowledge:

1. Abstraction hierarchies on attribute domains. First, the notion of abstraction hierarchy is introduced (5.2.1). Then, the generic forms of exploitation of such hierarchies, both on input attributes (5.2.2) and class attributes (5.2.3) in learning are presented, with the following motivations:

   - hierarchies on input attribute values enable to extend the language of induced rules with abstract terms, making them more readable and meaningful.
   - class hierarchies enable to proceed from one-shot classification to systematic refinement of the conclusion, and to generate explanations of the reasoning of the classifier.

   Finally, a possibility to construct hierarchies automatically (by means of hierarchical clustering) is investigated (5.2.4).

2. Integrity constraints, which limit the state space of hypotheses (5.3).

The next three subsections describe details of implementation (5.4), testing on multiple datasets (5.5) and open problems and perspectives (5.6).

Finally, section 6 summarizes the (somewhat heterogeneous) outcomes of the work. The whole thesis is accompanied with a bibliography, a glossary of abbreviations and an index. A brief reference guide and the source code of (some of) the implemented programs has been included in one of the handouts, as a separate clip-on appendix.

æ

Contents

List of Figures

List of Tables

1 Introduction

1.1 Knowledge and data in knowledge-based systems

1.1.1 The notion of knowledge

Knowledge is one of the most general and ambiguous terms of the natural language; innumerable disciplines such as psychology, philosophy or pedagogy use it each in its own context. What is common to all of them is that knowledge is ascribed to human beings: it is an asset, which they can (somehow) acquire, hold and exploit. The fundamental difference of the particular meaning of knowledge used by the artificial intelligence (AI) community is the assumption that not only humans but also "artificial beings", namely *computer systems*, can operate with knowledge, be it in a particular way. Both humans and such "intelligent" computer systems are sometimes (especially in cognitive science) referred to as *intelligent agents*.

Intuitively, we feel that knowledge is a "deeper" or "richer" form of information; let us mention the definition introduced in a handbook for knowledge-based systems developers: "Knowledge is a *rich form of information*, such as that stored by humans as expertise in some restricted domain (e.g. problem-solving skills like medical diagnosis or resource scheduling). It is often expressed as facts, rules, concepts, relationships, assumptions, tasks etc." [TanHay93] A little more "intensional" definition, attempting to discriminate knowledge from "ordinary" information, may read: "Knowledge is what enables us to extract *new, previously non-existent information* from existing information sources". Under this vision, the most characteristic feature of *knowledge-based systems* (KBS)[1] would be their capability to apply limited information physically stored in the knowledge base on an unlimited, and often imprecisely defined, state space. This is the case of diagnostic consultation systems (e.g. MYCIN, see [Shortl76]), which can typically evaluate multiple diagnoses and choose the most plausible one for an unforeseen combination of symptoms; similarly, automated design systems can make up a new product by combining features according to given constraints and user's preferences, and so forth.

Beside the functional aspect of knowledge, we can assume some more formal features. First: in the context of knowledge-based systems, the term "knowledge" usually denotes

[1] At this point, we should probably make clear about the distinction between *expert systems* and *knowledge-based systems*, which is however not easy. Ethymologically, we would suggest the term expert system for a computer system miming the (mental rather than motorical) behaviour of an expert, i.e. a highly-skilled professional, and that of knowledge-based system for a computer system, whose behaviour is determined by a substance identifiable as "knowledge" (see later in this section for an approximate definition). From this point of view, we could imagine expert systems lacking explicit knowledge, e.g. those based on artificial neural nets, but also knowledge-based systems, which do not have a human counterpart and acquire their knowledge structures inductively from data. The latter case is important to this work, since it explicitely assumes symbolic *machine learning*. Historically, the notion of "expert system" is older; for a certain period, both terms were considered more or less equivalent. Recently, we can observe, at least within the "academic" AI community, a sort of reluctancy towards the term "expert system"; it is sometimes considered as a "dummy commercial label". In this work, we will mainly use the term "knowledge-based system", except when dealing with commercial applications.

a body of information kept in a *separate module* beyond the algorithmic part of the system, in a *declarative* rather than procedural form. It is this characteristics what justifies a formal distinction between knowledge-based systems and "intelligent-behaviour-exhibiting" programs with their ingredients of "intelligence" scattered around the program code. The architecture of a simple KBS serving for consultation with the user then may look as follows (Fig. 1).

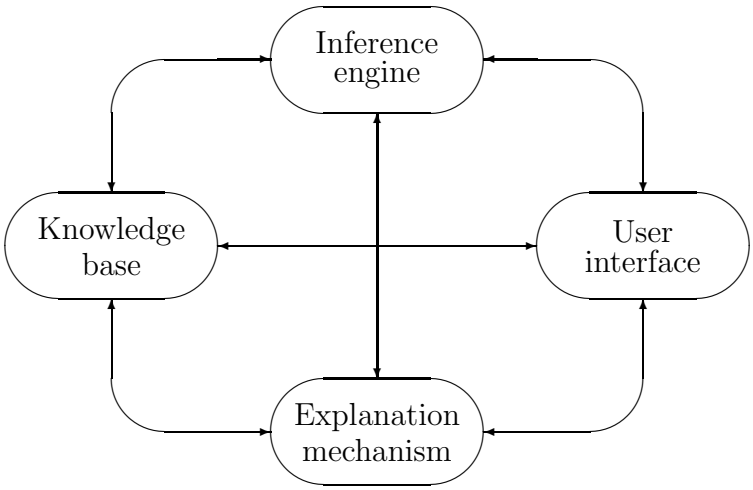

Figure 1: Architecture of a consultation KBS

Due to this modularity, different KBS can be made using the same (or, slightly modified) inference and explanation mechanisms, equipped with different knowledge bases (and, probably, user interfaces).

Another, although related feature of knowledge is its *intelligibility*. That means, computer knowledge, whatever its origin, should have a form compatible with human knowledge and be comprehensible to a human expert. This imposes restrictions on acceptable knowledge representations. Production rules, decision trees, semantic nets or frames are intelligible and can be for instance, with more or less effort, transcribed into a natural-language form. On the other hand, a setting of weights and thresholds in a neural net is opaque to a human expert, be the functionality of the system as a whole in perfect accordance with the expert's requirements. For the purpose of this work, we will further assume high-level, symbolic representation to be inherent to knowledge.

To summarize, we have restricted the intuitive notion of computerized knowledge into *"knowledge expressed symbolically, with declarative semantics, which is kept separately from other elements of a computer system"*. We could further refine and elaborate this vague definition, and introduce various knowledge representations used throughout the AI discipline; this would however broaden the scope and extent of the whole work beyond an acceptable limit. Moreover, as recent discussions within the AI community suggest,[2] a generally acceptable, rigorous definition of knowledge (and even of its computerized incarnation) represents an unsolvable problem.

1.1.2 Knowledge acquisition and modelling

The central issue for knowledge-based systems is the acquisition of knowledge. The traditional approach to knowledge acquisition focused on informal interviews with domain experts. This approach has, however, brought up many difficulties, the most important of which is usually denoted as so-called Feigenbaum's bottleneck: *Experts can use their knowledge in practice, but they are rarely up to formulate it on the paper"*. Many non-verbal techniques, such as *card sort* or *repertory grid* (see e.g.[ScoClG91], [TanHay93]) have been developed to ease the process of *transfer* of knowledge from experts to knowledge-based systems; the main goal however remained the same: to encode *all* knowledge necessary for solving the given task, in the *particular representational formalism* used by the system.

In the 1980s, it became obvious that this approach was deemed to thorough revision. The following is usually viewed as the main features of this so-called *paradigm-shift in knowledge acquisition*[3]:

- View of knowledge acquisition as of *modelling* rather than as of *transfer*. The models built in the process of knowledge acquisition are neither reprints of the expert's mental processes, nor operational knowledge structures ready to be put into an expert system shell. They should be as reusable and comprehensible as possible.

- The distinction between *knowledge-level* and *symbol-level* modelling, which has first been pointed out by A. Newell [Newel82]. A knowledge-level model is considered as independent of the particular knowledge representation; it is a unifying abstraction of what different problem-solving agents (humans and/or computers) may view somewhat differently due to their lower-level constraints. Among the knowledge-level ingredients, Newell enumerates agents, goals, and actions. More pragmatical knowledge engineering approaches concentrate on components of knowledge-level models built during the knowledge acquisition process: *tasks*, *problem-solving methods* and *"ontologies"* of the given problem domains [Schr95]. Symbol-level models,

[2]One of such discussions took place in spring 1994 at the KAW (Knowledge Acquisition Workshop) electronic mailing list. The archive of KAW list can be, in present, obtained via the FTP network service, see [KAW95]. Similar issues are however raised, from time to time, at many other conferences and workshops (both "physical" and electronic ones).

[3]In section 1.2.3, we make some notes on the consequences of this paradigm shift on the discipline of machine learning.

on the other hand, involve detailed problem-solving rules and inference mechanisms. Both symbol-level and knowledge-level models are forms of knowledge; the former are more operational (they can be more-or-less straightforwardly implemented) while the latter are more durable, flexible and transmissible[4].

- More-or-less clear separation of *domain knowledge* and *control knowledge* at the knowledge level. The former involves all terminology, descriptions of main domain concepts, relations among them etc. The latter captures the dynamics of problem-solving: steps to be performed so as to attain a goal. Several different problems can be, in principle, solved with the same body of domain knowledge, completed with appropriate problem-solving methods (control knowledge); the *generic part* of a problem-solving method can be, on the other hand, used in different domains.

One of conspicuous products of the paradigm shift was the creation of structured KBS development methodologies. The KADS methodology [WieScB92] represents a de facto standard for KBS development in Europe and covers the bulk of the KBS lifecycle. Both analysis and design are viewed as processes of *modelling*. As we are, for the purpose of this work, not interested in the practical aspects of system development, we will describe the only "truly-AI" part of the methodology[5] - the technique for development of the *model of expertise*. The model of expertise serves for capturing the required expertise (both domain knowledge and control knowledge) at the knowledge level, i.e. with little or no concern of how this expertise will be implemented in the finished KBS. It consists of four layers: domain layer, inference layer, task layer and strategy layer.

- The *domain layer* covers domain knowledge: basic facts, concepts and relations used within the domain. It can also comprise more complex structures such as qualitative models; it should not however involve any kind of control knowledge.

- The *inference layer* describes abstract *inferences*, which can be *possibly applied* on domain-layer knowledge, and the connections among them, thus forming a web - the *inference structure*. The remaining components of the inference structure are *domain roles*, which serve as input/output of inferences and are directly mapped onto domain-layer terms.

[4]Most recently, attempts have appeared to formalize and operationalize knowledge-level models so that they can verified for consistency, and even *executed*, as mock-ups of prospective implemented systems. Several *formal specification languages* have been suggested for this purpose (see [FenHar94] for an overview and comparison). As these languages express knowledge-level concepts by means of symbols, they can be viewed as a kind of "bridge" between knowledge level and symbol level [Fensel93].

[5]We actually refer to the version of KADS from the beginning of 1990s (KADS-I). In 1994, the completion of a new standardized version has been announced, under the name of KADS-II or Common KADS [KADS95], [DeHMaW94]. Some differences can be observed between the two, nevertheless, the main ideas persist. The acronym KADS originally stood for "Knowledge Acquisition and Design System", now it is rather felt as a genuine term.

- The *task layer* specifies which inferences, and in what order, are *actually applied*. Each inference from the inference layer is understood as a primitive task in the task layer. Primitive tasks are then regrouped into hierarchies, and connected by means of procedural operations, such as sequence, selection or iteration.
- The *strategy layer* corresponds to strategic decision-making of the expert, such as choosing among multiple task hierarchies, their dynamic scheduling etc. It is worth elaborating for large and complex projects only.

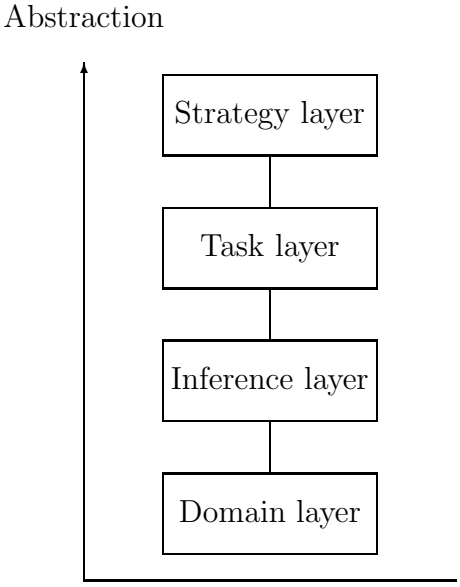

Figure 2: Degree of abstraction - theoretical view

When comparing the layers according to degree of abstraction, the above ordering suggests the idea of Fig. 2. However, the design of the strategy layer is, in practice, tightly bound with the problem domain, and can be viewed as more specific than the inference and task layers. Fig. 3 is thus more realistic. The inference and task layers have, on the other hand, many generic features which do not differ very much across problem domains; with a certain simplification, we can assume that they do not model the current, specific, task, but a generally defined, generic one. The reusability of inference and task layers, in the form of so-called *generic task models*, can save a notable amount of work compared to developing KBSs from scratch. In this respect, several versions of Generic Task Model (GTM) Library have been put into exploitation in the KBS development circles [TanHay93]. The set of GTMs contained in the library[6] has been organized into a hierarchical structure, the top nodes of which being the notions of system analysis,

[6]Some of the GTMs are derived from generic parts of actual KBSs, others have been contrived "artificially", by means of thorough analysis of human problem solving.

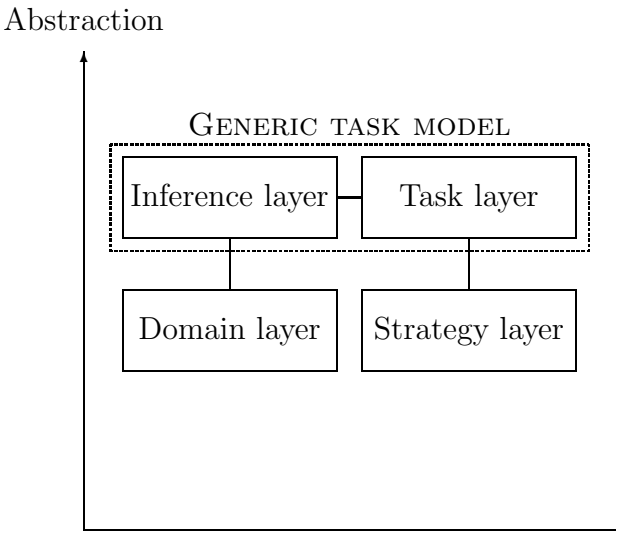

Figure 3: Degree of abstraction - realistic view

system modification and system synthesis, see Fig. 4[7]. To clarify the notion of generic task models, and of KADS descriptions in general, we have taken the example of *heuristic classification* - a generic task outlined by W. J. Clancey as early as in 1985.

The traditional outlook of *inference structure* of heuristic classification is in Fig. 5. Primitive inferences are in ellipses and domain roles in rectangles. In KADS, inferences are denoted by verbs; here, we have three of them: abstract, match and specialize. The *abstract* inference transforms specific observable values - *observables* - onto abstract values - *variables*. The *match* inference finds an abstract class of solutions - *solution abstractions* - relevant to the input variables. Finally, the *specialize* inference transforms the abstract class onto a real class - *solutions*.

The *task structure* of heuristic classification may vary. As the most obvious alternatives, we can view forward reasoning and backward reasoning. Both are displayed, in a pseudocode notation[8], in Fig. 6. In forward (or, data-driven) reasoning, the acquisition of observables leads to abstraction, match and specialization, and in the end, the solution is obtained. In backward (goal-driven) reasoning the overall inference process is triggered by the need of a solution; the inferences are called in the inverse order, with uninstantiated arguments, and the instantiation takes place only in the subsequent forward propagation.

In order to develop a real KBS, one or more GTMs are typically selected from the library, and modified according to the needs of the current task. In parallel, domain

[7]The diagram covers only a part of the library. Arrows indicate where further GTMs exist; the dashed boxes correspond to abstract, high-level categories of tasks, for which no actual GTM exists in the library.

[8]The "+" and "–" signs indicate input/output mode of arguments.

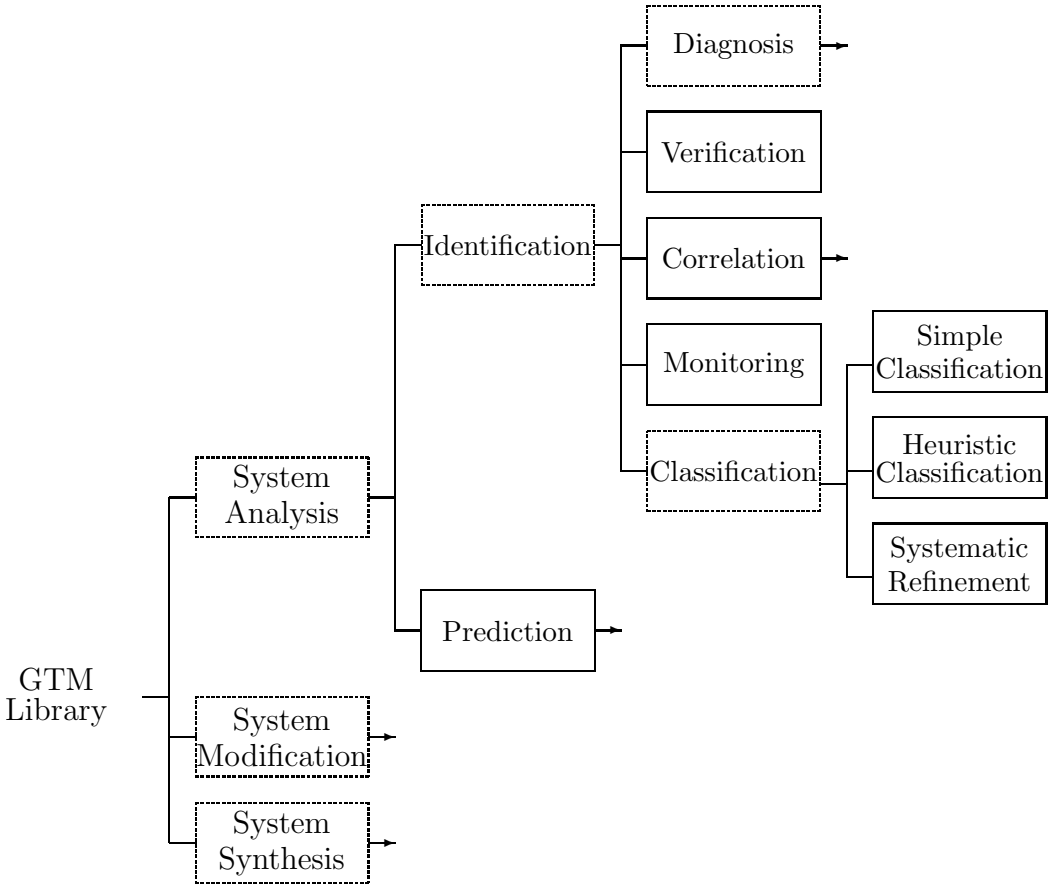

Figure 4: Fragment of the hierarchy of generic task models in the library

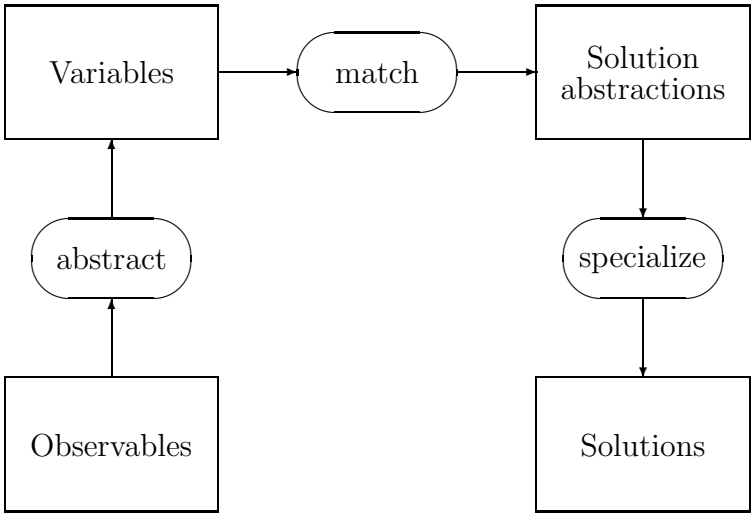

Figure 5: Inference structure of heuristic classification

```
/* Forward reasoning */
Heuristic classification (+Observables,-Solutions) by
     obtain data for Observables
     abstract (+Observables, -Variables)
     match (+Variables, -Solution abstractions)
     specialize (+Solution abstractions, -Solutions)

/* Backward reasoning */
Heuristic classification (+Observables,-Solutions) by
   specialize (-Solutions, -Solution abstractions)
        match (-Solution abstractions, -Variables)
             abstract (-Variables, -Observables)
                  obtain data for Observables
```

Figure 6: Alternative task structures of heuristic classification

knowledge is collected, associated with domain roles of the existing inference structure/s, and coupled with the pre-fabricate generic model/s to form the entire model of expertise.

In the *design* stage, which follows after analysis (but largely overlaps with it), the model of expertise is operationalized to become the knowledge base of the prospective KBS. It is important that design, in the KADS approach, is *structure-preserving*: objects from the detailed design models map onto objects from the global design model, and further to the analysis models (especially the model of expertise). This is favourable when errors occur within the development process, as they can be traced back to their origin; preservation of structure also extends the capabilities of the explanation mechanism, as the results of problem-solving can be justified not only by shallow operational knowledge but also by knowledge-level models.

Throughout this thesis, we will refer to KADS (namely, to its idea of expertise model) several times, both when describing the state-of-the-art of machine learning and the original work of the author.

1.1.3 Data vs. knowledge

Beside knowledge, any realistic knowledge-based system needs a certain amount of concrete *data* to operate on. For a medical consultation system, a unit of data may be the record of a particular patient; for an intelligent process control system it could be a set of measurements on process variables taken at a given time; for a knowledge-based bankruptcy analyser a set of economic indicators of a particular company. Often, we speak about the *database* of a knowledge-based system, which probably requires a certain analysis.

The role of data in early, *rule-based KBS* significantly differed from their role in conventional database systems (DBS). In a DBS, data are stored for a long period of time; in a typical rule-based KBS, data were the short-term operational component, while knowledge took over the role of the long-term one. A "classical" diagnostic KBS keeps an input database, often amounting to a single record of input features (symptoms), which is initially or gradually supplied by the user or by the environment. Besides, it maintains and updates its internal, dynamic database using the intermediate outcomes of reasoning, such as the weights of propositions or lists of diagnoses to be investigated. In the end of the session, relevant results are output and both the input and dynamic databases are cleared.

More recently, however, "second generation expert systems" (cf. e.g. [MarVlc92]) have appeared. They typically have a more complex architecture, and their knowledge component is not restricted to simple IF-THEN rules. Among the sources of information they exploit, we can find deep causal models, behavioural models, as well as static domain knowledge (such as structural relationships or taxonomies) and permanent *databases*. Also their problem-solving strategy is not reduced to a uniform set of diagnostic inferences (such as rule matching and firing). They do not consider a single data object (feature vector) at a time, but often combine information from multiple explicitely declared objects.

```
knows(peter,jane).
knows(X,Y) :- lives_with(X,Y).
knows(X,X).
knows(X,Y) :- X=Y.

man(peter).
man(john).
tall(peter).
short(john).
is_friend_of(peter,john).
```

Figure 7: Prolog clauses as knowledge vs. data

We can also name some specific approaches to reasoning, which are usually viewed as "belonging to AI". *Case-based reasoning* (CBR) systems (see e.g. [Kolodn93], [AamPla94]), which is a sort of alternative to rule-based systems, explicitely use previously encountered data objects to make assumptions about the object (decision situation) currently examined. Knowledge is thus "dispersed" in a set of selected (and sometimes partially generalized) data objects. Another discipline where knowledge and data are in particular relation is the discipline of *deductive databases*, situated at the borderline between knowledge engineering, relational database engineering and logic programming. In deductive databases, a body of knowledge (typically expressed as first-order rules) is used, together with a database of ground facts, to represent an extremely large database intensionally (for more information, see e.g. [Bry93]). Here (somewhat conversely to CBR), data is represented by means of knowledge.

As soon as we lift the restriction to propositional IF-THEN-rule (for knowledge) and feature-vector (for data) representations, we also partially loose the *syntactical distinction* between knowledge and data as such. In a logic programming environment, such as the Prolog[9] language, which is frequently used for prototyping KBSs, we enter the input information in two syntactically distinct forms - as facts and as rules. It is, however, not guaranteed that the facts have the semantic interpretation of data; finer syntactical measures have to be applied. Consider e.g. the examples of clauses at Fig. 7 (first part).

The first clause, stating that Peter knows Jane, belongs to the class of so-called ground facts - facts without variables - which can usually be interpreted as *data*. The second clause is a rule stating that if X lives with Y then he/she knows him/her; this can be viewed as a fragment of *knowledge*. The third clause is, again, a fact; this time, however, a non-ground one. It may be read as "everyone knows himself"... which is, however, sort

[9]The underlying formalism of Prolog is that of Horn clauses, i.e. restricted first-order logic. A larger part of (especially applied) research on logic programming still exploits Horn clause or derived formalisms.

of *knowledge* rather than data. Finally, the fourth clause has the same meaning as the previous one but it has the form of rule.

In addition, in a relational language, it is difficult not only to distinguish between knowledge and data, but also to separate individual *data objects* (which can be done straightforwardly for feature vectors). A relational description of a real-world object may consist of several clauses, each of them mapping a particular aspect of the object. Moreover, a single clause can span over several real-world objects and express a relation among them. Examples are again at Fig. 7, in the second part. Clauses of unary predicates `man`, `tall` and `short` describe different properties of objects `peter` and `john`; the last clause indicates a binary relation between these two objects.

From what was said is, again, evident that when speaking about knowledge versus data, we should take account of semantic meaning rather than of syntactic appearance. Also, different subdisciplines and co-disciplines of AI have developed their proper interpretation of both terms. Throughout this work, we will come across several of them; with a certain cautiousness, we can hopefully avoid confusion. As a last resort, we could perhaps serve ourselves with the pragmatic distinction introduced by Widerhold: what can be obtained exclusively with the help of an *expert* is knowledge, while what can be efficiently collected by a *clerk* is data (see [Wider86]). If we admit that the expert's assistance may also have indirect forms, such as provision of examples or selection of relevant features to observe, Widerhold's statement preserves its efficiency despite the far-reaching metamorphoses of the AI discipline.

1.1.4 Observational data vs. examples

At the end of the last section, we have shifted from the syntactic viewpoint to a more pragmatic one. Here, we will attempt to discern the types of data which differ in their origin, and also in their properties, especially with respect to their use for empirical learning: examples, experimental data, and observational data.

At one end of the scope, we can identify carefully selected or even artificially constructed *examples* of the expert's decisions. This extreme form of data is rather similar to knowledge, as the collection of representative examples requires a considerable effort of the expert himself. His expertise should be a guarantee that the description of examples contains all relevant features and not too many irrelevant ones. The size of example sets is usually low: partly because experts have their knowledge already arranged in a concise manner, partly due to the limited time they can allot for this activity. The examples are typically free of contradictions and redundancy.

The data can be also obtained as a result of *experiments*. Then it is, sometimes, possible to generate a representative collection of data; if there is a "gap" in the coverage of the state space, it can be deliberately filled with a new experiment; also, new descriptors can be added if the results suggest that they may be relevant. However, both the quantity and quality of data is limited by the (not only financial but possibly social, ecological etc., according to the nature of the task) cost of experiments. Experimental data can also contain some noise, if the feedback from the environment is not fully reliable.

Raw *observational* data stand as the other extreme. There, we do not have any possibility to influence the representativeness of data and to fill in the gaps, as we have to take the data "as they are". Again, the data can be burdened with noise. Observational datasets are typically rather large, and they may have been collected for purposes substantially different from the current problem-solving task.

For learning in *relational* domains, especially in inductive logic programming (ILP) [LavDze94], relatively small sets of examples (up to a few tens) are typically used; these are either constructed manually or generated automatically from the target predicate definitions (the latter case however means that what is to be learned is known in advance). The datasets exploited within inductive data engineering (cf. [Flach93], [KukPop94]), an interesting discipline laying on the intersection of ILP and deductive databases, are often much larger (this is related to the fact that the hypothesis space is slightly more restricted compared to the space of logical programs in "classical" ILP). Recently, there have been promising attempts to use experimental or observational data described in a sort of first-order logic; they mostly relate to the area of control of complex systems - e.g. robotic perception [KliMoR95] or flight simulation [Camac95].

Nevertheless, the most typical format for observational datasets remains the *attribute-value* (also known as feature-vector) representation, which corresponds to the propositional language in terms of expressivity (the slang term "zeroth-order" is thus sometimes used). The research on attribute-value learning techniques started as early as in the 1970s; gradually, both the efficiency of algorithms and the increased power of computers enabled to proceed from small example sets to real-world databases containing many thousands of objects. This stream of machine learning research is now more and more associated with the boosting discipline of knowledge discovery in databases (for references, see e.g. [Nakh95]). Learning from observational data expressed in attribute-value format is also the main focus of this thesis.

1.2 Learning and problem solving

1.2.1 The position of machine learning

Inductive learning can be characterized as acquisition of general and structured information - *knowledge* - from specific examples or data. This phenomenon is studied within multiple disciplines, such as logics or statistics; however, it is only in the (predominantly experimental) discipline of *machine learning* (ML) that it plays the *central* role. Machine learning is actually a vaguely defined notion: it covers a host of dissimilar methods and approaches. The unifying ideas are perhaps:

- stress on *inductive* reasoning processes,
- use of *symbolic* (in contrast to numerical) representation,
- relaxation of some formal criteria of "soundness", in exchange for more *pragmatical* notions of "plausibility" and "performance", which are easier to verify and/or satisfy,

- assumption that knowledge learned is not important *per se* but only when coupled with a certain *problem-solving* mechanism.

There are, nevertheless, many methods which are considered as belonging to ML but lack at least one of the above features.

When the notion of machine learning first appeared (in the end of the 70s), it was in close connection with knowledge acquisition for expert systems. This was, in fact, one of its two main distinctions from the by-then well established disciplines of exploratory data analysis and pattern recognition (the other one was the emphasis on symbolic rather than numerical descriptions). In exploratory data analysis, the task is to find interesting relations in data; the subsequent use of these relations is of lesser importance. Pattern recognition, on the other hand, attempts to find discrimination functions useful for recognizing previously unseen objects; the details of such functions are, however, not necessarily made explicit.

The vaguely defined area of machine learning, in this respect, spans between the two. Some of its streams (namely, those related to knowledge discovery in databases [Nakh95]) stress the explorative character of learning, and thus are similar to classical data analysis. Other approaches aim at maximizing performance even at the expense of comprehensibility especially those methods which are fitted to numerical or mixed domains, such as multivariate decision trees [MurKaS93],[BroUtt95] or neural networks. The mainstream of machine learning can be, however, characterized as learning *explicit* knowledge usable for *problem solving*.

1.2.2 The inseparability of learning and problem solving

In the context of intelligent agents (or, systems), the activity of learning is usually defined as "the capacity of an agent (system) to improve its behaviour in time"; if the agent has the character of problem-solving agent (such as a knowledge-based system) this translates as "to improve the quality of its problem solving". From this follows that the learning task strongly depends on the problem-solving task.

The character of the problem-solving task may vary. At the "lower" end of the scope we can find simple actions which consist in establishing a *many-to-one* mapping: the induced knowledge structure serves to suggest one of fixed conclusions based on a larger set of input information. Generally, this sort of tasks can be viewed as an assignment of a class to an object described by a set of features; therefore, the notion of *classification*[10] is typically used. At the "upper" end, we can find complex problem-solving tasks such as planning or configuration, where the output has the character of a newly built structure. Powerful problem-solving architectures have been suggested, which, in theory, tackle any

[10]This, somewhat ambiguous, term is commonly used in the machine learning community. In the context of expert systems, the term "diagnostic" was traditionally preferred. Some researchers suggest other terms, such as "prediction" (not in the temporal meaning introduced in the next subsection, but rather as "prediction of class for newly encountered data objects"). We will, according to the practice of ML, stick to the term "classification", understanding it as a general concept covering other, more specific ones - see the following subsection.

problem requiring state-space search; learning the relevant knowledge usually has the form of remembering and generalizing sequences of problem-solving steps, such as the so-called "chunks" in the SOAR system [LaiRoN86]. A particular problem-solving task is *theorem-proving*; specific learning methods exist which induce knowledge usable for logical theorem-proving, such as inductive logic programming [LavDze94] or explanation-based learning [MitKeK86], [DeJMoo86].

Learning and problem solving in intelligent systems may be either *tightly* or *loosely* coupled. In the first case, learning is triggered by a failure of the problem solver ("failure-driven learning", cf. [VSom93]). After the learning phase (which has the form of revision of existing knowledge), the problem solver reassumes control. In the second case, the bulk of learning precedes problem solving. If difficulties arise during the problem-solving phase, they are merely recorded, and the information on the failure is used as soon as the learner is invoked again. This distinction between tightly and loosely coupled learning and problem solving is closely related (but not identical) to the distinction of batch and incremental learning systems, presented in section 2.1.1.

The original contribution of this thesis falls under the subgroup of learning techniques *loosely coupled* with problem solving; problem solving has the form of *classification*. Therefore, the next subsection is restricted to this category of tasks.

1.2.3 Classification as problem solving

A problem-solving task can be characterized as *classification* if it corresponds to the following description:

- Input information consists in a set of observations (observable, measurable or otherwise identifiable features) about some real-world entity; we will denote this set as *data object*[11].
- The observations are matched with problem-solving knowledge and a conclusion (or, possibly, several alternative or even compatible conclusions) is selected from a predefined (most often, finite) set.

The semantic of the task, and the nature of the conclusion may vary. The following is a (probably not exhaustive) list of different semantics of classification tasks. For each of them, we outline the essence of the task, point out some of its syntactic peculiarities, and also attempt to grasp the *inverse relation*, which maps the conclusion to the observables.

Recognition The data objects represent collections of properties of a concept; some of them may be relevant while some other irrelevant. Available knowledge states that the given combination of properties is *characteristic* for one of known concepts (i.e. conclusions). Hence, the inverse relation is "has_properties". *Example: given a collection of features of an animal observed, determine which of the known animals it is.*

[11] The data object can be viewed as a particular representation of the real-world entity.

Recommendation The data objects represent particular situations. The conclusions are recommendations of actions; each conclusion could be also understood as a class of situations in which the given action is suitable. The inverse relation would be something like "can_be_used_for".

Example: given certain requirements of the customer, determine which of the products on stock should be offered to him/her.

Control Very close to recommendation, but with a dynamic aspect. The data objects describe *states* of a single real-world object rather than individual real-world objects; conclusions correspond to control actions to be applied on the observed object itself ("closed loop"). Input observables are often numerical, and so can be even the conclusions (the phenomenon of "continuous classes").

Example: given a state of measurements inside a nuclear reactor, determine which of the possible control actions should be performed.

Diagnosis The data objects represent sets of symptoms; conclusions are diagnoses (of human diseases, machinery faults and the like). Since the diagnosis causes the presence of symptoms, problem-solving rules are sort-of inversed causal rules, and the inverse relation typically is "causes". *Example: medical diagnostic - given a set of symptoms, determine the disease which may have caused them.*

Prediction The data objects represent situations or states of an object; so do the conclusions. Individual observables represent either factors which can *cause* the validity of the conclusion in the future, or manifestations which have *common causes* with the conclusion. The inverse relations may be "is_caused_by", and/or "has_common_causes_with". *Example: given a certain situation at the capital markets, determine whether the stock index will grow, drop or remain stable.*

Real-world problem solving often combines multiple tasks with different semantic. For example, a fault in a technical system can be e.g. pursued by means of *diagnostic* reasoning, followed by establishing a *recommendation* of a *control* action; the future state of the system may be then *predicted* based on all relevant information.

It might also be interesting to compare the above suggested typology with the traditional distinction of *concept acquisition* and *descriptive generalization* in machine learning, introduced by Michalski [Michal83]. He identifies the former with *learning from examples* and the latter with *learning from observation* (giving these terms a somewhat different flavour than we do in section 1.1.4), and presents a simple example:

- A recognition rule for the *concept* "philosopher" might be "a person who pursues wisdom and gains the knowledge of underlying reality by intellectual means and moral self-discipline".
- A *descriptive generalization* gained from the observation that the philosophers Aristotle, Plato, and Socrates were Greek while Spencer was British may read "most philosophers were Greek".

Concept acquisition thus can be viewed as learning the properties of the concept itself while descriptive generalization as learning the properties of objects satisfying the concept.

In this respect, the knowledge necessary for performing *recommendation*, *control* and *prediction* (in the above outlined meaning) should probably be obtained via descriptive generalization, while learning the knowledge for *recognition* would require *concept acquisition*. *Diagnosis* would fall either to the first group (e.g. if the symptoms are considered as characteristics of *patients* having a certain disease) or to the second (e.g. if the symptoms are considered as characteristics of the *disease* itself).

We can also compare our list with the list of generic task models (GTM) from the GTM Library of the KADS methodology ([TanHay93], cf. section 1.1.2), namely with the class of generic tasks falling under the concept of *system analysis*, in the taxonomy from Fig. 4, page 15. It should be kept in mind that the KADS library is destinated for general KBS analysis and design and is not restricted to classification tasks which can be represented in the many-to-one fashion. Most models are actually too complex for the knowledge involved to be learned as such using machine learning techniques, except as "dissolved" in shallow problem-solving rules.

The KADS taxonomy distinguishes two types of analysis tasks, with respect to the aspect of time: *Identification* tasks (related to present time) and *Prediction* tasks (related to the future). Among the Identification tasks, there is one - in fact, the most trivial task within the whole GTM Library - which corresponds to direct reasoning (in one primitive inference) from observables to the class: *simple classification*. There is no other input information than the set of observable features (although implicitely, plausible classification requires some knowledge); it thus corresponds to our notion of classification. There is also a set of *diagnosis* tasks, which are however richer than our many-to-one notion: there is a distinct element of input information - the *complaint*, which represents the initial impetus of the reasoning process. Among the tasks which consist of multiple inferences and/or require additional input knowledge, we can also mention *heuristic classification* (cf. section 1.1.2) and *systematic refinement* tasks; the research part of this thesis (in particular, section 5) actually shows an approach how to learn structured problem-solving knowledge, which is, in a sense, usable for such complex tasks.

In fact, the existing machine learning research often concentrates on details of learning techniques and looses the ampler, goal-directed perspective; this problem has resounded during several workshops on the relation between machine learning and knowledge-level modelling (cf. e.g. [Fensel94],[ScmAit95]). In our work, we refer to knowledge-level models at different places, taking advantage of the well elaborated descriptions of the KADS methodology. The marriage of knowledge-level modelling and ML can be, according to [Nedel96] observed in two variations:

- knowledge-level models of problem solving are used to find such subtasks in the *knowledge acquisition* process for complex systems which are suitable for application of ML algorithms, and to provide these algorithms with learning bias (e.g. in the ENIGME project [ThoLaG93], see section 2.2.2 for more detail);

- knowledge-level models of ML algorithms themselves are built which make the biases of the algorithms explicit and thus enable to choose among several algorithms the suitable ones for a particular application task (e.g. in the HAIKU project [NedRou94], see section 2.2.1).

In the future, an integration of these two streams is expected, which would further increase the importance of KADS (or similar) method.

1.3 Learning with prior knowledge

Among the sources of information a knowledge engineer disposes when building a model of expertise, two are of particular importance - domain expert and data. Traditional approaches to knowledge acquisition relied on interviewing the domain expert, who should be, ideally, able to communicate the body of knowledge (e.g. in the form of rules) he/she uses for solving the given problem. Concrete examples ("past cases") were only used as starting points for individual elicitation sessions; in the hands of a skillful knowledge engineer they could however serve for verification and detection of inconsistencies in the expert's assertions.

When *machine learning* (ML) first touched the knowledge engineering ground, it was with the assumption that experts are rarely up to formulate rules, never mind plausible ones. The role of experts should be therefore reduced to that of "generators" of examples. Furthermore, in some areas, they might be outplayed by large databases containing observational data. From the examples/data, empirical rules can be inductively extracted.

This prominent stream of machine learning, sometimes denoted as empirical learning[12] or learning from examples/data, has long been viewed as "learning without prior knowledge". It was however obvious that efficient use of empirical learning techniques required a substantial amount of additional information in the form of so-called *learning biases* (see section 2.2.1). Recently, there is a growing interest, within the machine learning community, in making these biases explicit, declarative, in the form of *background knowledge* (see section 2.2.1). These efforts are connected with a reintegration of the machine learning and knowledge acqusition disciplines (see [Kodrat94]), which have, for a certain period of time, followed divergent paths due to the "paradigm-shift" in knowledge acquisition (cf. e.g. [Nedel95], [Fensel94]). Knowledge acquired from an expert (often, in the form of abstract models) is used for constraining the search space (in constrained induction, section 2.2.2), or for transforming it by means of language shift (in constructive induction, section 2.2.3).

Specific streams in machine learning, on the other hand, stressed the role of prior knowledge from the very beginning:

- In *knowledge revision* (section 2.1.2), input knowledge is operational but partially incomplete or incorrect. *Incremental learning* (section 2.1.1) represents an inter-

[12] In this overview part of this work, we do not describe traditional empirical learning, as it is sufficiently covered in textbook-like literature. Most prominent approaches are e.g. top-down induction of decision trees [Quinl86], [Quinl93], "star"-based approaches [MicCaM83], [ClaNib89], or version spaces [Mitch77].

mediate notion between "pure" empirical learning and knowledge revision, as it assumes that target knowledge is learned uniquely from data but by means of successive updates of the body of knowledge which is initially empty; examples are entered one at a time.

- In *explanation-based learning* [MitKeK86], [DeJMoo86], input knowledge is complete and correct, but non-operational, i.e. it is not immediately suitable for problem solving.

- *Apprenticeship learning* [Wilk90], [TecKod90], [NedCau92], a borderline approach related to both machine learning and knowledge acquisition, concentrates more specifically on refinement of knowledge bases by means of tracing the problem-solving process, especially in complex problem-solving tasks such as medical diagnostics or configuration of technical systems; a human expert is usually involved as an external oracle.

In the following section, we will briefly review the principles and show some examples of methods belonging to two areas: *learning with prior problem-solving knowledge*, with stress on knowledge revision, incremental learning, and knowledge integration, and *learning with prior static domain knowledge*, represented by constrained induction and constructive induction. We will analyze the roles these approaches assign to various forms of knowledge and data. We must keep in mind that each of the approaches is by itself fairly complex, and has many instantiations and variations; there are also many learning methods (e.g. within explanation-based learning or apprenticeship learning mentioned above) which may fall under several "headings", as the approaches have partially evolved from each other and still largely overlap. It would be extremely difficult (or, probably, impossible) to make out a uniform framework encompassing the approaches with all their intricacies. We therefore present each approach in its original context and illustrate it on original examples where needed. Yet, we do not completely abandon the need for mutual comparison; we limit it, however, to the abstract level, represented by "neat" I/O (input/output) diagrams.

In all diagrams, we consider one form of output - target *problem-solving knowledge* (PSK), and at most five different inputs:

1. *Problem-solving knowledge* (PSK): an imperfect (e.g. incomplete, inconsistent or overly complex) form of target knowledge.

2. *Static domain knowledge* (SDK): factual knowledge from the problem domain; it cannot be directly used for problem solving but the learning algorithm exploits it in the process of forming target problem-solving knowledge.

3. *Meta-knowledge* (MK): knowledge influencing the way how other forms of input knowledge (PSK and SDK) are treated; it can be domain-dependent but, unlike static domain knowledge, it is meaningful in the context of the learning task only.

4. *Data* or *examples* (D/E).

5. *Oracle*: a human user of the system (usually considered as domain expert), who possesses his/her own knowledge and can provide input information on request.

For each approach described in this overview, the generic names of inputs are instantiated with names which are customary within the approach itself. The instantiation is only rough, as some knowledge structures are difficult to assign to a single category. The reader should be aware that the role of this section (and of the diagrams in particular) is to enable this, somewhat superficial, comparison, rather than to provide a deeper insight into the principles of each method.

In this overview, the choice was limited to the methods the author of this work is (to a certain extent) familiar with, namely to methods which can be roughly characterized as "symbolic". Other methods, which could also be characterized as learning in presence of prior knowledge, have been omitted, such as e.g.:

- Purely *probabilistic* approaches, such as learning in bayesian networks where the structure is defined by expert and the probability distributions estimated from data.

- "Subsymbolic" approaches to learning, in particular to knowledge refinement, where prior problem-solving knowledge is refined by means of *neural networks* or *genetic algorithms*.

These methods may have similar goals but they are based upon particular theoretical backgrounds, the presentation of which would improperly increase the volume of this work (set aside the limited acquaintance of the author).

æ

2 Related research - learning with prior knowledge

2.1 Learning with prior problem-solving knowledge

2.1.1 Incremental learning

According to a widely accepted opinion, human learning is *incremental* in its nature. New knowledge, whether acquired "ready-made" from outside or derived by the person himself, is repeatedly added to previous knowledge and integrated with it into structures far more complex than any computer system may ever posess. Obviously, nobody starts to learn by forgetting everything he knew before. This fact being probably in the minds of ML researchers, several early empirical learning algorithms were also incremental - they learned from one example at a time and updated successively the concept description induced. As one example, we can consider the famous Winston's ARCH learning program [Winst75], [Winst92] for recognizing objects in the block world, devised as early as in 1975. As another, let us recall Shapiro's MIS system [Shap83], considered as the first serious attempt at an inductive logic programming system. For this first generation of incremental learners, a high-level description language (restricted first-order logic) was characteristic. They were conceived as academic research experiments rather than as starting points for real-world applications; in this context, also input data were understood as collections of selected examples rather than masses of observational data, and the presence of noise was not taken into consideration.

Later on, *batch* mode systems, processing the whole source data set at the same time, started to proliferate in the field of empirical learning. AQ and TDIDT[13] families of algorithms entered the scene in the period of "ML revival" spurred by progresses in the area of expert systems. Despite the habitual label of "learning from examples", clones of these mainstream algorithms have been implemented in a form suggesting the term of "symbolic data analysis". A flagrant example is the commercial version of the ACLS (descendant of ID3) procedure, destinated for "discovering rules in database files".

During the 80s, batch-mode, attribute-based learning systems were preferred for most application domains. Another "incremental wave", however, swelled up in the second half of the decade. First "incremental learners of the second generation" were mostly incremental versions of existing batch algorithms: ID5 [Utgoff88] added capabilities for decision tree update to the original ID3; the AQ15 incremental rule learning system [MicCaM86] was an extended version of AQ, etc.

Recently, more sophisticated algorithms have been proposed. Some were designed to operate in dynamically changing environment with so-called *concept drift*, like the FLORA system by Widmer and Kubat [WidKub93] or the STAGGER system by Schlimmer and Granger [SchGra84]; some other paid specific attention to domains with noisy data - e.g. YAILS system by Torgo [Torgo93], which preserves redundancy in concept descriptions to achieve robustness. This family of powerful incremental systems is still based on the

[13]This stands for "top-down induction of decision trees"; the most famous member of this family was Quinlan's ID3 [Quinl86].

attribute-based description language; their extension into first-order logic was, though, envisaged by their authors.

Last, we will see that incrementality is, somewhat implicitely, present in knowledge revision systems making part of knowledge modelling environments, such as MOBAL (sections 2.2.2 and 2.1.2). This can be probably justified by the above mentioned incremental character of human learning, as the primary role of knowledge modelling tools is to model *human expertise*.

Let us enumerate the most frequently stated strengths and weaknesses of incremental (versus batch) algorithms and define conditions under which they are most significant.

Time economy Incremental algorithms are "more *economical in time* since they do not always start from scratch" [Utgoff88]. In other words, with an incremental learner the user need not recreate the whole amount of output knowledge (which he has to do with a batch learner) when he wants to take a new example (or, quite typically, an observation or measurement) into account. This was, as it seems, historically the first (often implicit) argument for incrementality. The time economy phenomenon, however, has a practical impact only if the source of examples or observations is incremental by nature and supplies them in real time. Given an extensive protocol of collected examples or observations, batch learners usually outperform incremental learners even in terms of speed.

Space economy Incremental algorithms are more *economical in space* as they operate with a single example rather than with large amounts of data. This "advantage" is, however, rather arguable. Many (especially older) incremental learners utilize "full memory", i.e. store all input examples into an internal database to ensure that the theory is constantly in accordance with them. On the other hand, "partial memory" algorithms are "extremely difficult to design" [Bruha91] and their accuracy is usually lower (unless we consider the case of dynamic environment, discussed right below).

Adaptability Incremental algorithms can continuously adapt the induced theory in a *dynamic environment*, where target concepts change over time. This is an urgent, application-driven (cf. e.g. [Kubat89]) incentive for the development of incremental systems. In a dynamic environment, the full-memory model is useless, since old items of input data get outdated. As a memory model appropriate for this group of algorithms, so-called *windowing*[14] was suggested: a limited set of (most recent) data objects is kept in the database and compared with the theory. Sophisticated programs, such as FLORA3 [WidKub93], use heuristics for dynamically expanding and contracting the window according to "concept drift indicators" - simply speaking, when the target concept is suspected to change, the tendency to "forget" old examples is reinforced compared to a period of relative target concept stability.

[14]Windowing was tested also on batch learning systems, e.g. by Quinlan [Quinl86]; in the course of time, the advantages of windowing in batch mode however showed spurious.

Familiarity As was already stated at the beginning of this subsection, incremental learning is *close to human reasoning*. This ressemblance can be beneficial for implementing heuristics borrowed from human learning into computer systems. Conversely, incremental learning steps are easier to interpret in a "human fashion" than operations on large data sets. In this respect, e.g. the STAGGER system [SchGra84] may feature "psychologically plausible" conservativeness when it is "unwilling" to abandon a concept that it repeatedly and successfully recognized for a certain period of time.

As the preceding paragraphs suggest, *incremental learning* is particularly useful for applications where data objects arrive on input subsequently, in real time (in addition, incrementality is almost inevitable in the case of dynamic environment). *Batch learning*, on the other hand, manifests its superiority (or, at least, equality) when the whole dataset is available at the same time - intuitively, "global" learning is more likely to lead to a globally optimal, or near-optimal, solution.

The two cases above are, however, somewhat extreme examples of situations in which empirical learning is applied. Consider a quite realistic situation of input data presented to the learning system in "repeated batches". Every repetition will then stand for a kind of "macro-incremental" learning step. As we can see, the application of either batch or incremental learning mode is not obvious here. The model of "repeated batch" is, moreover, only a specific case of a more general situation: a coincidence of a body of knowledge (formulated either by a learning system or by a human expert) and a dataset, within a particular problem-solving task. This brings us close to another area of learning, namely to *knowledge revision*.

The difference of the two tasks can be, in a simple form, visualized by means of I/O diagrams. In Fig. 8, we can see the diagrams of incremental learning and of knowledge revision; the former differs from the latter in the presence of loop - output problem solving knowledge enters the learning process immediately as input, as the overall learning task consists of multiple micro-learning tasks, one per example.

2.1.2 Knowledge revision

Among the terms at our disposal, we have chosen that of *knowledge revision*, as we view it as the most neutral one. The name of *theory revision* is typically used in ILP circles and somewhat implies that the input knowledge has the form of a (first-order) theory consisting of formulae. The name of *knowledge refinement* is sometimes put as its counterpart in the context of (production-) rule-based systems [CraSle90], [CraSlB94], [Sleem95]. The term *refinement* might be however a little restricting as it might be understood as covering only the process of *specialization*; comprehensive revision techniques however involve *generalization* as well.

A unifying idea of all approaches falling under this heading is the starting situation: we have a body of problem-solving knowledge and a set of examples or data. As the knowledge does not perform well enough, we adapt it to data. In this respect, the knowledge revision

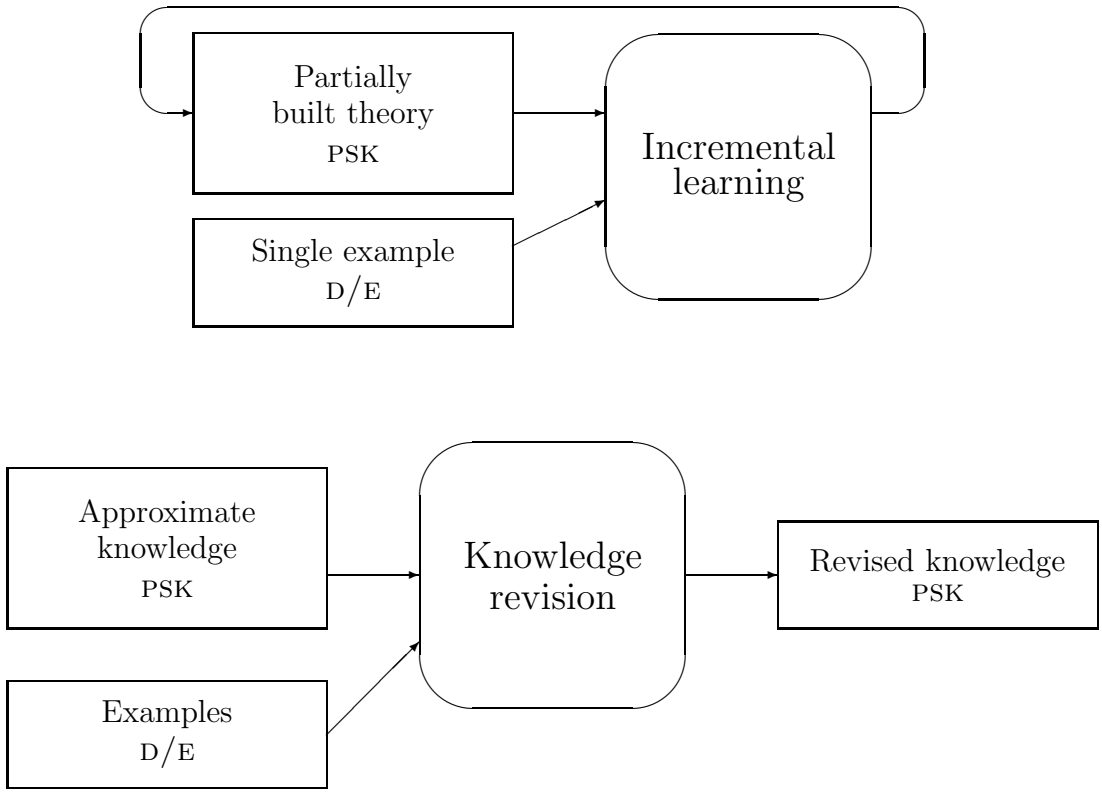

Figure 8: I/O diagrams of incremental learning and knowledge revision

process ressembles the process of empirical validation and subsequent modification of any knowledge-based (or even conventional software) system; the difference lays in the fact that the process is not conducted by a human (knowledge engineer) but by an autonomous learning system. The degree of autonomy may be various.

Knowledge revision systems can be categorized across several, rather interdependent, axes. Let us mention here at least the following:

- The degree of interaction with the user (possibly domain expert).
- The way how examples/data are used to control the revision.
- The robustness with respect to noise in data.
- The language of the knowledge base.

The first criterion discerns between *interactive* and *autonomous* revision systems. Interactive systems propose (often several alternatives of) revisions to the user who decides; autonomous systems rely on embedded heuristics. Many systems belong to one or to the other group, depending on parameter settings, e.g. the knowledge revision tool (KRT) of the MOBAL system [Wrob94a]. Interactive revision is often used in connection with first-order representation[15], to limit the huge search space, i.e. in the ILP system CLINT (see e.g. [LavDze94]) or in the APT learning apprentice system [NedCau92].

The second criterion discerns between *data-driven* and *generate-and-test* systems[16]. Data-driven (or, sequential) systems ressemble incremental learning systems as they usually process *one* example at a time and adapt the knowledge to it; some precaution must be kept so as not to disturb the consistency with examples seen previously. Generate-and-test systems first generate several alternatives to the current knowledge base (or to a part of it) and then evaluate them on *all* (or, at least, large portion of) data, in a "statistical" manner; the generate-and-test cycle may be repeated several times, but the number of cycles is usually much lower than for sequential systems.

Traditionally, research concentrated on the task of revising an approximate theory using presumably *correct* data. Some recent methods can, however, cope with *noisy* data in a way similar to pruning in standard machine learning systems (e.g. decision tree learners [Quinl87]). An advanced version of the EITHER system, by Mooney and Ourston [MooOu91a], thus fights noise by avoiding making changes to the theory that

[15]Craw [CraSlB94] claims that "external" sources of knowledge are typically exploited in refinement of "real" expert systems' knowledge-bases (which are mostly expressed in propositional-level languages), and views the (more academic?) ILP systems as autonomous (at the cost of needing more training examples). In fact, (at least optional) interactivity of learning systems seems to become necessity for advanced systems, be it for the reasons of language complexity or for pragmatical demands of safety-critical real-world applications.

[16]The distinction was first mentioned by Mitchell [Mitch82], as a dichotomy of empirical learning systems in general. Actually, the class of knowledge revision systems can be viewed as subsuming the class of systems learning without prior problem-solving knowledge, as it is possible to "revise" even an empty body of knowledge.

account for a small amount of data; only if the respective part of a rule is in conflict with multiple examples, it may be revised. The LATEX system by Tangkitvanich and Shimura [TanShi93] uses a sophisticated credit-balancing criterion based on coding length - the minimal description length (MDL) principle.

The *language* of the knowledge base varies from purely propositional, e.g. in DUCTOR [Cain91] or EITHER [MooOur93], through attribute-value representation (which still has the propositional power only) in knowledge-base refinement systems like SEEK [GinWeP88] or KRUST [CraSle90], [CraSle91], to restricted-first-order systems like FORTE [RicMoo95], MOBAL [MorWrK93], CLINT, and other ILP systems.

Knowledge revision is also related to *explanation-based learning*, which reformulates a body of knowledge, expressed as a set of logical clauses (which can be viewed as rules) so as to make it operational and cover a certain goal concept. As this reformulation is guided by examples, it can be viewed as *generalization* of these examples - therefore, most of existing research falls under the concept of *explanation-based generalization* (EBG) [MitKeK86], [DeJMoo86]. Unlike knowledge revision, EBG is deductive, truth-preserving, and requires input knowledge to be complete and consistent. Another related approach is that of *apprenticeship learning*, which consists "refining and debugging the knowledge base of an expert system during the course of problem solving" ([ShaDie90], p.627).

Among the high number of existing knowledge refinement systems, we have chosen two examples: the KRUST system, which refines a production-rule knowledge base, and the revision component of the MOBAL system, which refines a Horn clause theory in interaction with the user.

Refinement of a production-rule knowledge base in KRUST

KRUST [CraSle90], [CraSle91] refines a knowledge base consisting of IF-THEN production rules ordered according to priority. Each rule has a conjunction of attribute-value pairs in its condition. Only one rule can fire for a given example, namely that with highest priority among all *enabled* rules (rules whose condition is satisfied by the example).

The refinement process starts when a rule fires for an example and suggests a conclusion C_R while the expert suggests a different conclusion, C_E. KRUST then generates a large number of refinements. Each refinement consists in a minimal change, i.e. a minimal step needed for the example to be classified correctly, and is one of the following:

1. modify (usually generalize) the condition of some rule with conclusion C_E so that it can fire for the example,
2. change the conclusion of the error-causing rule to C_E,
3. modify (usually specialize) the condition of the error-causing rule so that it can no longer fire for the example,
4. lower the priority of the error-causing rule,
5. increase the priority of some rule with conclusion C_E,

6. perform both 1 and 3, or 1 and 4,

7. insert an entirely new rule, which specifically covers the example.

This large set of examples is then passed through two subsequent filtering phases. During *refinement* filtering, all refinements which e.g. contain contradictory combinations of changes, change good rules or contain unlikely changes are removed. The remaining refinements are implemented, which may still result in a high number of alternative knowledge bases. Then, *knowledge-base* filtering is applied, which removes all knowledge bases which do not correctly classify the example (this may occur due to rules' interaction), and also knowledge bases which do not correctly classify some of the small set of "chestnut" cases defined by the expert. After these two phases of filtering, only a relatively small number of knowledge bases remains, which is eventually tested on a larger example set, and the knowledge base with best accuracy is selected.

As the main advantage of KRUST, its authors view the possibility to exploit multiple alternative refinements while keeping the complexity reasonably bounded (thanks to heuristics and "chestnut" cases). In [CraSle91] it is shown that this approach, among other, enables to reconstruct a knowledge base which has been corrupted, as the original form is (almost) sure to be generated as "refinement" and is very likely to survive the filterings.

Interactive knowledge revision in MOBAL

MOBAL [MorWrK93], [Sommer94] is a complex tool for incremental and interactive knowledge modelling based on first-order logic[17]. Modelling can have the forms of manual input from the user, consistency checking, as well as autonomous derivation of new knowledge. The system has an inherently modular structure; however, individual modules interact on the same items of knowledge, their impact thus being conflued from the user's point of view. In this work, we will focus on two modules: the knowledge revision tool KRT and the concept learning tool CLT (both in this section), and the rule discovery tool RDT (in section 2.2.2).

Later, in section 2.2.2, we will describe the empirical learning component of the MOBAL system (RDT), in more detail. However, learning in MOBAL may also have the form of *revision*, which is performed by means of KRT - the knowledge revision tool. When the user indicates that a (ground) fact that has previously been derived by MOBAL's inference mechanism is not valid, the revision process starts. First, the user is shown the derivation graph of the spurious fact, i.e. the set of all facts and rules participating on the derivation; each rule has an associated confidence value computed as the ratio of known applications and known exceptions. Based on the graph, the system builds a list of the so-called *minimal removal sets* ([Wrob93], [Wrob94a]); each of them may consist of

[17]It is actually Horn-clause logic, with some restrictions (absence of function symbols) and some extensions (disjunctive clauses as integrity constraints, higher-order structures...).

a number of facts and rule applications (i.e. rules with terms substituted for variables).[18] The user selects one of these sets, the elements of which are then removed. The removal of facts is straightforward; to remove rule application/s, the respective rule/s have to be *reformulated*.

For each overly general rule, KRT attempts to apply, in turn, four reformulation operators. From the simplest to the most complicated, these are the following [Wrob94c]:

Minimal specialization Adding the forbidden applications to the list of global rule's exceptions (this is always the first, though often insufficient, step).

Localization Making the global exceptions local, i.e. constraining a single variable of the rule.

Addition of an existing predicate An existing predicate is added to the rule as a new premise, sharing some variable/s with existing premises.

Concept formation An entirely new (usually unary) predicate is formed to be, after the user's approval, added to the rule.

The first two operations are performed by KRT itself, the third requires inductive learning performed by means of a learning module (RDT, see section 2.2.2), and the fourth requires both RDT and the concept formation module CLT (concept learning tool).

As an example, consider the following rule[19], which states that the owner of a vehicle involved in a traffic violation is always responsible:

```
r1: involved_vehicle(X,Y) & owner(Z,Y) --> responsible(Z,X)
```

As the rule has no exceptions, its *support set* - the set of allowed instantiations of left-hand side variables - is unrestricted, i.e. `all * all * all` (one `all` symbol for each of `X`, `Y`, `Z`). Using this rule and the facts

```
involved_vehicle(event1,abc_45-56).
involved_vehicle(event2,abc_45-56).
involved_vehicle(event3,gr_12-34).
involved_vehicle(event4,ble_85-35).
involved_vehicle(event5,ai_64-65).
involved_vehicle(event6,ai_64-65).
owner(john,abc_45-56).
owner(peter,gr_12-34).
owner(david,ble_85-35).
owner(martin,ai_64-65).
```

[18] The list is ordered from removals which affect the knowledge base "the least" to those which affect it "the most", according to a so-called *two-tiered confidence model*; this model consists of discrete confidence classes as well as of a continuous confidence measure [Wrob94b].

[19] This example is taken from [Wrob94c] and slightly modified.

MOBAL derives

```
responsible(john,event1).
responsible(john,event2).
responsible(peter,event3).
responsible(david,event4).
responsible(martin,event5).
responsible(martin,event6).
```

Let us assume that the user rejects the last three of the conclusions (i.e. claims that David and Martin are not responsible for the respective violations), and indicates that rule `r1` is to blame for all of the improper derivations. Then, KRT first adds three global exceptions to the rule:

```
r1: involved_vehicle(X,Y) & owner(Z,Y) --> responsible(Z,X)
    all * all * all \ {(event4,ble_85-35,david),
                       (event5,ai_64-65,martin),
                       (event6,ai_64-65,martin)}
```

In this form, the rule has only three positive applications, and three exceptions; for standard parameter setting, reformulation of such rule will be required.

The first attempt of reformulation consists in *localization*. We will assume that the inapplicability of the rule is, for each of the three instantiations, due to a single variable. There are thus three possible reformulations of the rule (we list only the support sets as the rule itself remains the same):

```
(all \ {event4,event5,event6}) * all * all
all * (all \ {ble_85-35,ai_64-65}) * all
all * all * (all \ {david,martin})
```

In the next reformulation attempt, MOBAL will try to find an *existing predicate* to be added to the rule as additional premise. The inductive learning module RDT (cf. section 2.2.2) is invoked with a set of partially instantiated rule models (metapredicates), such as:

```
P(X) & involved_vehicle(X,Y) & owner(Z,Y) --> responsible(Z,X)
```

Let us assume that, in the knowledge base, there are also predicates determining the sort of violation, with facts

```
parking_viol(event1).
speed_limit_viol(event2).
speed_limit_viol(event3).
driving_drunk(event4).
driving_drunk(event5).
collision(event6).
```

Then RDT proposes to replace the predicate variable `P` in the above rule model with `speed_limit_viol`, yielding the rule

```
speed_limit_viol(X) & involved_vehicle(X,Y) & owner(Z,Y)
     --> responsible(Z,X)
```

Such a rule would have two positive applications and no exceptions, but the valid conclusion `responsible(john,event1)` would no longer be derived. In addition, the reformulated rule is more complex (i.e. longer) than the original rule.

As a final resort, MOBAL can apply *concept formation* using the concept learning tool (CLT)[20]. Let us assume that a new concept (i.e. unary predicate) `c1` is invented by CLT, which regroups three of the instantiations of `X` in the rule, `event1`, `event2` and `event3`. The facts

```
c1(event1).
c1(event2).
c1(event3).
```

are added to the knowledge base, and the rule is reformulated into

```
c1(X) & involved_vehicle(X,Y) & owner(Z,Y) --> responsible(Z,X)
```

which provides all required derivations and none of the incorrect ones.

Furthemore, concept `c1` is *characterized*: rules having it on the right-hand side are sought by means of RDT[21]. The following rules could possibly be found:

```
parking_viol(X) --> c1(X).
speed_limit_viol(X) --> c1(X).
```

The instances of the concept, together with the characteristic rules above, are finally presented to the user, who may e.g. replace the default name of the concept (`c1`) by a more informative one, such as `minor_violation`. The ultimate form of the reformulated rule, obtained thanks to the united forces of system and user (Morik [MorWrK93] uses the term *balanced cooperative modelling*) is

```
minor_violation(X) & involved_vehicle(X,Y) & owner(Z,Y)
     --> responsible(Z,X)
```

i.e. "the owner of a vehicle involved is responsible for minor traffic violations".

[20] As the concept formation method of MOBAL is fairly complex, we describe it here in an extremely simplified way; details can be found in [Wrob94a] or [Wrob94c].

[21] Actually, the characterization involves also search for rules having the concept predicate in the left-hand side; this subsequently enables to restructure the whole knowledge base, see [Wrob94c].

2.1.3 Knowledge integration

The notion of *knowledge integration* has been traditionally used in the context of distributed AI, namely in multi-agent learning, when the knowledge collected by separate agents is to be put together in a consistent and efficient manner. The approach suggested by Brazdil and Torgo (see [BraTor90], [TorKub91]) orders rules learned on different subsets of data according to their quality wrt. the whole dataset. The quality is computed from the *consistency* (number of objects satisfying both sides of the rule, divided by the number of objects satisfying the left-hand side) and *completeness* (number of objects satisfying both sides, divided by the number of objects satisfying the right-hand side). A scheme of knowledge integration as it is understood in distributed AI is on Fig. 9. The input to the integration task, in this simple form, thus consists in multiple bodies of knowledge and possibly the source dataset, which however serves only for rating the pieces of knowledge to be integrated.

Empirical studies (e.g. [SikSha92]) have shown that distributed learning may lead to better results than single-agent learning, since different agents can use their different biases with greater chance of fitting the target patterns hidden in data. Some authors claim that combining (integrating) knowledge learned on partitions of data may be efficient even if an identical learning algorithm is applied in all learning subtasks. If the different batches of data provide some variation of data representation in the description space, individual theories induced become "specialists" in different parts of the space [TinLow97]; the so-called model combination (which is more or less identifiable with knowledge integration) then allows cooperation between these specialists. In the above paper, a hypothesis has been raised (and partially verified) that for small amounts of data, *data combination* (prior to learning) is the better choice, while for large amounts of data, *model combination* is worth trying.

2.2 Learning with prior static domain knowledge

2.2.1 Bias and background knowledge in empirical learning

As the name suggests, *empirical learning* is a reasoning process, in which specific observations (examples, instances, data...) are transformed into more general knowledge. This phenomenon has been studied by many disciplines; however, it has acquired particular attention as the key topic in machine learning. There, it has been also called (more-or-less interchangeably) *inductive learning* (in contrast to deductive approaches such as explanation-based generalization) or *learning from examples/data*. For a long period, examples or data were considered as the only input to the empirical learning task. Yet, it became obvious that "pure empirical learning" as such would not give valuable results: as induction is not truth-preserving, an extremely high number of examples would be needed to confirm the validity of knowledge acquired.[22] As a matter of fact, empirical learning systems have, from the beginning, taken advantage of some implicit information

[22]This is (informally presented) one of important outcomes of research on the so-called PAC (probably-approximately-correct) learnability theory, cf. e.g. [Vali84].

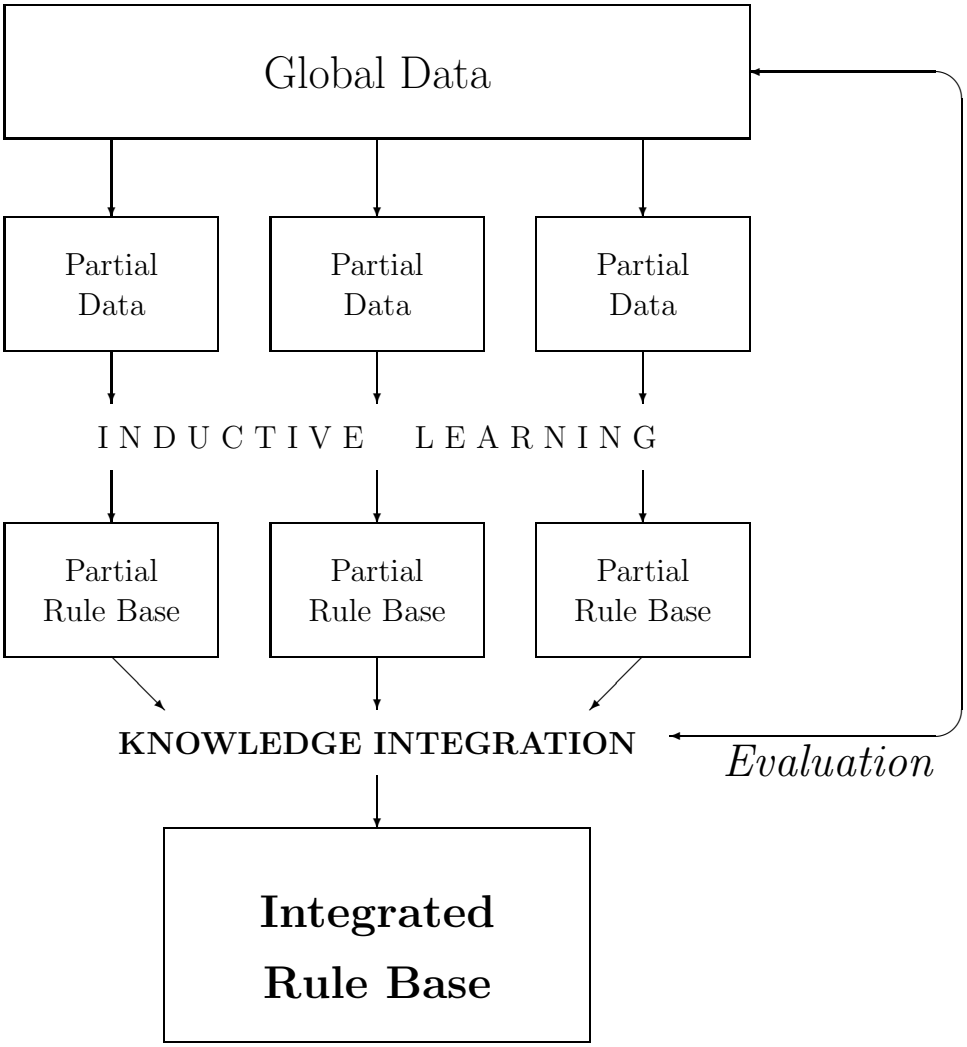

Figure 9: Knowledge integration in distributed AI

that helped them to focus on relevant types of knowledge only. This information, denoted as learning *bias*[23], has later been subject to thorough study.

Attempts have recently been made to convert implicit biases of existing and new learning algorithms into explicit, *declarative* biases. Several (especially European) ML research projects have provided existing families of ML algorithms with abstract, knowledge-level models (cf. section 1.1.2), in order to facilitate their integration into complex problem-solving architectures. This "knowledge-level analysis" has been already performed e.g. for explanation-based learning [VVelde94] and for certain types of empirical learning systems, such as TDIDT (top-down induction of decision-trees) systems [Slodz94] or generate-and-test learning systems [RouAlb94], [NedRou94].

Most authors (e.g. [GorDeJ95]) suggest the following typology of biases:

1. *Representational* biases defining the states in the search space. They consist of two parts: set of *primitive terms* (allowable features, their types and range of values), and *language* of hypotheses (e.g. first-order predicate calculus or DNF expressions).

2. *Procedural* (or algorithmic) biases determining the order of traversal of the states in the space defined by a representational bias.

3. *Instance* (or sample) biases determining the selection of instances (examples, data objects...) which enter the learning process.

In the context of the MLNet Familiarization Workshop on Declarative Bias, in 1994 [DecBia94]), a similar typology has been formulated:

1. Language biases *restricting* the hypothesis space.

2. Biases which guide the *search* in the hypothesis space.

3. Biases which *evaluate* hypotheses.

Obviously, the notion of language bias is closely related to that of representational bias from the previous list. We can assume that a certain initial representational language (e.g. attribute-value formalism) exists; what is called language bias here is actually a restriction of the set of primitive terms and/or of constructors of expressions (e.g. decision trees). On the other hand, the constructors of expressions can be viewed as a bias *extending* the basic hypothesis space, as it is in *constructive induction* (cf. section 2.2.3). It is a matter of discussion whether we should view the initial language as bias, or whether this notion should be reserved for specific restrictions and extensions.

The remaining two categories (search and evaluation biases) can be viewed as refinements of the notion of procedural bias.

[23]As the perhaps most universal definition, we can introduce that given by Gordon & desJardins [GorDeJ95]: learning bias is "any factor (including consistency with the instances) that influences the definition or selection of inductive hypotheses." Even this definition does not cover some particular types of bias, especially the bias influencing the selection of instances (sample bias).

Language (representational) biases and search (procedural) biases are closely interrelated. Particularly, a rich (i.e. weakly biased) representational language enforces the use of strong search biases to keep the complexity of learning reasonably bounded. It is the reason why nearly all inductive logic programming systems rely either on large amounts of background knowledge, or on user-given hints (i.e. work interactively), cf. [LavDze94].

We can imagine several possible *sources* of the bias. First, it can be (in particular, for representational bias) the inherent principles of the learning algorithm, incurred by the *representational language*. A classical restrictive bias of symbolic, attribute-value learning algorithms is thus the assumption of orthogonal concept boundaries, i.e., in the geometric interpretation, concept descriptions are understood as axis-parallel hyperplanes in a space where each coordinate maps to an attribute[24]. This bias is, on the other hand, relaxed[25] by most statistical, neural-net and case-based methods (see [DecMer94]) as well as by certain decision tree methods which learn the so-called multivariate (also called "oblique") decision trees ([MurKaS93], [BroUtt95]). Even further, concept boundaries may be not only non-orthogonal but even non-linear, especially in neural network representation. In this way, even a simple backpropagation neural network can recognize so-called *m-of-n concepts*, which are beyond the scope of most symbolic learning algorithms (cf. e.g. [Thrun91] for empirical comparison). Disposing the orthogonality assumption may improve the "fitness" of a concept description, but usually reduces comprehensibility and incurs computational overhead. The implicit bias for comprehensibility may, on the other hand, requisite the use of close-to-natural-language representations, such as predicate logic, if the character of the domain favourizes it.

A second source of bias is the already mentioned domain-independent *principles of reasoning*, such as the Occam's razor or entropy minimization, which can be "grafted" on different learning techniques and knowledge representations; these mostly generate search and evaluation biases. Quite often, they rely on the achievements of statistics and information theory, including e.g. statistical significance testing, average error minimization, or entropy and information gain criteria (as examples, see e.g. [IvaSte88], [Quinl86], [Quinl93]). A closely related field is that of description *minimality*, involving the use of the Occam's Razor [BluEhH87], minimal description length (MDL) [TanShi93], [Quinl94], [Pfahr94] and minimal message length (MML) [OliBaW96] principles. Some methods combine both approaches. Last, we should not omit heuristic "rules-of-thumb"[26], which many practitioners prefer for their simplicity and immediate applicability.

Yet another, third, source of bias, may be the *user* himself, who can tune the parameters of the algorithm so as to adapt it to the task at hand. What is most important in the context of this work, is however the fourth source: explicit, domain-dependent *background knowledge*.

[24]Matheus [Math91] denotes this type of bias as *logical-representation bias*.

[25]This is obviously only possible if the domains of attributes are numerical.

[26]An example of such rule without theoretical justification is given in [Pfahr94]; it is however not universal but pertaining to the domain of medicine - the so-called 3-σ heuristic which suggests a test result to be pathological if its value differs from the mean value of the healthy population by more than three standard deviations...

As *implicit* background knowledge we can understand even such indispensable inputs to empirical learning as attributes by means of which the data are described; the success of simple learning algorithms on many "benchmark" datasets was actually due to the "friendly nature" of datasets, in which most attributes were relevant with respect to class discernment.

Explicit background knowledge, on the other hand, is truly separated from data; it is an extra input to the learning process. If we consider the types of input knowledge we want to distinguish in I/O diagrams (cf. section 1.3), we can more-or-less identify background knowledge with *static domain knowledge* and *meta-knowledge*. The remaining type, *problem-solving knowledge*, can be hardly labelled as "background", as it is syntactically akin to target, "foreground" knowledge[27].

Background knowledge often does not satisfy the characteristic property of "expert knowledge", i.e. *scarcity*. Instead, it can fall upon the heading of either the following types:

- Specific but *rudimentary* knowledge, which is possessed by anyone who has worked in the domain even for a short period of time.
- General *common-sense* knowledge, for which no in-depth acquaintance with details of the domain is necessary.

Background knowledge can enter the learning process:

1. Before learning actually starts
    - to verify the consistency of data and possibly remove inconsistent, redundant, or irrelevant data objects ([VSom89] denotes this type of background knowledge as "factual knowledge");
    - to extend given data with newly constructed attributes or predicates.
2. During learning (i.e. search for concept descriptions)
    - to constrain the search and to guide it by means of preference criteria;
    - to define a stop-condition which suggests when to terminate the search.
3. After the tentative concept descriptions are found - to validate them against an external global criterion.

[27]Input problem solving knowledge can be nevertheless viewed as a source of *search* bias. In knowledge-revision and incremental learning systems, the search usually starts in the neighbourhood of input (imperfect) knowledge - this corresponds to the general principle of *minimal change*, incarnated e.g. in the *minimal base revision* operation of the MOBAL system (section 2.1.2). The situation is still different in inductive logic programming, where the distinction between problem-solving and static domain knowledge (and even data) dissolves in the uniform Horn-clause formalism.

We can thus see that all types of bias (cf. the list on page 40) can be provided to the learning algorithm via explicit background knowledge.

In our overview, we will make a distinction between *restrictive* and *constructive* ways of use of background knowledge, and thus between constrained and constructive induction. The first is treated in section 2.2.2 and the second in section 2.2.3.

2.2.2 Constraining learning with explicit knowledge

Assuming that output knowledge has the form of rules, background knowledge determines/suggests which description elements (attributes, predicates...) should occur in the left-hand sides and right-hand sides of the rules, and which examples/cases are to be used to guide and evaluate the search; this form of inductive learning can be called *constrained induction.*[28] As examples of use of background knowledge for constraining the search, we will present two approaches which have lately received particular attention, especially due to their effort to put together the viewpoints of machine learning and knowledge acquisition - MOBAL and ENIGME. Other approaches will be mentioned only very briefly.

MOBAL - learning as one of interactions among knowledge items

MOBAL [MorWrK93], [Sommer94] is a complex tool for incremental and interactive knowledge modelling based on first-order logic[29]. Modelling can have the forms of manual input from the user, consistency checking, as well as autonomous derivation of new knowledge. The system has an inherently modular structure; however, individual modules interact on the same items of knowledge, their impact thus being conflued from the user's point of view. The *rule discovery tool* RDT serves for derivation of rules from facts[30]. This is the main problem in inductive logic programming, which is known to be computationally intractable in its pure form. In MOBAL, the state space of potential rules is limited by means of two types of "background knowledge": *abstract rule models* and *predicate topologies*. These two types of knowledge are written in full rectangles[31] in the I/O diagram, in Fig. 10. However, the use of RDT is not restricted to learning rules "from scratch"; it has an important role in concept formation and thus in knowledge revision (cf. section 2.1.2), which is interactive. The impact of the user and of the imperfect input rule is mediated by the knowledge revision tool (KRT - see section 2.1.2) - it is thus marked as dashed in the diagram.

Abstract rule models - *metapredicates* - can be derived from existing rules by replacing predicates with predicate variables (and possibly constants with constant variables). As

[28] This constraining process can be possibly viewed as a sequence of selection and projection operations on a relational database, i.e. on the input dataset.

[29] It is actually Horn-clause logic, with some restrictions (absence of function symbols) and some extensions (disjunctive clauses as integrity constraints, higher-order structures...).

[30] MOBAL is also able to cooperate with learning tools other than RDT, e.g. FOIL [QuiCam93] or GOLEM [MugFen90].

[31] The remaining input marked by full rectangles (i.e. as "direct" input) is the (ground) *facts*.

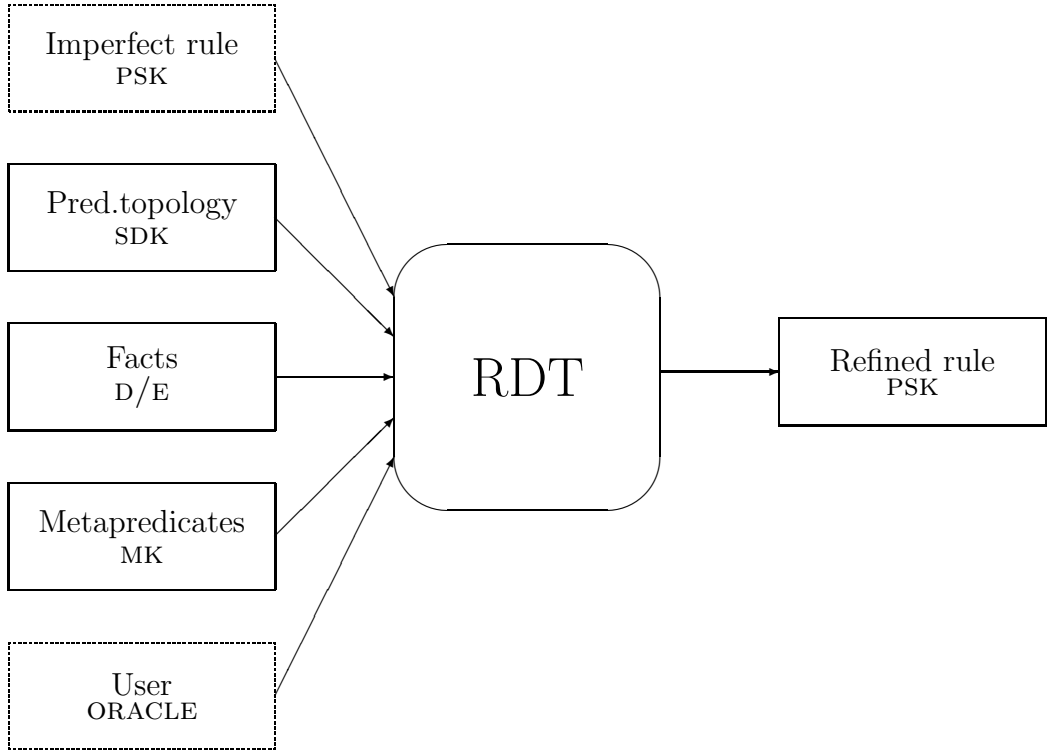

Figure 10: I/O diagram of MOBAL's rule discovery tool

an example, we can take a rule:

```
works_for(X,Y) --> company(Y).
```

expressing that if someone works for `Y` then `Y` is a company. The corresponding metapredicate, which can be generated automatically in MOBAL, is:

```
m1(P1,P2):P1(X,Y) --> P2(Y).
```

In addition to the metapredicate, a *metafact* is also derived, corresponding to the particular instantiation of the metapredicate `m1`:

```
m1(works_for,company).
```

Metapredicates can be generated or input by the user. When the RDT is invoked for a particular target predicate (namely, to find rules with this predicate on the right-hand side), it examines all metapredicates present in the knowledge base, and selects those with right-hand side of the same arity and argument types as the target predicate. The left-hand sides of these, compatible, rule models are then instantiated with existing predicates. For every tentative rule thus formed, validity is computed with respect to facts in the knowledge base. Rules which satisfy a certain, user-specifiable, plausibility criterion (based on the number of examples and counter-examples of the rule) are inserted into the knowledge base.

Predicate topologies are directed graphs, expressing semantic relations (namely the relation of "derivability") among groups of predicates. They can be used to constrain the rule induction in the following way: a rule is admissible wrt. a topology if all predicates from its premises belong either to the same topology node as the predicate of the conclusion, or to the predecessors of this node.

Unlike most machine learning systems described in this work, MOBAL is not designed to perform a single, however complex, transformation of one form of information into another (such as "data" into "knowledge"). The derivation relation among different types of its knowledge items is multiway, and automatically derived items are kept in the knowledge base at the same level with items input by the user. For example, the user can provide rules and some facts compatible with the rules' premises; the system then deduces new facts through modus ponens. On the other hand, new rules can be induced from both input and deduced facts with the help of metapredicates (as described above), and immediately used for further deduction; this feature is denoted as *closed-loop-learning*. The rules can be also generalized into abstract rule models; in addition, rules can be composed into a rule graph, from which a predicate topology can be automatically extracted. All this gives MOBAL a remarkable flexibility, suitable for incremental modelling of not-well understood, complex domains.[32]

[32]There is, presently, a discussion whether MOBAL suits for actual development of KBS, as its complexity together with close-loop-learning and a host of other "spontaneously-triggered" actions may lead to loss of control over the modelling process. Authors of the system however claim that MOBAL can be used e.g. for building the domain layer of the expertise model in the KADS methodology, cf. section 1.1.2.

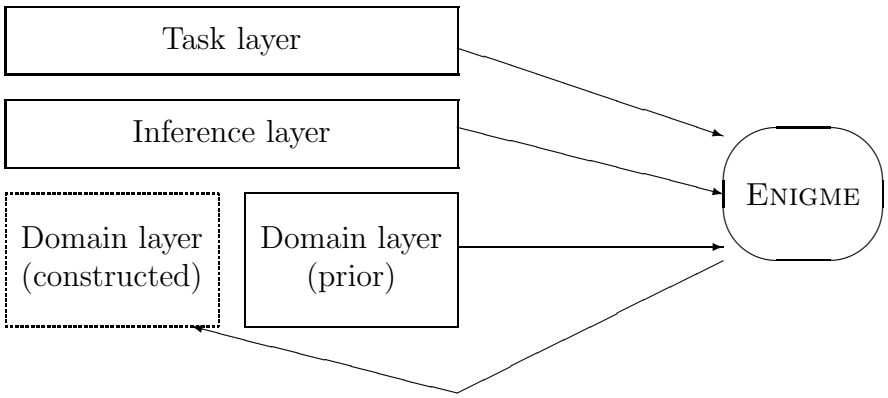

Figure 11: Interaction between KADS and ENIGME

ENIGME - learning constrained with models of problem solving

The approach taken by Thomas et al. in the ENIGME project [ThoLaG93], [GanThL93], embeds empirical learning into the KADS methodology of knowledge modelling (cf. section 1.1.2). Learning represents but a support task within the global task of building a knowledge-based application. On the other hand, knowledge previously acquired from the expert acts as a bias for the learning task. ENIGME uses a partial model of expertise, i.e. knowledge from the inference layer, from the task layer and from the "static" part of the domain layer (semantic networks) as a bias so as to learn the "dynamic" part of the domain layer (problem-solving rules), as depicted in Fig. 11.

ENIGME has been tested on a non-industrial but fairly complex problem: opening in the game of bridge. The process of opening selection is performed in several steps. It can be represented with the inference structure at Fig. 12, where ellipses correspond to inferences and rectangles to inference roles. Inferences in double ellipses (e.g. "compute") are not learnable within ENIGME, as they consist in numerical computation rather than in rule-based reasoning. The learning process thus concentrates on the remaining inferences, such that each corresponds to one learning step.

The inference structure by itself, however, does not determine the order of inferences. The control flow in the inference diagram is determined by the task structure, here written in pseudo-code (Fig. 13)[33].

The overall task is to find a solution (suitable opening) based on source data (the hand). In the application, several hundreds of "hand-opening" pairs were available. A naive approach to learning (not taking advantage of the model of expertise) would try to learn classification rules based on the examples. The number of observables that can be derived from a hand is however extremely high, this leading to barely tractable

[33]The pseudo-code of the task structure has been transcribed to the format used before in this thesis. In original work (e.g. [Thomas94]), a different but compatible format is used.

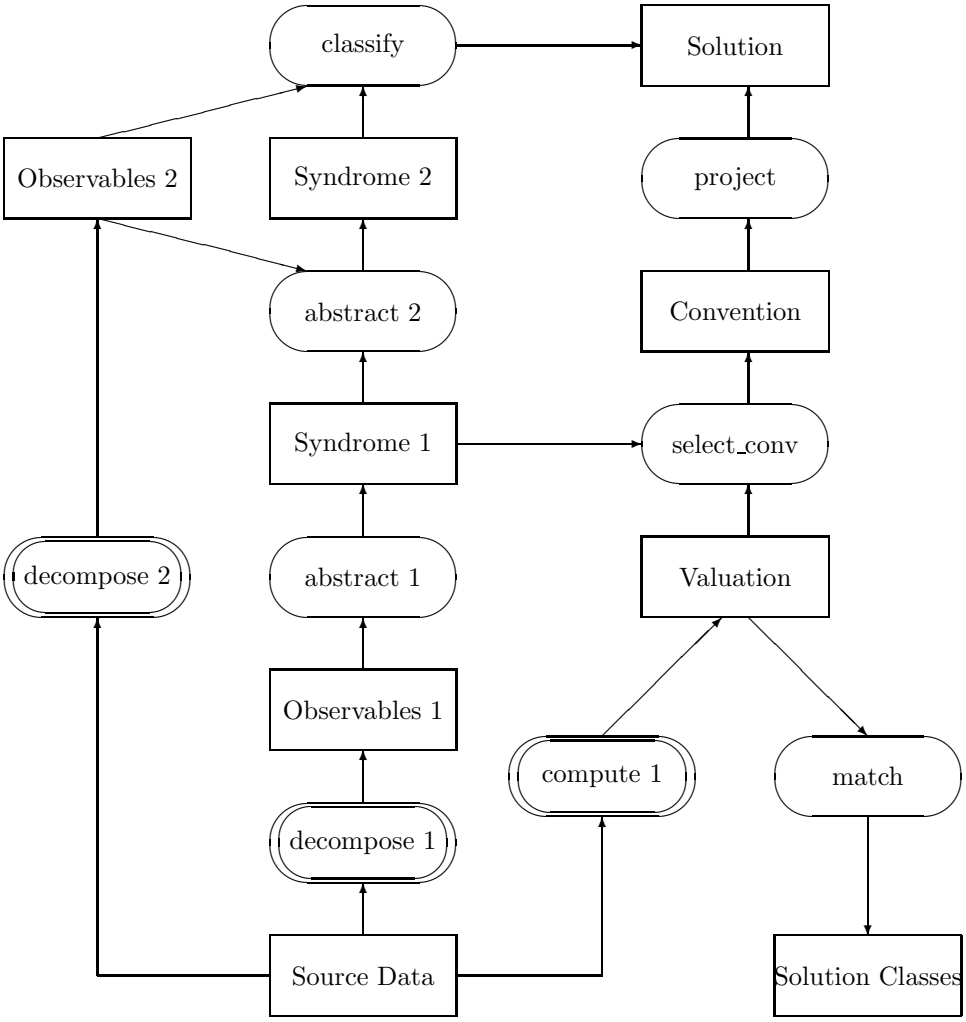

Figure 12: Inference structure of the bridge example

```
choose_opening(+Source_Data,-Solution) :-
     compute(+Source_Data,-Valuation),
     match(+Valuation,-Solution_Classes),
     if opening=yes
     then decompose_1(+Source_Data,-Observables_1),
          abstract(+Observables_1,-Syndrome_1),
          select_conv(+Syndrome_1,+Valuation,-Convention),
          if conventional-opening = club
             or conventional-opening = no-trump
          then project(+Convention,-Solution)
          else decompose_2(+Source_Data,-Observables_2),
               abstract_2(+Syndrome_1,+Observables_2,-Syndrome_2),
               classify(+Syndrome_2,+Observables_2,-Solution).
```

Figure 13: Task structure of the bridge example

search space. In addition, the rules learned will be flat, with overly complex (and thus incomprehensible) left-hand sides; also the accuracy will be low, as many potential observables are relevant only in *some* cases.[34]

The mechanism of ENIGME enables to learn the rules for each inference step separately. In each learning task, only the *input attributes* which are relevant wrt. *inference structure* are considered.[35] Similarly, only the *examples* which meet all test-conditions in the *task structure*, necessary for performing the particular inference, are considered; e.g., for the *project* inference, the composite test-condition is:

opening = yes `and`
(conventional_opening = club `or` conventional_opening = no-trump)

As a third possible bias, ENIGME can also use semantic networks (e.g. is-a hierarchies); their use is, in fact, a domain-layer counterpart to constraining by inference structures described above.

If we wanted to categorize the types of input knowledge in the way introduced in this work (section 1.3), we will see that the mapping is not obvious. Clearly, semantic networks from the domain layer describe static relations between concepts, and thus definitely belong to *static domain knowledge*. Further, the task structure is procedural and has the

[34]We can see by the task structure that some parts of the inference structure (and thus some inference roles) are not used to process every example. The decision to pass, for example, can sometimes be made based on very few findings, others being irrelevant.

[35]For example, in the left-hand side of rules realizing the *select_conv* inference, only the attributes mapping on *valuation*, *syndrome1* and *convention* inference roles are considered.

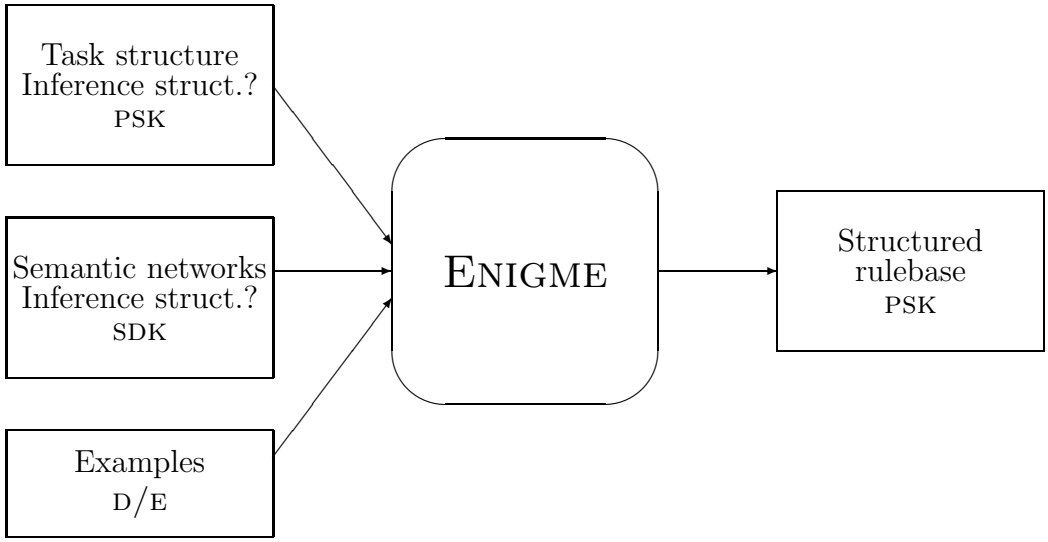

Figure 14: I/O diagram of ENIGME

character of *problem-solving knowledge* - although, due to its *knowledge-level* nature, it would not be directly applicable for bridge-playing and could be seen as "static" from the symbol-level perspective. The remaining type of background knowledge - inference structure - is the hardest to classify. On the one hand, it is immediately related to the task to be performed (and thus to problem-solving). On the other hand, it has declarative nature - the relation of "derivability" among domain roles can, in a sense, be considered as any other relation which hold in the domain. Among the inputs to ENIGME, there does not seem to be any which would have the character of *meta-knowledge*; the activity of user as oracle is not required, either. The whole situation is depicted in the I/O diagram at Fig. 14.

Constrained learning of ENIGME leads to a structured rule base, possibly with better accuracy (with even a low number of training examples), but certainly with better explanation capabilities.[36] Therefore, it seems to be useful in the context of development of "second-generation" expert systems, where modularity and comprehensibility play the key role. With this respect, ENIGME is currently being integrated into the VITAL knowledge engineering workbench (cf. e.g. [LeRoux94]), which supports a variation of the KADS methodology.

[36]Empirical results in terms of accuracy and comparison with "naive learning" approach are presented in [Thomas94].

Other approaches to constrained induction

The above methods were representative examples of the "constrained empirical learning" stream. We will mention other, similar, approaches only very briefly.

The approach taken by van Someren and his colleagues at the University of Amsterdam [VDoVSo94], [HelSom95] is similar to ENIGME in using the inference layer of the KADS model of expertise for biasing the learning of domain-layer rules; however, the induction does not have the form of batch learning "from scratch" but of *knowledge refinement* (cf. section 2.1.2). If a correct statement cannot be derived, its dependency tree is passed top-down, and potential gaps identified in dialogue with the user. Only the statements relevant wrt. the inference structure are considered. The missing rules are specified again in dialogue with the user - a sort of *version space* dialogue, where the final form is reached after a sequence of moves in the generalization lattice. There are two variations of the algorithm: the ML-3 system [VDoVSo94] can refine incomplete knowledge bases automatically, under the single-fault assumption; the system described in [HelSom95] can also repair incorrect knowledge bases via specialization, and handles multiple faults, these capabilities being traded off by increased interaction with the user.

The idea of splitting the set of attributes/examples for batch learning, central for ENIGME, has also been raised by Bruha [Bruha95]. There, the split is achieved by means of *attribute hierarchy trees*, i.e. decision trees formulated by the expert. Some leaves of the tree may be final, while other represent a learning task, which is solved using only examples matching the branch and attributes not present in the branch. Here, prior knowledge is at the same degree of abstraction as the knowledge to be learned.

Clark & Matwin [ClaMat93], on the other hand, constrain learning with a distinct type of knowledge: *qualitative models*. Qualitative models link the concepts (quantities) of the application domain and indicate the influence relations, such as "the derivative of quantity Y (dY/dt) varies monotonically with quantity X"[37]. From the relations in the qualitative model, *rule schemata* are extracted, such as (for the above simple example): "if $X > k$ then Y will increase"; k is a constant to be instantiated when converting the rule schema into an operational prediction rule. Rule schemata are stored in a look-up table. Then, a standard inductive tool - the CN2 rule learning system [ClaNib89] - is applied on a set of training examples; the search is limited to prediction rules consistent with existing rule schemata. The qualitative-model approach has been successfully applied in engineering domains (with qualitative models of physical behaviour) [ClaMat93] and in the domain of economics (macro-economic qualitative model) [ClaMat94]. In the economics application, even the mapping between qualitative relations and formats of prediction rules (such as the operational definition of "high" GNP) was not given in advance; instead, two types of rules - qualitative prediction rules (such as "gnp is high → rates will increase") and

[37] Unlike qualitative simulation models, Clark & Matwin's qualitative models can be used only statically, for constraining rule learning; they are incompletely specified for use on their own.

definitions of quantities (such as "$GNP > 7\% \rightarrow$ gnp is high") are induced[38]. Qualitative models, similarly to knowledge-level descriptions, provide for better explainability of rules found via induction.

Brazdil & Jorge, in their ILP[39] system SKIL [BraJor93], exploit what they call *algorithm sketches* - generalized computation graphs of examples of the target predicate, which suggest some dependencies among predicates and their arguments. This particular type of knowledge is formulated by the user, who can (for his/her convenience) omit some details, such as predicate names, values, or links between predicates. Algorithm sketches are fed as input to the learner, together with examples (which can be fewer than is usual for ILP systems) and standard background knowledge (i.e. definitions of predicates available for use). The sketches considerably limit the space of possible literals explored in the search for the definition of the target predicate. If we compare their role in SKIL with the roles of rule models and predicate topologies in MOBAL, it can be viewed as "spanning" between both. Similarly to rule models, the sketches enable to keep "free slots" for unspecified predicates; similarly to topologies, they enable to convey information on predicate hierarchy. The authors claim that this flexibility is a way to scale up current ILP methods to the design of non-trivial programs.

As empirical learning constrained with background knowledge is one of hot topics for today's machine learning community, the number of methods and applications is already much higher than we could show in this overview, and is still rapidly growing. Some more references can be found e.g. in [VDoVSo94], [ClaMat93] or [Thomas94]. Part of the original research in this thesis (section 5) may also be ranged under this heading.

2.2.3 Shift of bias and constructive induction

Making the bias declarative, in the form of explicit background knowledge, is one step toward more flexible and comprehensible learning systems. Another achievement of up-to-date machine learning research was the introduction of *bias shift*. The bias may not necessarily be fixed for a given learning task but can change in order to comply with the peculiarities of the task, which are possibly discovered as late as in the middle of the learning process. According to [GorDeJ95], bias shift can be viewed as *search* at the bias level, similarly as learning can be viewed as search at the level of problem-solving knowledge.

[38] This brings up the problem of mutual dependency - learning of one type of rule depends on the other type.

[39] In our survey, we otherwise omit the area of inductive logic programming (although e.g. MOBAL also has some aspects of ILP). In "genuine" ILP, background knowledge seems to be indispensable; however, its position is different from most methods mentioned above. ILP systems, such as FOIL [QuiCam93], use background knowledge in the form of definitions of predicates that act as "building blocks" of the target predicate definition (which is to be learned). Rather than constraining the search space, background knowledge in ILP *forms* it. Clark [ClaMat93] draws the dichotomy between the two types of background knowledge even further; he claims that in ILP, background knowledge actually *expands* the hypothesis language, thus aggravating the search problem. The state-of-the-art of ILP is well described in [LavDze94].

An important notion, in this context, is that of "hard" vs. "easy" concepts to be learned [RenRag93]. A concept can be viewed as "easy" if it is "localized in the instance space", i.e. if it occupies a contiguous region in the space defined by values of input attributes (in attribute-value learning); it can be thus represented by a few rules with simple left-hand sides (possibly containing only conjunctions of attribute values). "Hard" concepts, on the other hand, are "spread out", and often unlearnable by standard ML algorithms.

"Hardness" of the concept however depends not only on the nature of the task itself, but also on the *representation* of data (i.e., language bias). As an example, we can take the game of chess. Let us assume that we want to learn general rules for discernment between good and bad (or, winning, uncertain and losing) positions, given a set of examples. If we describe the positions merely by means of coordinates of pieces, the target concept is extremely "hard". If we, on the other hand, use some higher-level attributes, such as pieces advantage or center control, the target concept becomes much better learnable.

Constructive induction is an approach to empirical learning, which attempts to move the instance space "closer" to such "hard" concepts, by means of construction of new features (e.g. attributes) from those present in data[40]. Feature construction is the basic way of shifting the *representational bias*, and it is the form of bias shift which is most often performed by means of explicit prior knowledge.[41] According to [Pfahr94], the generic form of a constructive induction system is that of Fig. 15.

Given raw data, the constructive induction module constructs new attributes and applies them on the examples. The selective learner works as a usual inductive learner. Finally, the evaluator decides (according to some measure) whether the hypothesis (learned knowledge) is of sufficient quality, or another cycle of construction/induction is needed.

First constructive induction systems were "grafted" upon standard inductive learners, such as those from the AQ family [Michal83]. More recently, generic architectures, such as the CiPF system by Pfahringer [Pfahr94] have appeared, where each of the agents involved (constructive induction module, selective learner and evaluator) can be instantiated by different systems.

As representative examples of feature construction operations we can consider the spectrum implemented in CiPF [Pfahr94] (here, we simplify the whole list):

- comparison of values of attributes (equal-to, greater-than etc.) [42],
- discretization of numerical attributes,
- conjonction of possible values of nominal attributes into sets,

[40]Earlier, the notion of constructive induction was used more generally for "inductive processes that engage significant amount of background knowledge" [MicKod86].

[41]Some learning systems perform shift of *procedural bias* [GorDeJ95]; this type of bias however seems to be harder to formulate by means of declarative prior knowledge.

[42]Leng and Buchanan [LenBuc91] have developed a set-theoretic approach to this particular form of constructive induction.

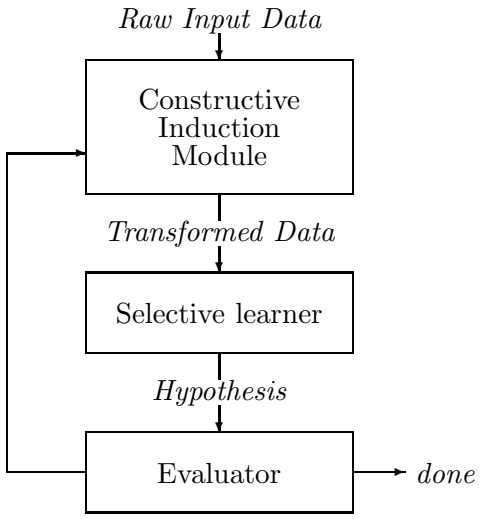

Figure 15: Generic architecture of constructive induction

- counting the number of true/false boolean attributes,
- operations on attributes occuring in "good rules" learned in the preceding learning step (e.g. conjoining or dropping attributes).

The construction operations can be embedded into the program code, but they can also be represented as prior knowledge. This knowledge has typically the form of *meta-knowledge* (in particular, knowledge dedicated for feature construction); some authors ([Math91], or perhaps also section 5 of this thesis) also mention *static domain knowledge* (cf. Fig. 16).

Constructive induction systems with a wide scope of operations have already shown efficient on (artificially contrived) tasks in which the nature of target concepts was obscured. This was the case of the so-called MONK's problems [Thrun91], which were distributed in 1991, as a "quizz", to many developers of ML algorithms. Each of the three problems consisted in a training and a testing set of examples, such that concept descriptions were to be induced from the training set and then evaluated on the testing set. Standard inductive learning systems have usually failed on the MONK2 problem which involved a *m-of-n* concept ("exactly three of the attributes have their first value"); constructive induction systems, on the other hand, have mostly captured the concept and thus attained reasonable accuracy on the testing set (the accuracy obtained by neural-net-based methods was even higher; there, however, the concept descriptions induced were completely opaque...).

In inductive logic programming, constructive induction has the form of invention of new predicates where existing ones are insufficient for building a definition of the target

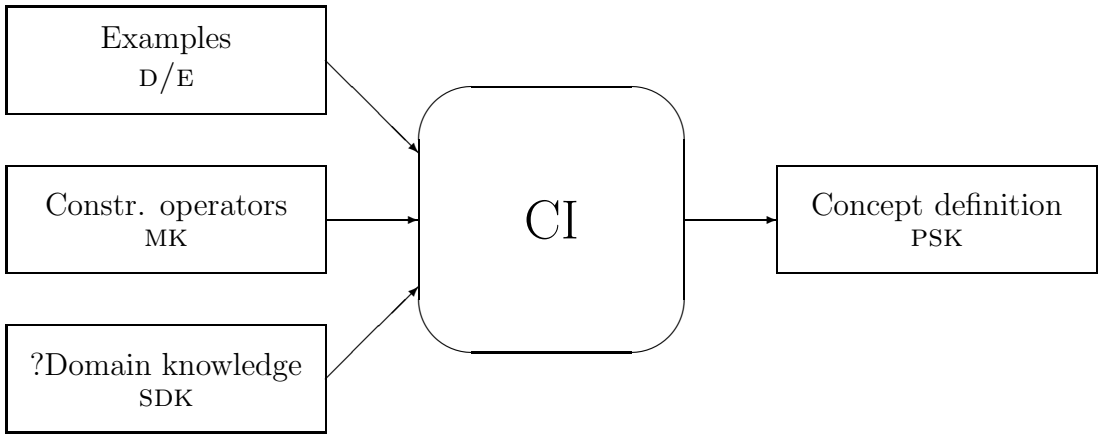

Figure 16: I/O diagram of constructive induction (with explicit knowledge)

predicate. Examples of this approach, sometimes denoted as *concept formation* can be found e.g. in [DeRLaD93] or [Wrob94c] (an example of predicate invention in MOBAL can also be found in section 2.1.2 of this thesis).

2.2.4 Value hierarchies and integrity constraints as static domain knowledge

In the previous paragraphs, we have reviewed some common approaches to learning with prior static domain knowledge. Here, we will narrow our focus to two particular forms of such knowledge - hierarchies of attribute values and integrity constraints - which have been subjected to original study in section 5 of this work.

The concept of *hierarchical attributes* and of "generalization as climbing up in a hierarchy" was notoriously mentioned in the mainstream ML literature - the "climbing-up-hierarchy" operation was known to be a generalization technique alternative to the common "drop-literal" one. Nevertheless, it was mainly used for "textbook", demonstrative purposes, in connection with small example sets (e.g. the notorious taxonomy of simple geometric objects). Only recently, attempts appeared to exploit hierarchies to discover important relations in *raw data*. This is the case of the RL knowledge discovery system [ClePro90], which has been used in several domains to discover useful relations; its more advanced successor KBRL [AroPrB96] uses role links in addition to taxonomies. The phenomen of tree-structured attributes has also been studied in the context of decision-tree learning, by Núñez [Nunez91] and Almuallim et al. [AlmAkK95], with the primary aims of cutting down computation time, increasing accuracy and decreasing the size of the tree learned (although comprehensibility was also mentioned as a secondary issue). In KDD, hierarchies (even non-tree dags) are used to aggregate data from large databases, along the so-called *domain generalization paths* [HamHiC96].

Abstraction hierarchies (and is-a relations within semantic nets) have been also accepted as a rudimentary form of domain knowledge for *relational systems*, such as the apprenticeship-learning systems DISCIPLE [TecKod90], APT [NedCau92] and ODYSSEUS [Wilk90]. Another related approach may be that of EITHER, a knowledge revision system, one version of which was adapted to constructive induction [MooOu91b]. EITHER works with chained rules on propositional concepts; some of these rules are similar to is-a links (e.g. "has_handle → graspable" in the "cup" concept). If such rules lead from observable features to intermediate features, the truth-values of the latter can be deduced for data objects by forward chaining, added to the descriptions of these objects and used as input features in the subsequent empirical induction process. However, EITHER's rules can be more complex (at least, they mostly involve conjunctive antecedents) and their semantic may only incidentially become that of concept generalization.

Integrity constraints are commonly used in *deductive database* systems, in order to verify the internal consistency of data (see [Bry93]). In machine learning, the term is not commonly used although various knowledge structures providing bias to the learning algorithm (cf. section 2.2.1), such as inference/task structures or qualitative models, can be pragmatically characterized as "constraints". Among the ML systems mentioned in this work, it is however only MOBAL which uses knowledge structures explicitely named as integrity constraints, in the form of disjunctive Horn clauses, and includes a mechanism for their checking. There are three syntactic types of constraints:

- constraints with both right-hand side (RHS) and left-hand side (LHS): if the LHS can be unified with some facts in the knowledge base then so must the RHS (which can have the form of disjuction). As an example, consider the constraint

  ```
  manager(X) --> education(X,university) OR (worked(X,Yrs) AND Yrs>=3)
  ```

  stating that a manager must have either higher education or a three-year work experience.

- constraints with right-hand side only ("unconditional requirements"). As an example, consider the constraint

  ```
  --> director(X)
  ```

  stating that there must be an information about the identity of the director (in a KB describing the organizational structure of a company).

- constraints with left-hand side only ("bans"). As an example, consider the constraint

  ```
  director(X) AND director(Y) AND X=Y -->
  ```

  stating that must not be two different persons occupying the post of director.

If an integrity constraint is violated, a message is inserted into the agenda; messages are objects which are manipulated by MOBAL in a way similar to knowledge items - they can be, for example, resubmitted to MOBAL so that it can suggest a way of handling. Integrity constraints in MOBAL can be, as it seems, used to check formal consistency of data as well as undesirable states in the world described by the knowledge base. In the experiments conducted in a mass transit application ([Vitas97]), both cases appeared; this application being a "toy" one, strong conclusions should however not be drawn from.

3 Background project - ESOD method of learning and classification

In this section, we show the main principles of the ESOD[43] method for learning a compositional rulebase from data. There are good reasons for including the description of ESOD in this thesis, although it does not, by itself, exploit prior knowledge in a way similar to techniques described in the main overview section (section 2). Namely, ESOD became a starting point for a large part of the original research presented here; results of this research are presented in sections 4.4 and 5 (the theoretical framework for algebraic knowledge integration, presented in section 4.3, is not immediately related to ESOD).

In section 3.1, the distinction between "logical" and compositional rulebases is outlined. The ESOD learning method itself is briefly described in section 3.2.2. The use of ESOD-generated rulebases for classification in described in section 3.2.1, and some later extensions to the method (beyond the work of the author of this thesis) are mentioned in section 3.3.

3.1 "Logical" vs. compositional rulebases

A possible way how to formalize a theory is to write it as a set of rules (and, possibly, facts) using the apparatus of *classical logic*. This idea is not too far from current trends in knowledge acquisition and modelling[44]. Moreover, most machine learning systems output knowledge in a logically describable form - in a propositional or restricted first-order language. The bulk of inference in "logical" rule bases is performed in a classical deductive manner. In the simplest case: if we know that the decision situation satisfies the condition of a rule, we can claim that its conclusion also holds. It is however known that in practice, experts often use *uncertain* - stochastic or heuristic rules. This is reflected in the conception of expert systems such as MYCIN [Shortl76] or PROSPECTOR [DudGas79], where individual rules are associated with *weights*. Inference is realized as composition of weights; therefore, we speak about *compositional* rule bases. *Combination functions* are typically tailored to fit the given interpretation - beliefs in MYCIN, subjective probabilities in PROSPECTOR etc.

The majority of empirical learning systems output a "logical" rulebase[45]. Even if, in some approaches, induced rules are assigned weights derived from their validity on data, these mostly serve for *selection* of a single rule from a candidate set ("select-and-fire" way of operation). An often mentioned exception is the hybrid (rule-based and

[43] Stands for "Expert System from Observational Data".

[44] There are actually continuous discussions in the Artificial Intelligence community whether the apparatus of classical logics (and its "orthodox" extensions) is a sufficient tool for representing and processing knowledge, cf. e.g. [Nilss91], [Birnb91]. In this work we, however, reduce the meaning of "logical" knowledge to a set of rules considered as "true", i.e. not endowed with certainty factors of any sort.

[45] There are of course many systems outputting other forms of "logical" knowledge, such as decision trees or lists. Naive Bayesian classifiers are also sometimes considered as empirical learning systems (cf. [BraCeK95]); these are capable of providing a numerical quantification of the classification decision.

probabilistic) system ITRule [SmyGoH90]. As another, we can view the ESOD method, which is capable of learning a compositional (although, single-layered) rulebase directly from data.

3.2 The ESOD classification and learning algorithms

3.2.1 Classification in ESOD

The task of *classification*, as it is understood in ESOD, consists in assigning an object o into a class c, based on values of nominal attributes; a *classification pair* (o, c) is thus established. Each attribute-value pair is called *category*, and a conjunction of them is called *combination*. A special kind of combination is the *empty combination* $\emptyset$, which contains no categories. A *description* of an object is a *full-length* combination, i.e. a conjunction of categories of all attributes. A *subcombination* of a combination $Comb$ is a combination which contains only such categories which exist also in $Comb$. An object o *satisfies* a combination $Comb$ if its description is a subcombination of $Comb$.

A *rule* r maps a combination $Comb$ onto class c; we will write it in the form $Comb \Rightarrow c$, where $Comb$ is denoted as *condition* of the rule. A rule with empty combination as condition is called *empty rule*. A *ruleset* R is a set of rules; its *classification domain* is the set of all classes which appear in rules from R. We assume that the classification domain is disjoint with the set of categories used in rules' conditions (in other words, rules in ESOD are not supposed to be chained). A *weight* of a rule is a real number from the interval $[0, 1]$.

Let us assume that a composition function $\oplus$ is given, which can be applied on combining the weights of rules leading to the same class. In most implementations of ESOD, composition operation

$$x \oplus y = \frac{x.y}{x.y + (1-x).(1-y)} \tag{1}$$

is used, which corresponds to the pseudo-bayesian rule combination function from the Prospector system, transformed from the interval $[-1, 1]$ to the interval $[0, 1]$. A *global weight function* $\mathcal{G}$ is then a function $\mathcal{G}(R, c, o)$, which returns the composed (using the basic composition function $\oplus$) weight of all rules from R with class c and a condition $Cond$ which is satisfied by the description of object o. Given a global weight function $\mathcal{G}$, a ruleset R with classification domain C, and an object o: a class $c_i \in C$ is the *best-weight class* of o wrt. R iff for every $c_j \in C$, $c_j \neq c_i$ holds $\mathcal{G}(R, c_j, o) < \mathcal{G}(R, c_i, o)$.

Performing classification with a given ruleset corresponds (at the simplest) to finding the best-weight class.[46] For practical purposes, implemented ESOD classifiers output the composed weight for each class in turn, in a sorted list and possibly including the conditions of those rules which have been applied - this provides a smooth quantification of classification decisions. The ESOD classifier with an attached ruleset actually corresponds to a *compositional expert system*, which ressembles (assuming the combination function

[46] If the best-weight class does not exist (there are more classes with the same weight), such simple classification fails; this is however rare in most real situations, and could easily be handled.

is (1)) the PROSPECTOR system by its inference engine; the ruleset is, however, typically not contrived in a dialog with expert (as are traditional knowledge bases) but learned from data, as we will see in the following subsection.

3.2.2 Learning in ESOD

The *learning* task in ESOD consists in finding a set of weighted rules which *describes* a source dataset as closely as possible, but can also be used to perform *classification* of previously unseen objects, using the compositional method described in the previous subsection. Before we describe the learning algorithm itself, some simple notions have again to be explained[47]:

- A *dataset* D is a set of objects described by values of a finite set of attributes A; each object also has a class from a finite set C. Every attribute has a finite, unordered domain of values. All values of the same attribute are mutually exclusive; similarly, all classes are mutually exclusive.

- The *coverage*[48] of a combination $Comb$ in a dataset D, $Cov_D(Comb)$, is the number of objects from D whose description satisfies $Comb$. The *coverage* of a rule r in a dataset D, $Cov_D(r)$, equals the coverage of its condition. The *correct-coverage* of a rule r in a dataset D, $CCov_D(r)$, is the number of objects which have the same class as r and whose description satisfies the condition of r.

- The *validity* of a rule r in a dataset D, $Val_D(r)$, is the ratio of the correct-coverage and coverage:
$$Val_D(r) = \frac{CCov_D(r)}{Cov_D(r)}$$
The validity equals to the (empirical) conditional probability $P(c/Comb)$ in data D.

- Rule p is *more general* than rule q iff they have the same class, and the condition of p is a subcombination of the condition of q.

For clarity, we will add a trivial example. Consider the dataset which is written as a matrix in Fig. 17 (objects corresponding to rows and attributes to columns), and a rule

$$r: \quad (a_1 = a \ \wedge \ a_3 = p) \Rightarrow x$$

The rule has:

[47] Note that the scope of the terminology involved is limited to the description of the ESOD system and its various modifications and extensions. The terms are not always consistent with the same terms used throughout the overview part of the thesis (section 2) or elsewhere.

[48] In some works on ESOD, the term "coverage" is used for the *relative* coverage, i.e. the proportion of covered objects in the whole dataset.

a_1	a_2	a_3	$Class$
a	i	p	x
b	j	p	y
a	i	p	x
a	j	q	y
a	j	p	y
b	k	p	x
a	k	p	x
b	i	q	x

Figure 17: Example dataset D

- coverage $Cov_D(r) = 4$,
- correct-coverage $CCov_D(r) = 3$,
- validity $Val_D(r) = 0.75$.

We could find three more rules[49] more general than r, one of them being an empty rule:

$$(a_1 = a) \Rightarrow x \qquad (a_3 = p) \Rightarrow x \qquad \emptyset \Rightarrow x$$

The way of constructing a rulebase in ESOD can be viewed as analogical to building an axiomatic theory. Rules-axioms are inserted only if they cannot be inferred from other axioms, i.e. if they "bring new information". The resulting rulebase is thus, in a certain sense, minimal [Ivanek94].

The algorithm can be written, in a simple form, as in Fig. 18. As statistical test T, the standard χ^2 goodness-of-fit test has been mainly used, and the composition operation is (1) (the one used also by the classification mechanism).

The algorithm has a relatively high computational complexity, due to the systematic search method. The complexity increases exponentially with the number of attribute values. This leads to the method being applicable to medium-size data: for too small datasets (up to a few tens of objects), the statistical bias may hinder learning any but empty rules, for too big datasets (especially in connection with high numbers of attribute values) the execution time becomes prohibitive.

It should be pointed out that, unlike most ML algorithms, an ESOD rulebase usually contains rules which are algebraically subsumed by each other. The inference mechanism complies with the *correction principle* suggested by P. Hájek and thus overcomes the common drawback of PROSPECTOR-like techniques - the assumption of conditional independence. Nevertheless, the combination scheme remains fixed and need not always

[49] We will write rules with or without identifiers, as more appropriate.

Input: Data D, goal combination C^*, composition operation $\oplus$, statistical test T, coverage threshold t.

Output: KB = set of rules with weight $\in$ [0,1];

Computation:
Let KB contain only empty implication $\emptyset \Rightarrow C^*$ with relative frequency of C^* in data D as weight.
Let CAT be a list of categories (nominal attribute-value pairs) jc sorted in the descending order of coverage; only the categories with coverage at least t are included.
Let OPEN be a list of implications $jc \Rightarrow C^*$ for each jc from CAT, sorted in the descending order of coverage.
While OPEN is not empty repeat

- delete the first implication $Ant \Rightarrow C^*$ from OPEN;
- compute the *composed weight* w_c from the weights of all *subrules* of $Ant \Rightarrow C^*$ which are already in KB (using operation $\oplus$);
- if the validity of $Ant \Rightarrow C^*$ significantly (by test T) differs from w_c then add $Ant \Rightarrow C^*$ to KB with weight w such that w composed with w_c equals to validity;
- for each jc which precedes in CAT every category from Ant and attribute j is not included in Ant, compute the coverage of combination $jc \,\&\, Ant$; if the coverage is at least t then insert $jc \,\&\, Ant$ into OPEN according to the coverage.

Figure 18: The learning algorithm of ESOD

conform to the actual dependency of premises. Some observations and comments to this problem can be found in section 6.

Let us attempt to characterize the original ESOD learner in terms of input and output. In the informal description of the algorithm, five input items appear: data, combination function, composition operation, statistical test and coverage threshold. Among them, data is the mandatory input of any inductive learner. The remaining four can be characterized as low-level algorithic parameters rather than knowledge. In this respect, ESOD is a "knowledge-weak" ML method; knowledge appears only on the output, in the form of weighted classification rules, i.e. problem-solving knowledge.

The types of input knowledge (set aside the human oracle) we have distinguished in the overview section (section 2) are problem-solving knowledge, static domain knowledge and metaknowledge. The first two are not considered in original ESOD (their use is actually

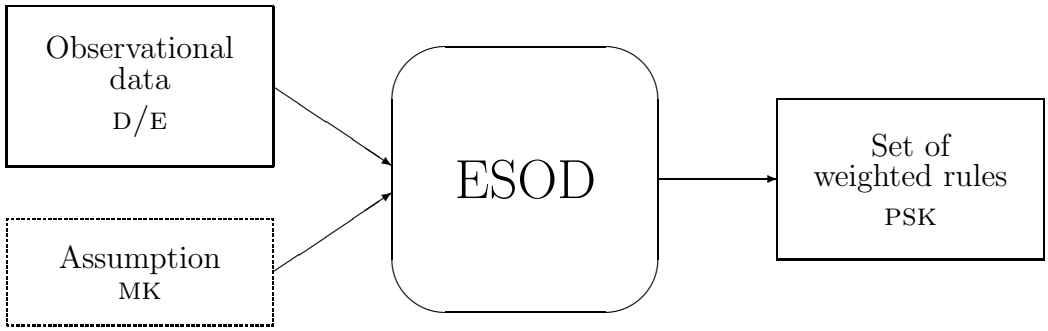

Figure 19: I/O diagram of ESOD.

the main topic of the present thesis, in sections 4.4 and 5). As a particular type of metaknowledge, somewhat similar to metaknowledge used e.g. in MOBAL (section 2.2.2), can be viewed the central *assumption* of ESOD: *"New knowledge is what cannot be inferred from existing knowledge."* This "qualitative metarule" is however embedded into the algorithm instead of being kept separately in some meta-knowledge base. In the I/O diagram of ESOD in Fig. 19, we present it as metaknowledge, but without claiming that it actually is such.

3.3 Implementations of ESOD

Original versions of the ESOD algorithms have been implemented by Ivánek and Stejskal on HP9845 computer (in 1986) and on PCs (in 1988). The original learning algorithm differed from the one presented above in separation of the two phases - generation of combinations, and testing of rules. Most later improvements to ESOD are due to P. Berka, and are connected with the development of a new knowledge engineering toolbox called KEX - Knowledge Explorer. The theoretical outcomes of KEX are the *combinational data analysis* and the above described ESOD method of knowledge acquisition. We will not describe KEX in detail (comprehensive information in Czech can be found in [Berka95]; a concise description in English is in [BerIva94]) but only point out some new features of the up-to-date ESOD implementation:

- Rules for *multiple classes* are learned in one turn.
- *Unknown* attribute values are handled.
- A pre-processor for class-sensitive *discretization* of numerical attributes has been added [Berka93c].

The whole KEX system runs on PCs and (in a restricted version) on Sun and Apollo workstations.

For experimental purposes in the framework of the present thesis, a restricted version of ESOD has been implemented in the Prolog language, with extensions described in section 5. æ

4 Learning with problem-solving knowledge using knowledge integration - methodology

4.1 Objectives

In section 2.1 we have described some existing techniques of learning with prior problem-solving knowledge. Of them, only knowledge revision (section 2.1.2) applies genuinely to the situation when two sources of information - data and prior knowledge - are available. Incremental learning (section 2.1.1) largely overlaps with knowledge revision in algorithmic principles but applies to the situation when no *prior* knowledge is available at the start. On the other hand, knowledge integration (section 2.1.3) does not use data as direct input, and cannot be characterized as *learning* method, in the narrow sense.

Our research project carried out in 1993-1994 was concentrated on the following problem:

Problem 1 *Could learning with prior problem-solving knowledge be realized by means of* knowledge integration *instead of knowledge revision? How can this approach be formalized for* rules without weight *("logical" rules) and* weighted rules*?*

The first subproblem led to the formulation of an abstract "bypass" model, which replaces knowledge revision with empirical induction and knowledge integration; some informal assumptions concerning the applicability of this model have been formulated (section 4.2). The integration itself was formalized for both types of rules (with and without weight). Rules without weight can be integrated based on the notions of conditional truth and consequence values. For two algebraically identical rules with different weights (one given by expert and one learned from data), integration can be performed by means of weight interpolation.

The research on integration of *isolated* weighted rules naturally led to the question whether the integration method could be extended to *compositional* rulebases, in particular those learned by the ESOD method (the background project, see section 3):

Problem 2 *Could the integration of weighted rules be used as a method for learning a compositional rulebase* from expert's rules and data*, as an extension to the existing ESOD learning method?*

Preliminary analysis has revealed the problem as extremely difficult; although an experimental implementation of the approach has been realized, the direction has not been followed any further as a theoretically sound solution did not seem to be within the reach of the author's project. Possible modifications of the problem which could be easier to attack have been formulated but not yet investigated in detail.

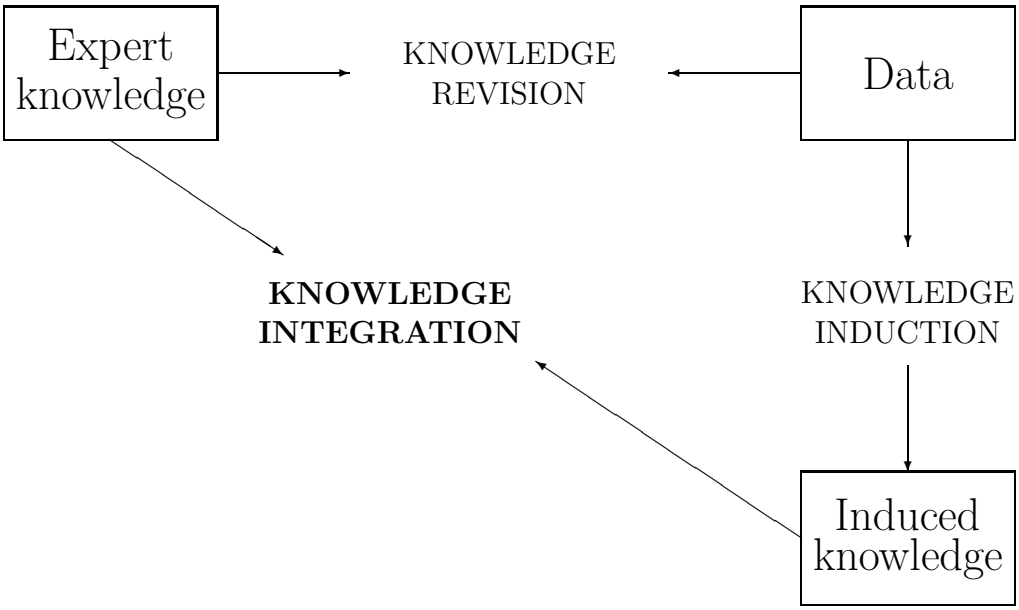

Figure 20: Induction and integration as a "bypass" of revision

4.2 The "bypass" model of learning from expert and data

In distributed AI (cf. section 2.1.3), the motivation for integration is to put together sets of knowledge *learned* by independent agents. In our opinion, this task is a specific case of the more general task of *integrating information from multiple, independent, information sources*. In section 2.1.2, we have briefly surveyed the state-of-the-art of another area with similar characteristics - *knowledge revision*. Knowledge revision can be viewed as integration of information from *two* resources - a dataset (or example set), and a body of knowledge. If an "alternative" knowledge base were induced from the dataset, we would instead have two knowledge bases ready for *integration*. In this way, the revision process could be "bypassed" (see Fig. 20).

The bypass has, obviously, some arguable points. The starting point of knowledge revision is the evaluation process, in which the expert rules are confirmed or rejected (or none) by relevant data objects. In our "bypass" model, there is no guarantee that the bias of the inductive algorithm *allows to learn* appropriate "confirming" or "rejecting" rules; the mutual effect of both resources thus may vanish. Another problem is the need for inductive learners which output knowledge in the same *form* (not only syntactically but also semantically) as experts.

The potential benefits of the "bypass" model arise from the *balanced* roles of both information resources. In knowledge revision, in contrast, expert knowledge represents the starting point of search for plausible hypotheses. The assumption usually is that

expert knowledge requires "fine tuning rather than major overhaul" [CraSle90] using the data. If, however, some important relations are left intact by the expert, it may be difficult to arrive at them via revision. Another caveat of revision is the noise in data, which may lead, in the case of *data-driven* (incremental) revision techniques, to frequent undoing of premature changes and thus to inefficient looping; *generate-and-test* revision techniques (which can better cope with noise) may show computationally expensive if modified knowledge is to be tested on a large dataset in every revision step. In the "bypass" model, on the other hand, we assume that the induction can be performed with one of a host of noise-robust learning algorithms, and only once.

We have formed a hypothesis that the "bypass" model can be particularly useful under the following conditions:

- Neither expert nor data can be fully trusted; the expert is unreliable and the data are noisy observational data rather than purposefully selected examples.
- Both sources are relatively independent. This e.g. means that the expert has not built his/her expertise upon analysis of data and, on the other hand, data are not merely a protocol of the expert's past decisions.

In this context, we have a credit-assignment problem: it is desirable to yield the best of both resources, while not fully trusting either of them. An integration method which treats both resources as equal would then be suitable. In our work, we have concentrated only on the *integration* task, in adopting the (probably too simplifying) assumption that the *induction* could be performed by almost any of existing symbolic learning algorithms.

Our problem situation is different from the previously described situation in distributed learning. We do not have any external dataset to evaluate the rules *empirically*. The single dataset has already been used to induce the empirical rules; if the rule quality were measured wrt. the same dataset, empirical rules would be inappropriately favourized compared to expert rules. We can therefore rely merely on algebraic properties of the rulesets themselves, which can be verified *syntactically*. In the following subsection, we describe a simple method, which is currently restricted to integration of rules without weights nor chaining; the method has been presented at the MLnet Familiarization Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases, in Heraclion, April 1995 [Svatek95].

4.3 Algebraic approach to integration of rules without weight

4.3.1 Basic notions

Let $\mathcal{D}$ be a set of decision situations in a given problem domain. Let $\mathcal{K}$ - the space of *conditions* - be a set of expressions; each expression can be evaluated as either true or false for a given decision situation; let $D(e)$ denote the set of decision situations from $\mathcal{D}$ for which expression e evaluates as true. Let a *generality relation* $\geq_g$ be defined over $\mathcal{K}$,

such that if $k_1, k_2 \in \mathcal{K}, k_1 \geq_g k_2$ (k_1 is at least as general as k_2[50]) then $D(k_2) \subseteq D(k_1)$. Obviously, such generality relation inherits the *transitivity* property from the set-inclusion relation.

Let $\mathcal{L}$ (the space of *conclusions*) be a set of *atomic* expressions; the expressions are mutually exclusive, i.e. if two or more of them are true for a given decision situation, a contradiction arises. However, the conclusions cannot be evaluated as such, for a given decision situation; this implicitely means that the expressions from $\mathcal{K}$ cannot contain expressions from $\mathcal{L}$, and that decision rules leading from conditions to conclusions cannot be chained. We will consider a universe $\mathcal{U}$ of all *decision rules* in the form $r : k \rightarrow l$, where r is a unique rule identifier, k is an expression from $\mathcal{K}$ and l is an expression from $\mathcal{L}$. For clarity, we will denote the condition of rule r as $L(r)$ and its conclusion as $L(r)$.

First of all, we are interested in the correctness of rules, which can be most simply understood in the following way. Each rule r has exactly one of two *truth values*, in a way analogical to logical implication:

- $\|r\|^T = \texttt{t}$ (true) iff $D(k) \subseteq D(l)$, i.e. in all situations where r could be applied, its application would lead to a correct conclusion;
- $\|r\|^T = \texttt{f}$ (false) otherwise.

Especially when considering large numbers of potential rules, we are however interested not only in their correctness but also in their *importance* (or, significance). Here, we use the notion of importance in terms of *domain of application*. Let n be a fixed importance threshold (in the following, we will consider it as constant for the whole problem domain); every rule r from $\mathcal{U}$ then has exactly one of two *importance values*:

- $\|r\|^I = \texttt{i}$ (important) iff $card(D(L(r))) \geq n$
- $\|r\|^I = \texttt{u}$ (unimportant) otherwise.

To read, a rule r is considered important if the number of decision situations in which it is applicable is at least n.

On the universe $\mathcal{U}$ of rules, we can define the *truth-consequence* operation Cn^T:

$$Cn^T(r) = \{r' \mid R(r) = R(r') \wedge L(r) \geq_g L(r')\}$$

i.e. as the set of all rules with the same conclusion and less (or equally) general condition than r. The operation can be interpreted as: if a rule r is true (i.e. correct) then all rules from $Cn^T(r)$ also true:

[50]Note that two syntactically different (e.g. first-order) expressions can be equally general. Determining whether a pair of expressions belongs to the generality relation may be untrivial in first-order languages. In this thesis, we are, however, interested in expressions having the form of combinations (conjunctions) of categories, on which the generality relation can be determined more-or-less straightforwardly.

$$\forall r, r', r' \in Cn^T(r) \;\; \|r\|^T = \mathtt{t} \;\Rightarrow\; \|r'\|^T = \mathtt{t}$$

The truth-consequence of a set of rules will be simply the union of their truth-consequences:

$$Cn^T(\mathcal{R}) = \bigcup_{r_i \in \mathcal{R}} Cn^T(r_i)$$

This definition of consequence conforms to the notion of (Tarskian) *logical consequence*, as Cn^T is reflexive, monotonous and transitive: for any sets of rules X and Y holds

1. $X \subseteq Cn^T(X)$;
2. if $X \subseteq Y$ then $Cn^T(X) \subseteq Cn^T(Y)$;
3. $Cn^T(Cn^T(X)) \subseteq Cn^T(X)$.

The proofs are omitted for the sake of brevity; the reflexivity and transitivity immediately follow from the monotonicity, and from the reflexivity and transitivity of both $\geq_g$ and $=$ relations.

Similarly, we can define the *falsity-consequence* operation Cn^F:

$$Cn^F(r) = \{r' \mid R(r) \neq R(r') \;\wedge\; L(r) \geq_g L(r')\}$$

i.e. as the set of all rules with different conclusion and less (or equally) general condition than r. The operation can be interpreted as: if a rule r is true (i.e. correct) then all rules from $Cn^F(r)$ are false:

$$\forall r, r', r' \in Cn^F(r) \;\; \|r\|^T = \mathtt{t} \;\Rightarrow\; \|r'\|^T = \mathtt{f}$$

The falsity-consequence of a set of rules is the union of their truth-consequences:

$$Cn^F(\mathcal{R}) = \bigcup_{r_i \in \mathcal{R}} Cn^F(r_i)$$

Unlike truth-consequence, falsity consequence cannot be understood as logical consequence in terms of classical logic, as it does not satisfy the transitivity property.

The present method assumes that we cannot evaluate the rules, even not empirically (on data), their unconditional truth values are thus impossible to determine. Instead, as we have suggested in the foreword, we will concentrate on interrelations between rules and sets of rules. For this purpose, we will need the notion of *conditional* truth values.

Given a rule $r \in \mathcal{U}$ and a ruleset $\mathcal{R} \subset \mathcal{U}$, we can assign to r one of four *conditional*[51] *truth values* wrt. $\mathcal{R}$:

[51] As we will speak only about conditional truth values since now, we will mostly omit the attribute "conditional".

- $\|r\|^T_{\mathcal{R}} = \mathtt{t}$ (true) iff $r \in Cn^T(\mathcal{R}) \wedge r \notin Cn^F(\mathcal{R})$;
- $\|r\|^T_{\mathcal{R}} = \mathtt{c}$ (contradiction) iff $r \in Cn^T(\mathcal{R}) \wedge r \in Cn^F(\mathcal{R})$;
- $\|r\|^T_{\mathcal{R}} = \mathtt{u}$ (unknown) iff $r \notin Cn^T(\mathcal{R}) \wedge r \notin Cn^F(\mathcal{R})$;
- $\|r\|^T_{\mathcal{R}} = \mathtt{f}$ (false) iff $r \notin Cn^T(\mathcal{R}) \wedge r \in Cn^F(\mathcal{R})$.

The values actually indicate whether the validity of rule r is "declared" or "denied" (or both, or none) by the rules from the ruleset $\mathcal{R}$. We can arrange them into the lattice at Fig. 21, with *true* on the top and *false* in the bottom.

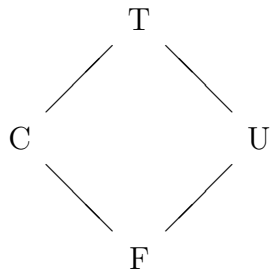

Figure 21: Lattice of conditional truth values

Analogically to previously introduced truth- (and falsity-) consequence, we can define the *importance-consequence* operation Cn^I, which will however concern only conditions of rules:

$$Cn^I(r) = \{r' \mid L(r') \geq_g L(r)\}$$

$$Cn^I(\mathcal{R}) = \bigcup_{r_i \in \mathcal{R}} Cn^I(r_i)$$

It can be understood as: if a rule is important then all rules with at least as large domain of application is also important. Importance-consequence is reflexive, monotonous and transitive, it can be thus viewed as a form of logical consequence.

Given a rule $r \in \mathcal{U}$ and a ruleset $\mathcal{R} \subset \mathcal{U}$, we can assign to r one of two conditional[52] *importance values* wrt. $\mathcal{R}$. The values will indicate whether the rule r is "declared" important by the ruleset $\mathcal{R}$. Note that conditional unimportance, unlike falsity, is defined via closed-world assumption[53]:

- $\|r\|^I_{\mathcal{R}} = \mathtt{i}$ (important) iff $r \in Cn^I(\mathcal{R})$;

[52] Again, as we will speak only about conditional importance values since now, we will mostly omit the attribute "conditional".

[53] This can be interpreted as: "All important decision situations are assumed covered."

- $\|r\|^I_{\mathcal{R}} = \mathtt{u}$ (unimportant) otherwise.

Our method deals with a particular class of rulesets, namely with rulesets which are already, in some sense, minimal and consistent. For this purpose, we will define the formal notions of *minimality* and *consistency* of rulesets.

Definition 1 *A ruleset $\mathcal{R}$ is minimal iff it does not contain a pair of rules p, q such that $p \in Cn^T(q)$.*

Definition 2 *A ruleset $\mathcal{R}$ is consistent iff for every rule r from $Cn^T(\mathcal{R}) \cap Cn^I(\mathcal{R})$ holds $\|r\|^T_{\mathcal{R}} = \mathtt{t}$.*

A ruleset which is not minimal will be called *redundant*; a ruleset which is not consistent will be called *inconsistent*.

Note that while minimality is related only to a given ruleset $\mathcal{R}$ itself (it should not contain redundant rules), consistency must be verified for the set of all rules which are both truth- and importance-consequences of $\mathcal{R}$. We will show later that under certain conditions, the consistency property shrinks to "internal consistency", which can be verified on $\mathcal{R}$ only.

4.3.2 Construction of the integrated set

Finally, we will proceed with the integration task itself. Let $\mathcal{P}$ and $\mathcal{Q}$ be two *minimal and consistent* rulesets to be integrated, we will refer to them as to *source* rulesets. We want to obtain a new ruleset, $\mathcal{I}(\mathcal{P}, \mathcal{Q})$, which will be also minimal and consistent, and will capture (heuristically) as much information from both original sets as possible.

From the whole universe $\mathcal{U}$ of possible rules (of which only a subset belongs to either of source rulesets), we will immediately reject all rules not belonging to the truth-consequence of either $\mathcal{P}$ or $\mathcal{Q}$, and those not belonging to the importance-consequence of either $\mathcal{P}$ or $\mathcal{Q}$. Hence,

$$\mathcal{I}(\mathcal{P}, \mathcal{Q}) \subseteq (Cn^T(\mathcal{P}) \cup Cn^T(\mathcal{Q})) \cap (Cn^I(\mathcal{P}) \cup Cn^I(\mathcal{Q})) \tag{2}$$

The remaining rules (i.e. those which are "somehow" considered as true as well as important) can be assigned to six disjoint subsets, according to their truth-values wrt. the original sets - TT, TC, TU, TF, CU, CF:

- TT will contain the rules that are true wrt. both sets;
- TC will contain the rules that are true wrt. one (any of the two) set and contradictory wrt. the other;
- TU will contain the rules that are true wrt. one (any of the two) set and unknown wrt. the other;

- TF will contain the rules that are true wrt. one (any of the two) set and false wrt. the other;
- CU will contain the rules that are contradictory wrt. one (any of the two) set and unknown wrt. the other;
- CF will contain the rules that are contradictory wrt. one (any of the two) set and false wrt. the other.

With respect to the partial ordering of individual conditional truth values (lattice at Fig. 21), the new subsets will again form a lattice (Fig. 22).

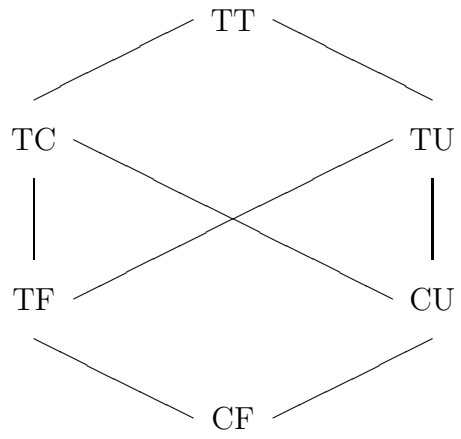

Figure 22: Lattice of subsets wrt. truth values

It can be proven (the proof is in the next subsection) that:

Theorem 0 *If both source rulesets are minimal and consistent then $TT \cup TC$ is a minimal and consistent ruleset.*

The situation is depicted at Fig. 23, with the subsets belonging to the minimal and consistent ruleset (i.e. candidate for integrated set) written in bold typeset.

Such integration would be rather restrictive; nevertheless, if we add any of other subsets to the union, the result is no longer guaranteed to be minimal and consistent. We will thus proceed by determining particular subsets of TU and TF; we will denote these subsets by TU_1 and TF_1 (and their complements as TU_2 and TF_2):

- TU_1 will contain the rules that are true and important in one set, and unknown and unimportant in the other;

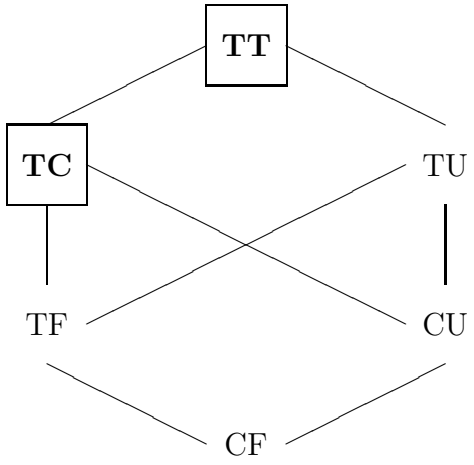

Figure 23: First candidate for integrated ruleset

- TF_1 will contain the rules that are true and important in one set, and false and unimportant in the other;

It can be proven (the proof is in the next subsection) that:

Theorem 1 *If both source rulesets are minimal and consistent then* $TT \cup TC \cup TU_1 \cup TF_1$ *is a minimal and consistent ruleset.*

The situation is depicted at Fig. 24, with the subsets belonging to the minimal and consistent ruleset (i.e. candidate for integrated set) written in bold typeset.

Therefore, let

$$\mathcal{I}(\mathcal{P}, \mathcal{Q}) = TT \cup TC \cup TU_1 \cup TF_1$$

The ruleset $\mathcal{I}(\mathcal{P}, \mathcal{Q})$ can be immediately considered as a simple, deductive "knowledge base". It will contain only rules from $\mathcal{P} \cup \mathcal{Q}$ (we omit the proof for the sake of brevity); its extent is thus "reasonably" restricted. The remaining subsets - CU, CF, TU_2 and TF_2 - may be redundant and/or inconsistent. They, however, contain "interesting" rules (some of them being new wrt. $\mathcal{P} \cup \mathcal{Q}$), and may be subjected to further evaluation, i.e. by an independent expert. This reevaluation could be performed successively (from "better" rules to "worse" rules), with respect to the existing partial ordering of subsets from Fig. 22. Practical aspects of human evaluation of rulesets would deserve a further study.

We will close this subsection with a simple example. Let the set of decision situations be the set of conjunctive expressions on the alphabet

$$A = \{a, b, c, d, e, f, g, h, i, j, k\}$$

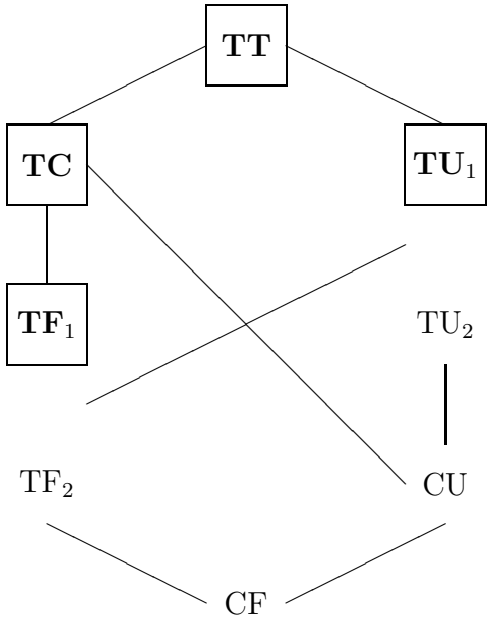

Figure 24: Accepted candidate for integrated ruleset

and the set of classes be

$$C = \{x, y, z\}$$

(with mutually exclusive classes). Let the source rulesets be

$$P = \{a \ \wedge b \rightarrow x, c \rightarrow y, f \rightarrow z, g \rightarrow z, h \wedge i \ \wedge j \rightarrow y, k \ \rightarrow x\}$$

$$Q = \{a \ \wedge b \rightarrow x, a \ \wedge c \wedge e \rightarrow y, a \ \wedge c \wedge f \rightarrow y, h \rightarrow x, k \ \rightarrow z \ \}$$

The subsets of rules wrt. conditional truth values, corresponding to the lattice from Fig. 22, are in Table 1. Rules which have not been present in P or Q are written in bold typeset.

For example, rule $a \wedge b \rightarrow x$ belongs to TT because it is present (and thus true) in both source rulesets; rule $a \wedge c \wedge f \rightarrow y$ belongs to TC because it is present (and thus true) in Q, and contradictory wrt. P (being "supported" by rule $c \rightarrow y$ and "rejected" by rule $f \rightarrow z$). The first candidate for integrated ruleset (according to theorem 0) will then be

$$\mathcal{I}_0(P, Q) = \{a \wedge b \rightarrow x, a \ \wedge c \wedge e \rightarrow y, a \ \wedge c \wedge f \rightarrow y\}$$

We will further decompose the subsets TU and TF in order to include more rules into the integrated set, see Table 2.

Class	Rules
TT	$a \wedge b \rightarrow x$, $a \wedge c \wedge e \rightarrow y$
TC	$a \wedge c \wedge f \rightarrow y$
TU	$g \rightarrow z$, $c \rightarrow y$, $k \rightarrow x$, $k \rightarrow z$, $\mathbf{a} \wedge \mathbf{c} \rightarrow \mathbf{y}$, $\mathbf{c} \wedge \mathbf{e} \rightarrow \mathbf{y}$
TF	$h \wedge i \wedge j \rightarrow y$, $h \rightarrow x$, $f \rightarrow z$, $\mathbf{h} \wedge \mathbf{i} \rightarrow \mathbf{x}$, $\mathbf{h} \wedge \mathbf{j} \rightarrow \mathbf{x}$, $\mathbf{h} \wedge \mathbf{i} \wedge \mathbf{j} \rightarrow \mathbf{x}$
CU	$\mathbf{c} \wedge \mathbf{f} \rightarrow \mathbf{y}$
CF	$\mathbf{c} \wedge \mathbf{f} \rightarrow \mathbf{z}$, $\mathbf{a} \wedge \mathbf{c} \wedge \mathbf{f} \rightarrow \mathbf{z}$

Table 1: Table of assignment of rules to subsets in the example

Class	Rules
TU_1	$g \rightarrow z$
TU_2	$c \rightarrow y$, $k \rightarrow x$, $k \rightarrow z$, $\mathbf{a} \wedge \mathbf{c} \rightarrow \mathbf{y}$, $\mathbf{c} \wedge \mathbf{e} \rightarrow \mathbf{y}$
TF_1	$h \wedge i \wedge j \rightarrow y$
TF_2	$h \rightarrow x$, $f \rightarrow z$, $\mathbf{h} \wedge \mathbf{i} \rightarrow \mathbf{x}$, $\mathbf{h} \wedge \mathbf{j} \rightarrow \mathbf{x}$, $\mathbf{h} \wedge \mathbf{i} \wedge \mathbf{j} \rightarrow \mathbf{x}$

Table 2: Further decomposition of subsets TU and TF in the example

For example, rule $g \rightarrow z$ belongs to TU_1 because it is present (and thus true as well as important) in P and unknown and unimportant in Q (as there is no rule with literal g in the condition). Rule $c \rightarrow y$, on the other hand, belongs to TU_2; it is present (and thus true as well as important) in P, but *important* also wrt. Q due to more specific rules $a \wedge c \wedge e \rightarrow y$ and $a \wedge c \wedge f \rightarrow y$. The larger integrated set (according to Theorem 1) is then

$$\mathcal{I}(P,Q) = \{a \wedge b \rightarrow x, a \wedge c \wedge e \rightarrow y, a \wedge c \wedge f \rightarrow y, g \rightarrow z, h \wedge i \wedge j \rightarrow y\}$$

It contains one rule shared by P and Q, two rules specific for P, and two rules specific for Q, i.e. only rules from $P \cup Q$ as stated earlier.

4.3.3 Proofs of minimality and consistency

We want to prove theorem 1 stating that $TT \cup TC \cup TU_1 \cup TF_1$ is a consistent and minimal ruleset. First, we will prove the following two lemmas:

Lemma 1 *If a ruleset $\mathcal{R}$ is minimal and for every rule $r \in \mathcal{R}$ holds $\|r\|^T_{\mathcal{R}} = \mathtt{t}$ then $\mathcal{R}$ is consistent.*

Lemma 2 *If a ruleset $\mathcal{R}$ is minimal and consistent then $Cn^T(\mathcal{R}) \cap Cn^I(\mathcal{R}) = \mathcal{R}$.*

Proof of lemma 1 We will carry out the proof by contradiction. Suppose that the premises hold but $\mathcal{R}$ is inconsistent, i.e. there is, by definition of consistency, a rule $r \in (Cn^T(\mathcal{R}) \cap Cn^I(\mathcal{R}))$ such that $\|r\|^T_{\mathcal{R}} \neq \mathtt{t}$. From $r \in Cn^T(\mathcal{R})$ follows that there is a rule $r' \in \mathcal{R}$ such that $L(r') \geq_g L(r)$. Similarly, from $r \in Cn^I(\mathcal{R})$ follows that there is a rule $r'' \in \mathcal{R}$ such that $L(r) \geq_g L(r'')$. Due to the transitivity of $\geq_g$, $L(r') \geq_g L(r'')$. Then, we will analyse the following possibilities:

1. If $R(r') = R(r'') \wedge L(r') = L(r'')$ then $r' = r'' = r$, and r thus belongs to $\mathcal{R}$; then it is not true that for every rule $r \in \mathcal{R}$ holds $\|r\|^T_{\mathcal{R}} = \mathtt{t}$ $\square$
2. If $R(r') = R(r'') \wedge L(r') \neq L(r'')$ then $r'' \in Cn^T(r')$ and $\mathcal{R}$ is thus not minimal $\square$
3. If $R(r') \neq R(r'')$ then $\|r''\|^T_{\mathcal{R}} \neq \mathtt{t}$; then it is not true that for every rule $r \in \mathcal{R}$ holds $\|r\|^T_{\mathcal{R}} = \mathtt{t}$ $\square$

Proof of lemma 2 We will carry out the proof by contradiction; it is nearly identical to the previous one. Suppose there is a rule r such that $r \in (Cn^T(\mathcal{R}) \cap Cn^I(\mathcal{R})), r \notin \mathcal{R}$. From $r \in Cn^T(\mathcal{R})$ follows that there is a rule $r' \in \mathcal{R}$ such that $L(r') \geq_g L(r)$. Similarly, from $r \in Cn^I(\mathcal{R})$ follows that there is a rule $r'' \in \mathcal{R}$ such that $L(r) \geq_g L(r'')$. Due to the transitivity of $\geq_g$, $L(r') \geq_g L(r'')$. Then, we will analyse the following possibilities:

1. If $R(r') = R(r'') \wedge L(r') = L(r'')$ then $r' = r'' = r$, and r thus belongs to $\mathcal{R}$ $\square$
2. If $R(r') = R(r'') \wedge L(r') \neq L(r'')$ then $r'' \in Cn^T(r')$ and $\mathcal{R}$ is thus not minimal $\square$
3. If $R(r') \neq R(r'')$ then $\|r''\|^T_{\mathcal{R}} \neq \mathtt{t}$ since $r'' \in Cn^F(r')$; $\mathcal{R}$ is thus not consistent $\square$

We will proceed by proving the (auxilliary) theorem 0, stating that $TT \cup TC$ is a consistent and minimal ruleset.

Proof of theorem 0

Proof of minimality of $TT \cup TC$ We will prove the property by contradiction. Suppose there is a pair of rules, $p, q \in (TT \cup TC)$, such that $p \in Cn^T(q)$. By (2) and by the specific properties of TT and TC, we know that:

1. $p \in Cn^T(\mathcal{P}) \wedge p \in Cn^T(\mathcal{Q})$
2. $q \in Cn^T(\mathcal{P}) \wedge q \in Cn^T(\mathcal{Q})$
3. $p \in Cn^I(\mathcal{P}) \vee p \in Cn^I(\mathcal{Q})$
4. $q \in Cn^I(\mathcal{P}) \vee q \in Cn^I(\mathcal{Q})$

where $\mathcal{P}, \mathcal{Q}$ are the original sets. Then exists a set $\mathcal{R} \in \{\mathcal{P}, \mathcal{Q}\}$ such that $p \in (Cn^T(\mathcal{R}) \cap Cn^I(\mathcal{R})$. As $\mathcal{R}$ is minimal and consistent, from lemma 2 follows that $p \in \mathcal{R}$, and thus $p \in \mathcal{P} \vee p \in \mathcal{Q}$. Similarly, $q \in \mathcal{P} \vee q \in \mathcal{Q}$. For each of $\mathcal{P}, \mathcal{Q}$ to be minimal, p and q cannot both belong to $\mathcal{P}$ or to $\mathcal{Q}$. Assume that $p \in \mathcal{P}$ and $q \in \mathcal{Q}$ (the other case would be treated analogically). From the above list of properties follows that $q \in Cn^T(\mathcal{P})$, hence there is a rule $q' \in \mathcal{P}$ such that $q \in Cn^T(q')$. By transitivity of Cn^T, $p \in Cn^T(q')$; since both p and q' are from $\mathcal{P}$, $\mathcal{P}$ is not minimal □

Proof of consistency of $TT \cup TC$ From the previous proof, we know that $TT \cup TC$ is minimal. By lemma 1, it thus suffices to prove that for every rule $r \in (TT \cup TC)$, $\|r\|^T_{TT \cup TC} = \mathtt{t}$.

We will carry out the proof by contradiction. Suppose there is a rule $r \in (TT \cup TC)$ such that $\|r\|^T_{TT \cup TC} \neq \mathtt{t}$. Since, obviously, $r \in Cn^T(TT \cup TC)$, the only truth value wrt. $(TT \cup TC)$ which remains possible for r is $\mathtt{c}$ (contradiction). From $\|r\|^T_{TT \cup TC} = \mathtt{c}$ follows that there is a rule $r' \in (TT \cup TC)$ such that $r \in Cn^F(r')$, i.e. $L(r') \geq_g L(r)$.

From (2) we know that there is at least one source ruleset $\mathcal{R}$ such that $r \in Cn^I(\mathcal{R})$; hence there is a rule $r'' \in \mathcal{R}$ such that $L(r) \geq_g L(r'')$. From $r' \in (TT \cup TC)$ follows that r' belongs to the truth-consequence of each of the source sets. Hence, there is a rule $r''' \in \mathcal{R}$ such that $L(r''') \geq_g L(r')$. By transitivity of $\geq_g$, $L(r''') \geq_g L(r'')$. As both r''' and r'' belong to $\mathcal{R}$, if $R(r''') = R(r'')$ then $\mathcal{R}$ is not minimal, otherwise $\mathcal{R}$ is not consistent □

Finally, we can make out the proof of the main theorem, 1.

Proof of theorem 1

Proof of minimality of $TT \cup TC \cup TU_1 \cup TF_1$ We will, again, prove the property by contradiction. Suppose there is a pair of rules, $p, q \in (TT \cup TC \cup TU_1 \cup TF_1)$, such that $p \in Cn^T(q)$. By (2) and by the specific properties of $(TT \cup TC \cup TU_1 \cup TF_1)$, we know that:

1. $(p \in Cn^T(\mathcal{P}) \wedge p \in Cn^I(\mathcal{P})) \vee (p \in Cn^T(\mathcal{Q}) \wedge p \in Cn^I(\mathcal{Q}))$
2. $(q \in Cn^T(\mathcal{P}) \wedge q \in Cn^I(\mathcal{P})) \vee (q \in Cn^T(\mathcal{Q}) \wedge q \in Cn^I(\mathcal{Q}))$

where $\mathcal{P}, \mathcal{Q}$ are the original sets. Then exists a set $\mathcal{R} \in \{\mathcal{P}, \mathcal{Q}\}$ such that $p \in (Cn^T(\mathcal{R}) \cap Cn^I(\mathcal{R}))$. As $\mathcal{R}$ is minimal and consistent, from lemma 2 follows that $p \in \mathcal{R}$, and thus $p \in \mathcal{P} \vee p \in \mathcal{Q}$. Similarly, $q \in \mathcal{P} \vee q \in \mathcal{Q}$. For each of $\mathcal{P}, \mathcal{Q}$ to be minimal, p and q cannot both belong to $\mathcal{P}$ or to $\mathcal{Q}$. Assume that $p \in \mathcal{P}$ and $q \in \mathcal{Q}$ (the other case would be treated analogically). Since $p \in \mathcal{P}$, and there is a rule (namely, q) in $\mathcal{Q}$ which has p in its truth-consequence, p cannot belong to TU nor TF; it thus belongs to $TT \cup TC$. Similarly, since $q \in \mathcal{Q}$, and there is a rule (namely, p) in $\mathcal{P}$ which has q in its importance-consequence, q cannot belong to TU_1 nor TF_1 (as it is important wrt. *both* sets); it thus belongs to $TT \cup TC$. Hence, both p and q belong to $TT \cup TC$, which is thus not minimal; this contradicts theorem 0 we have already proven □

Proof of consistency of $TT \cup TC \cup TU_1 \cup TF_1$ From the previous proof, we know that $TT \cup TC$ is minimal. By lemma 1, it thus suffices to prove that for every rule $r \in (TT \cup TC \cup TU_1 \cup TF_1)$, $\|r\|^T_{TT \cup TC \cup TU_1 \cup TF_1} = \mathtt{t}$.

We will carry out the proof by contradiction. Suppose there is a rule $r \in (TT \cup TC \cup TU_1 \cup TF_1)$ such that $\|r\|^T_{TT \cup TC \cup TU_1 \cup TF_1} \neq \mathtt{t}$. Since, obviously, $r \in Cn^T(TT \cup TC \cup TU_1 \cup TF_1)$, the only truth value wrt. $(TT \cup TC \cup TU_1 \cup TF_1)$ which remains possible for r is $\mathtt{c}$ (contradiction). From $\|r\|^T_{TT \cup TC \cup TU_1 \cup TF_1} = \mathtt{c}$ follows that there is a rule $r' \in (TT \cup TC \cup TU_1 \cup TF_1)$ such that $r \in Cn^F(r')$, i.e. $L(r') \geq_g L(r)$.

By (2) and by the specific properties of $TT \cup TC \cup TU_1 \cup TF_1$, we know that:

1. $(r \in Cn^T(\mathcal{P}) \wedge r \in Cn^I(\mathcal{P})) \vee (r \in Cn^T(\mathcal{Q}) \wedge r \in Cn^I(\mathcal{Q}))$
2. $(r' \in Cn^T(\mathcal{P}) \wedge r' \in Cn^I(\mathcal{P})) \vee (r' \in Cn^T(\mathcal{Q}) \wedge r' \in Cn^I(\mathcal{Q}))$

where $\mathcal{P}, \mathcal{Q}$ are the original sets. Then exists a set $\mathcal{R} \in \{\mathcal{P}, \mathcal{Q}\}$ such that $r \in (Cn^T(\mathcal{R}) \cap Cn^I(\mathcal{R}))$. As $\mathcal{R}$ is minimal and consistent, from lemma 2 follows that $r \in \mathcal{R}$, and thus $r \in \mathcal{P} \vee r \in \mathcal{Q}$. Similarly, $r' \in \mathcal{P} \vee r' \in \mathcal{Q}$. For each of $\mathcal{P}, \mathcal{Q}$ to be consistent, r and r' cannot both belong to $\mathcal{P}$ or to $\mathcal{Q}$. Assume that $r \in \mathcal{P}$ and $r' \in \mathcal{Q}$ (the other case would be treated analogically). Since $r' \in \mathcal{Q}$, and there is a rule (namely, r) in $\mathcal{P}$ which has r' in its importance-consequence, r' cannot belong to TU_1 nor TF_1 (as it is important wrt. *both* sets); it thus belongs to $TT \cup TC$.

From $r' \in (TT \cup TC)$ however follows that r' belongs to the truth-consequence of each of the source sets. Hence, there is a rule $r'' \in \mathcal{P}$ such that $L(r'') \geq_g L(r')$. By transitivity of $\geq_g$, $L(r'') \geq_g L(r)$. As both r'' and r belong to $\mathcal{P}$, if $R(r'') = R(r)$ then $\mathcal{P}$ is not minimal, otherwise $\mathcal{P}$ is not consistent $\square$

4.3.4 Problems and perspectives of the algebraic approach

The above method for knowledge integration is restricted to a rather weak formalism. It would be easily extensible to rules with negated literals allowed as conditions, and to categorial attribute-value representation. It would also be interesting to investigate whether a generalization of this method could be applied on *weighted* rules, as weights with bayesian semantic could easily be assigned to rules discovered in data; this might enable to unify the present approach with the weight compromizing technique described in section 4.4. In this respect, the notion of *minimality* introduced in section 4.3.1 should be compared with the notion of knowledge base minimality elaborated by Iványek [IvaSte88]. Another open problem is that of different *a priori credit* of both information sources; this would probably require a certain "fuzzification" of truth (and possibly importance) values introduced in our approach.

The formulation of the problem and the technique presented (in [Svatek95]) have inspired Posthoff et al. to the development of their rules/examples integration method, which is based on logical equations [PosScZ96]. The method involves computation of solution sets, which can be done efficiently by means of existing dedicated tools; in contrast to our method, it does not consider the notion of *importance*.

4.4 Integration of weighted rules from expert and from data

4.4.1 The problem of revision of weighted rules

In section 4.3 we have analyzed the task of learning with prior knowledge consisting of simple "logical" (i.e. non-weighted) rules. The task could be treated straightforwardly as that of knowledge revision, or, as we have shown, as that of knowledge induction and integration (the "bypass-model"); we have also suggested a simple algebraic approach for the integration subtask. We however know that expert knowledge often suffers from uncertainty and cannot be always represented in the form of "logical" rulesets. Let us now consider a more complex situation of *weighted* rules. Again, both the *revision* and *induction-and-integration* paradigms can be applied.

In section 3, we have stated that most inductive learning methods output knowledge in a "logical" form (i.e. without certainty factors). Similarly, most *knowledge revision* methods operate on logical theories or rulebases; revision of compositional rulebases is a lot more complex. It can be decomposed into *structure revision*, consisting in additions and removals of rule elements, and *strength revision*, i.e. adjustment of weights. Ling & Valtorta [LinVal91] have shown that even if the knowledge is represented by (weighted) propositional rules without chaining, and the data are noise-free, the strength refinement problem (i.e. tuning the weights to fit given data) remains NP-hard even for simple composition operations such as the probabilistic sum used e.g. in MYCIN [Shortl76].

We have approached the refinement problem from a different perspective, namely from that of knowledge *integration*, analogically to section 4.3. Instead of adapting the expert rules to one data object in turn, we assume that *empirical* rules are first discovered in data and then integrated with expert rules. We limit ourselves to the case of a single, isolated expert rule.

Let us suppose that the expert asserts a rule, and immediately associates a certainty factor, i.e. *weight*, with it: "I think that rule r holds with weight $w_E(r)$". Let us accept the pseudo-bayesian viewpoint, from which $w_E(r)$ can be considered as a subjective estimate of probability of conclusion (class) c given condition $Cond$. Then it makes sense to compare the weight $w_E(r)$ of expert rule r with the relative conditional frequency of c given $Cond$ in a dataset D, i.e. with the *validity* (cf. the definition in section 3.2.2) $Val_D(r)$.

We can distinguish three situations:

1. The coverage of r in D, $Cov_D(r)$, is equal to 0 ($Val_D(r)$ is then undefined)
2. $Val_D(r) = w_E(r)$
3. $Val_D(r) \neq w_E(r)$

The first case means that no information can be found in data to support or reject the expert rule. In the second case, the data perfectly match the expert's estimate of rule weight and we will probably keep the rule together with the agreed weight. In the third case, the expert and the data "disagree" and we are facing a "conflict".

In reality, the second case is quite rare and may occur rather as a product of chance than as a real expression of agreement between the expert and data. We thus need to replace the requirement of equality of $Val_D(r)$ and $w_E(r)$ with a more moderate one. Intuitively, we would suggest a verbal form similar to: $Val_D(r)$ *does not significantly differ* from $w_E(r)$. What the term "significantly differ" means can be defined in various ways; most likely, we will utilize statistical or heuristic measures.[54]

The conflict which arises for a rule with different expert weight and validity in data is an instance of the knowledge revision problem. Since our universe is that of weighted rules, we have the liberty of choice between *structure revision*, i.e. removal of the rule from our knowledge, and *strength revision*, i.e. adjustment of weight; the last possibility would be to keep the rule with the "original" weight (i.e. that given by expert).

4.4.2 Interpolation between weight in data and weight from expert

The weight adjustment task can be formulated as follows: having a rule r with weight $w_E(r)$ assigned by the expert, and a set of data objects D, we want to obtain a new, adjusted weight $w(r)$ as a "compromise" between the expert and the data - or (we anticipate here what comes later), an *interpolation* between the expert weight and the weight in data.

The adjusted weight can be expressed as a result of an *interpolation function* γ, which takes into account all relevant information. In our simple model, relevant information can be summarized into three numerical values such that other (especially $Val_D(r)$) are dependent on them[55]:

1. The weight given to r by the expert: $w_E(r)$.
2. The coverage of r in data: $Cov_D(r)$.
3. The correct-coverage of r in data: $CCov_D(r)$.

Hence:

$$w(r) = \gamma(w_E(r), Cov_D(r), CCov_D(r)) \tag{3}$$

Let us now formulate some requirements which should be fulfilled by any "reasonable" γ:

1. If $Cov_D(r) = 0$ then $\gamma = w_E(r)$.
2. If $Cov_D(r) > 0$ then $((w_E(r) \leq \gamma \leq Val_D(r)) \ \vee \ (Val_D(r) \leq \gamma \leq w_E(r)))$.

[54] We should not omit a third type of measure, the "statistically-heuristic" one, which is undoubtedly most popular within the AI community. It consists in applying statistical measures while relaxing some of their pre-conditions; we thus obtain heuristic measures which are often (but not always) more plausible than "ordinary" rules-of-thumb. As we will see later, this was the case for the present work, too.

[55] The definitions of coverage and correct-coverage can be found in section 3.2.2.

3. If $Cov_D(r) > 0$ then: if for two rules r, r' with validities $Val_D = Val_D(r) = Val_D(r')$ and expert weights $w_E = w_E(r) = w_E(r')$ holds $Cov_D(r) \leq Cov_D(r')$ then for the respective values of γ, denoted as γ and γ', holds $|Val_D - \gamma| \geq |Val_D - \gamma'|$.

The first requirement states that if the rule does not cover any data objects, the function should return the expert weight (no adjustment takes place). This requirement could be nicknamed as *vacuity* requirement, after one of Gärdenfors' postulates (cf.[Garden88]) for closed theory contraction, which has a similar semantic.

The second requirement is also quite natural: it states that the result of γ should lay *in the interval* between the expert weight and the weight computed from data (so that we can use the term "interpolation function").

The last requirement is derived from the assumption that with increasing coverage of the rule, we can trust the data more (or, at least, not less). From the statistical point of view, this amounts to the impact of *sample size* on the plausibility of estimate.

4.4.3 Heuristic interpolation function based on confidence interval

Statistical techniques have long been a decent source of inspiration for AI researchers. Here, for the instantiation of the interpolation function, we have borrowed the notion of *confidence interval*. Our *heuristic interpolation function* γ_{ci} is, for a given dataset D and a rule r, defined by two partial functions[56]:

If $Cov_D(r) = 0$ or $|w_E(r) - Val_D(r)| \leq \epsilon$, then

$$\gamma_{ci} = w_E(r) \tag{4}$$

otherwise

$$\gamma_{ci} = p.(Val_D(r) + \epsilon_{sign}) + (1 - p).w_E(r) \tag{5}$$

where

- $\epsilon_{sign} = \epsilon.sign(w_E(r) - Val_D(r))$; ϵ is the radius of bilateral (symmetrical) confidence interval for $Val_D(r)$, output by a certain statistical test T on data D.
- p is a heuristic "plausibility function" of the interval estimate, which takes values from the closed interval $[0, 1]$ and is monotonically increasing with $Cov_D(r)$.

The function may appear complex, its behaviour is nevertheless quite transparent. If the weight given by expert falls within the confidence interval, it is considered as (heuristically) "not rejected by data" and taken as such. If it falls beyond this interval, the result should intuitively be set to the interval boundary closer to $w_E(r)$; it is however weighted by the plausibility function, which further decreases the impact of data for very small sample sizes. At Fig. 25 we show an example of behaviour of such function.

It is not difficult to verify that such interpolation function satisfies the requirements declared in section 4.4.2.

[56]The definition has been first presented in [Svatek94a].

Figure 25: Graph of behaviour of interpolation function γ_{ci}

Proof for property 1 The requirement is fulfilled by definition.

Proof for property 2 The "interpolation" property follows from the fact that (for $Cov_D(r) > 0$) γ_{ci} is a linear combination of $w_E(r)$ and $Val_D(r) + \epsilon_{sign}$ such that the sum of coefficients equals to 1 ($p + (1 - p) = 1$). The result of such a combination necessarily lays between the values of both operands. Since the interval between $w_E(r)$ and $Val_D(r) + \epsilon_{sign}$ is a subinterval of the interval between $w_E(r)$ and $Val_D(r)$, any value laying in the former must also lay in the latter. For the limit case when $Cov_D(r) = 0$, the requirement is fulfilled by definition.

Proof for property 3 The "monotony" property follows from the assumption that the radius of confidence interval is to be non-decreasing with increasing sample size, for any statistical test. The set of values of $Cov_D(r)$ for which $|\, w_E(r) - Val_D(r) \,| \leq \epsilon$ is thus a lower set of the set of all values of $Cov_D(r)$, for fixed $w_E(r)$ and $Val_D(r)$. The value of γ_{ci} is equal to $w_E(r)$ in this case (according to (4)); from the interpolation property follows that for any value of γ_{ci}:

$$|\, Val_D(r) - w_E(r) \,| \geq |\, Val_D(r) - \gamma_{ci} \,|$$

i.e. the difference function $\Delta(Cov_D(r)) = |\, Val_D(r) - \gamma_{ci} \,|$ takes its maximum for $\gamma_{ci} = w_E(r)$. Now, we will analyze the situation when $|\, w_E(r) - Val_D(r) \,| > \epsilon$, $w_E(r) > Val_D(r)$. Then the value of difference function is

$$\Delta(Cov_D(r)) = (1 - p).(w_E(r) - Val_D(r)) + p.\epsilon$$

which is non-decreasing for non-decreasing ϵ (and thus for increasing $Cov_D(r)$). Analogically, for $|w_E(r) - Val_D(r)| > \epsilon$, $w_E(r) < Val_D(r)$,

$$\Delta(Cov_D(r)) = (1-p).(Val_D(r) - w_E(r)) + p.\epsilon$$

with the same result. Altogether, as $\Delta(Cov_D(r))$ is maximal in the lower set of the domain and non-decreasing in the rest of the domain, it is non-decreasing in the whole domain. The property 3 thus holds for any pair of rules.

4.4.4 Problems and perspectives of the weight interpolation approach

The ultimate aim (cf. [Svatek94b]) of the above described technique was to enable integration of (multiple) weighted rules suggested by an expert with rules extracted from data by means of the ESOD method. This aim however showed to be somewhat spurious. The rulebase learned by means of ESOD is closely linked to data from which it was learned. Any modification to the rulebase, such as weight adjustment, disturbs this link - some rules are influenced by the expert while some other are based on data only. If these rules are syntactically dependent, a serious inconsistency may occur. It can be proven that even a single, isolated weight adjustment may give rise to a rulebase which does not correspond to the original dataset any longer (this could be still viewed as more-or-less desirable) and even to *any dataset at all*, due to an inconsistent probability structure. Empirical experiments with a clone of the Knowledge Explorer system have shown that this situation is typical rather than exceptional[57].

To overcome the above problem, every weight adjustment would have to be accompanied with additional probability propagation steps. The choice of these steps, however, would not be fully deterministic, since the consistency of conditional probabilities could be achieved in different ways. The integration process could then be guided by *heuristics* and/or by a human *oracle* (the expert himself).

Another solution could be to resign on the real domain of weights and to introduce some *qualitative* weights instead, such as:

- "certainly 100%" rules,
- "possibly 100%" rules,
- "certainly above 50%" rules,
- "possibly above 50%" rules.

[57]For the purpose of experiments, we have implemented the interpolation-function-based weight adjustment mechanism in a very simple form, using a standard-deviation test (at significance level 0.05), where ϵ is understood as standard error of random choice from alternative distribution; the plausibility function p was derived from a sample-size heuristic commonly used in the context of this test. For every rule generated from data, the expert rulebase was examined; if an identical (but the weight) rule was found, the result of the interpolation function was computed and tested against the composed weight of subrules.

This type of weights would probably be easier to obtain from an expert, and could also make the propagation of weights more tractable. The rules could be organized into different *layers* and the weight adjustment could be performed according to a look-up table[58] instead of numerical computation.

æ

[58]The achievements of non-monotonic logic (research on stratified rulesets, cf. e.g. [Przym88]) could be taken into account.

5 Learning with hierarchies and constraints as static domain knowledge - methodology and implementation

5.1 Objectives

In section 2.2 we have described some existing techniques of learning with prior static domain knowledge. They have been tentatively divided into two groups - those where prior knowledge *constrains* the search space (section 2.2.2) and those where prior knowledge *extends* the search space (section 2.2.3). In addition, we have focused on projects involving two particular forms of such knowledge - value hierarchies and/or integrity constraints (section 2.2.4), namely the forms which are treated in this section describing original research.

Our project carried out in 1995-1996 concentrated on the following problem, in its more general forms as well as in the specific context of the ESOD method (the background project, see section 3):

Problem 3 *Which forms of static domain knowledge are applicable in learning from categorial data? How can their exploitation be incorporated into the ESOD learner as well as reasoner (classifier)?*

So far, three forms of static domain knowledge have been investigated: hierarchies of attribute values, hierarchies of classes and integrity constraints. A new implementation of ESOD can exploit all of them; it has been tested on three real-world tasks. First testing results suggest that the impact of hierarchies may consist in better comprehensibility as well as in higher reliability of classification; integrity constraints reduce the learning time under specific conditions, which should be assessed in advance.

An additional problem that arose in this context was the inavailability of hierarchical background knowledge in some cases. This entailed the need for *automatic* methods for hierarchy-building:

Problem 4 *Can hierarchies of input attribute values and of classes be constructed automatically from data? Are they meaningful (compared to hierarchies suggested by expert)?*

For this purpose, a simple hierarchical clustering technique has been suggested, which, unlike most such techniques used in ML, accounts for different extrinsic dissimilarity of values of attributes on which the clustering is based (so-called source attributes). The testing results are promising, but very preliminary.

5.2 Abstraction hierarchies

5.2.1 The notion of abstraction hierarchy

A hierarchy is understood as a transitive partial ordering over a set of elements, such that one of the elements ("root") precedes all others; the semantic of this precedence is usually

that of "superiority" of some kind. In artificial intelligence, two types of hierarchies are most often used (in particular, within semantic networks): abstraction ("is-a") hierarchies and decomposition ("part-of") hierarchies. Decomposition hierarchies are used to describe the physical decomposition of some system - one object is a part of another (see the example at Fig. 26). Abstraction hierarchies have to do with the degree of abstraction of terms, in the domain language - one notion is a special case of another (see the example at Fig. 27). An important property of abstraction hierarchies is the assumption of *inheritance*: if we know that the entity X has property p, we can expect that entity Y which is a special sort of X (Y is-a X) also has property p.

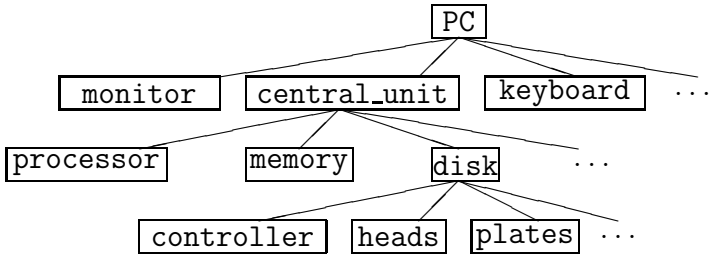

Figure 26: Example of decomposition hierarchy

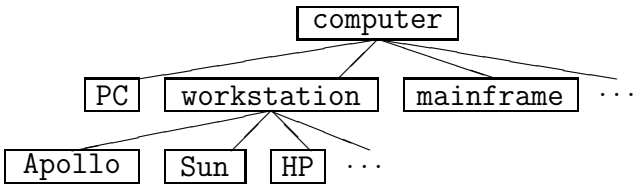

Figure 27: Example of abstraction hierarchy

In a language restricted to combinations of attribue-value pairs, hierarchies can be defined either on the attributes themselves, or on their domains (sets of values). An abstraction hierarchy defined on the set of attributes may regroup them into abstract classes of attributes; further domain knowledge can e.g. specify that attributes from certain groups often appear together in a rule, etc. We will not, however, consider hierarchies of attributes in this work, and will instead concentrate on the hierarchies of *attribute values* - both for input attributes (section 5.2.2) and the goal attribute (section 5.2.3).

5.2.2 Hierarchies of input attribute values

The notion of attributes with hierarchically structured domain (also structured, tree-structured, taxonomic...) is not new in the ML literature, cf. [AlmAkK95], [AroPrB96], [Nunez91]. Here, we adopt one of the common ways of describing it: Let a be an attribute and $V = V_O \cup V_A$ the domain of its values. V_O is the set of *observable* values (which can be observed on data objects), while V_A is the set of *abstract* values. V_O must contain at least two elements, and V_A must contain at least one element - the universal value *any*. As a *hierarchy* on a, we consider a directed acyclic graph[59] on V, with one source - the value *any* - and the sinks corresponding to values from V_O; internal nodes correspond to values from V_A. A hierarchy can be defined on a nominal or ordered attribute - in the latter case, each value from V_A should subsume a contiguous sequence of values from V_O[60].

The most typical examples of value hierarchies used in machine learning are probably taxonomies of natural objects, geographical/administrative partitionings, and linguistic thesauri. The application described in [AroPrB96], for example, involved the tree structure of US geographic areas, a large botanical taxonomy, and a smaller hierarchy of climate types. In the orienteering application mentioned in this paper (section 5.5.1), the largest hierarchy dealt with terrain objects of distinguishable size. Almuallim et al. [AlmAkK95] applied their method to trees of linguistic concepts.

The way how abstract values are introduced into target knowledge structures much depends on the particular learning technique: usually, some form of general-to-specific search is performed. Learning rules with abstract values is a natural means to improve the comprehensibility of target knowledge structures, as these shift the classification process to a more *abstract level*; cf. the notion of *heuristic classification* introduced by Clancey [Clanc85], with the abstraction of Variables from Observables as its first step. Another potential benefit is the *conciseness* of such rulebase, as one rule with abstract values expresses a hypothesis otherwise dispersed in several (lower-grained) rules. Furthermore, if the learner is biased towards high *coverage* to avoid overfitting, abstract (or anyhow conjoined) values are necessary for any rules to be induced from smaller datasets, as individual values have too low frequency.[61]

Let us make a brief remark on the relation between generalization and abstraction. According to [Saitta95], abstraction is an "ability to change the level of details of a representation"; in machine learning, it could "give solutions to the fundamental dilemmas

[59] It need not be a tree, as some branches may conflue; namely if the hierarchy addresses multiple ways of abstraction. For illustration, see the two fragments of a hierarchy at Fig. 31, in section 5.5.

[60] We do not consider partitions of abstract values orthogonal to the ordering of observable values, such as "even-odd" for integer values.

[61] The last two statements may seem somewhat contradictory; actually, without using the hierarchies, there is a tradeoff between a large size of the knowledge base and incapability to detect important relations. Decision trees are biased towards the former; therefore, Núñez [Nunez91] observed an abrupt decrease of size when using hierarchies. Rule-discovery systems with statistical constraints on coverage tend to the latter: hierarchies were thus needed to detect interesting rules by our learning algorithm in the orienteering domain (see section 5.5.1), similar observations have been presented also by Aronis et al. [AroPrB96].

involving the trade-off between knowledge simplicity and predictivity, knowledge meaningfulness and task-dependency". Abstraction should be viewed as complementary to generalization, as "generality is an extensional property of concepts and is based on instance set inclusion", while "abstraction is an intensional property and is based on hypothesis information content".

In this respect, rule learning with hierarchical attributes consists of both generalization and abstraction. As an example, consider a learning task aiming at finding rules which should determine whether a certain project environment (including e.g. hardware, software, legal conditions and human resources) is suitable for developing knowledge-based systems. The input dataset describes environments which have proven suitable or unsuitable for KBS design, in the past. In each data record, the "hardware" option is expressed by a hierarchical attribute, whose domain of values corresponds to the tree in Fig. 27. Let us assume that, in the dataset, all environments where hardware was `Apollo`, `Sun` or `HP` are labelled suitable for KBS design. A pure generalization would yield e.g. the rule

```
hardware = (Apollo OR Sun OR HP) --> class = suitable
```

while generalization combined with abstraction would yield

```
hardware = workstation  --> class = suitable
```

The first rule is more complicated - it contains an *internal disjunction*[62] - while the second is simpler and more meaningful.

Let us recall the notion of learning *bias*, introduced in section 2.2.1. With regard to the common distinction of bias types - *representational*, *procedural* and *sample* bias, the impact of input attribute value hierarchies amounts to representational bias: it *extends* the language in which the inductive hypotheses can be expressed.

The concept of *hierarchical attributes* and of "generalization as climbing up in a hierarchy" was notoriously mentioned in the mainstream ML literature - the "climbing-up-hierarchy" operation was known to be a generalization technique alternative to the common "drop-literal" one. Nevertheless, it was mainly used for "textbook", demonstrative purposes, in connection with small example sets (e.g. the notorious taxonomy of simple geometric objects). Only recently, attempts appeared to exploit hierarchies to discover important relations in *raw data*. This is the case of the RL knowledge discovery system [ClePro90], which has been used in several domains to discover useful relations; its more advanced successor KBRL [AroPrB96] uses role links in addition to taxonomies. The phenomen of tree-structured attributes has also been studied in the context of decision-tree learning, by Núñez [Nunez91] and Almuallim et al. [AlmAkK95], with the primary aims of cutting down computation time, increasing accuracy and decreasing the size of

[62]Internal disjunctions have been exploited in the AQ family of learning algorithms, in connection with Michalski's annotated predicate calculus [Michal83]. Construction of internal disjunctions can also be viewed as a simple form of *feature construction*, namely conjonction of nominal values of attributes; cf. the note on constructive induction, section 2.2.3 of this work.

the tree learned (although comprehensibility was also mentioned as a secondary issue). Our approach seems to lay between the "discovery" and "performance" streams, as the systematic search method of ESOD is akin to discovery techniques, while it disposes an embedded performance element. Unlike [AlmAkK95], we assume that the primary input source is the data, the size of the hierarchies being moderate; the common point of both approaches is the search method, which consists of pre-computing the frequencies for all abstract values, followed with search for a kind of optimal representation. Within the KDD stream, hierarchies (even non-tree dags) are also used to aggregate data from large databases, along the so-called *domain generalization paths* [HamHiC96]; these lead bottom-up, from most specific (leaf) values toward the root ("any" value), thus being reversions of top-down explanation paths we introduce in this work, in the context of class hierarchies (see section 5.2.3).

Abstraction hierarchies (and is-a relations within semantic nets) have been also accepted as a rudimentary form of domain knowledge for *relational* systems, such as the apprenticeship-learning systems DISCIPLE [TecKod90], APT [NedCau92] and ODYSSEUS [Wilk90]. Another related approach may be that of EITHER, a knowledge revision system, one version of which was adapted to constructive induction [MooOu91b]. EITHER works with chained rules on propositional concepts; some of these rules are similar to is-a links (e.g. "has_handle → graspable"). If such rules lead from observable features to intermediate features, the truth-values of the latter can be deduced for data objects by forward chaining, added to the descriptions of these objects and used as input features in the subsequent empirical induction process. However, EITHER's rules can be more complex (at least, they mostly involve conjunctive antecedents) and their semantic may only incidentially become that of concept generalization.

5.2.3 Hierarchy of classes

While hierarchies on input attributes have been used in the ML research for some time, little attention has been paid to the possibility of exploiting hierarchies of *classes*. Our understanding of class hierarchies can be characterized in three dimensions:

Syntax A hierarchy defined on the goal attribute is syntactically identical with hierarchies on input attributes, see previous section.[63]

Semantics For each path leading from the *any* node downto a leaf node, there are (at least) two possible interpretations (semantics); it can be viewed as either

- a *refinement* path, along which the initially coarse conclusion is refined (specialized);
- an *explanation* path, which explains the specific (leaf-level) conclusion given in advance.

[63] In our work, we have so far considered only tree-shaped class hierarchies - this limitation should be overcome in the future.

Pragmatics Classification (in the form of refinement) as well as explanation can be performed by means of rulesets linked to each non-leaf node of the hierarchy.

A learning algorithm can be used to provide each non-leaf node in the hierarchy with a partial ruleset[64], which is able to assign a data object into one of its immediate successor nodes. To induce such a ruleset, only a subset of data is used, and the class of each object replaced with that of the corresponding immediate successor node (see Fig. 28). This "macro-learning" algorithm is applicable irrespective of the particular learning technique.

The resulting hierarchical rulebase can be used for *classification*, which has the form of class refinement (see Fig. 29); again, the macro-algorithm can be coupled with any sort of classifier (compatible with the learner used for induction).

Another possibility is to perform two separate learning tasks - one with the class hierarchy, yielding a *hierarchical rulebase*, and the other without it (learning on the whole dataset), yielding a *flat ruleset*, which discerns among all leaf classes at once. We can then separate the *problem-solving* task (i.e. classification) from the *explanation* task - the former can be performed using the flat ruleset while the latter using the hierarchical rulebase. The explanation has the form of path from *any* to c, $E = (c_0 = any, c_1, c_2, \ldots, c_{n-1}, c_n = c)$, where c is the leaf class returned by the classifier based on the flat ruleset; within each node c_i in the path, those rules are displayed which support the conclusion c_{i+1}. If the classifier returns c_{i+1} for each c_i in E, $0 \leq i < n$ (based on ruleset R_i linked to c_i), then we will call E *perfect-fit explanation* - this also means that the hierarchical macro-classifier would agree with the "flat" classifier. An example of such explanation can be found at Fig. 34 in section 5.5.2. Fit can be also expressed numerically, as "m-of-n", where $m \leq n$; $m = n$ (i.e. 1 of 1, 2 of 2 and so forth) corresponds to perfect fit.

We assume that the use of class hierarchies can improve comprehensibility if the individual refinement steps are more *transparent* than the one-shot classification performed by the flat ruleset. The transparency of classification is, in turn, likely to increase with the decreasing number of rules which participate on the derivation. The plausibility of explanations naturally depends on the relevance of the hierarchy to the classification process: if the expert followed the same or similar hierarchy when assigning the training objects to classes, then the explanation attempts to *reconstruct* his behaviour, in contrast to the "shortcut" derivation performed by the flat ruleset; we can view it as *reconstructive explanation*. Experiments (section 5.5) suggest that the imperfect fit of explanation can also serve as indicator of lowered *confidence* in the conclusion, thus having indirect impact on accuracy; moreover, in some situations, the hierarchical rulebase can be itself more accurate that the flat ruleset.

In terms of learning bias, class hierarchies provide *representational* bias, as the language of classes (right-hand sides of target rules) changes due to relabelling, but also *instance* bias, as only a subset of objects is picked up in each micro-learning step.

Step-by-step refinement of the conclusion has first appeared in man-made expert systems. Early diagnostic expert systems like MYCIN or PROSPECTOR (which are related

[64]Instead of a ruleset, we can naturally conceive a different knowledge structure such as a decision tree or list.

Input: dataset O, class hierarchy H_C.
Output: hierarchical rulebase (H_C with a ruleset linked to each non-leaf node).
Computation:
At every non-leaf node c of the hierarchy, we

1. determine the set O_c of objects from O whose class c_i is successor of c in H_C;
2. replace the class of these objects with c_j which is immediate successor of c and predecessor of c_i[65] in H_C - we obtain a dataset O'_c;
3. run LEARNER on O'_c, which yields a ruleset R_c;
4. link R_c to node c.

Figure 28: Macro-learning algorithm (induction of a hierarchical rulebase)

Input: object o, class hierarchy H_C with linked hierarchical rulebase.
Output: one of the leaf-classes from H_C.
Computation:

1. Let NODE = *any*.
2. Let R be the ruleset linked to node NODE in H_C. Let c be the class returned by CLASSIFIER for o, based on R. Let NODE = c.
3. If NODE is a leaf-class then stop and return NODE, else go to 2.

Figure 29: Macro-classification algorithm

to ESOD in the sense of reasoning mechanism) were rather knowledge-monolitic: they reasoned from symptoms to hypotheses in one shot (be it through multiple layers of rules) and did not provide explanations other than simple chains of rules. Later, however, more modular systems have been built, where different forms of knowledge participated on the problem-solving task. An important improvement was the shift to successive *refinement* of the initial hypothesis; hierarchically structured knowledge also played an important role - cf. e.g. the work on NEOMYCIN, by Clancey [Clanc88]). This trend was recently reflected in the work on reusable problem-solving methods, e.g. within the KADS methodology. The KADS library of generic task models [TanHay93] distinguishes three generic forms of classification: simple classification, heuristic classification and systematic refinement. In this respect, we view the reasoning performed with the use of hierarchies (as shown in sections 5.2.2 and 5.2.3) as simple classification involving some aspects of heuristic classification (the abstraction of features of the given data object) as well as of systematic refinement (the refinement of classes in the class hierarchies).

Our use of class hierarchies for generating explanations seems to be novel. However, many approaches exist which exploit various forms of knowledge to break the classification task into simpler steps or to give deeper insight into it: the ENIGME system [ThoLaG93] is guided by a KADS model of expertise of the given task, the learning method of Clark & Matwin [ClaMat93] uses qualitative models of the domain, Bruha [Bruha95] uses man-made decision trees etc. All these approaches rely on *control* knowledge elicited from expert. In contrast, our method requires only static domain knowledge; this of course entails that the explanation proposed is fully valid only if the static knowledge structure (abstraction hierarchy) captures the refinement steps of the undercover problem-solving procedure. The problem of hierarchical explanations can be also compared with the current research in explanation for expert systems. One of the hot issues seems to be the generation of *reconstructive* explanations in addition to trace-based ones [Wick94]; our separation of ("shortcut") classification and step-by-step explanation can be viewed as a humble move in this direction. Moreover, the idea of separate classification and explanation has already penetrated to ML; note e.g. the GEM system [VMeDec95], where explanation arises as axis-parallel approximation of the complex representation found by an instance-based neural learner; van Someren [VSom95] explicitely distinguishes *learning for problem-solving* and *learning for explanation*, which corresponds to our separate learning of flat ruleset and hierarchical rulebase.

5.2.4 Constructing value hierarchies

The above methods for exploitation of value hierarchies assume that the hierarchies are given in advance. The source can be a human expert, but, as taxonomy knowledge often belongs to common knowledge of the domain, it can be as well a textbook or a person who is familiar with the domain only to some extent. In some situations, however, even such a source is not available; then we can consider building hierarchies automatically. This task is akin to two well-known ML tasks:

- *conceptual clustering*, which groups unclassified instances into tentative classes based on values of input attributes (see e.g. [Fisher87], [GenLaF89], [OliBaW96]),
- *class-sensitive discretization* of continuous input attribute values into intervals (see e.g. [Catl91], [LeeShi94]).

Both these tasks consist in joining together distinct values of an attribute wrt. the values of other attribute/s.[66] However, the former restructures the domain of the goal attribute based on values of input attributes, while the latter the opposite. To avoid ambiguity, we will now denote the attribute to be restructured as *target attribute* (TA) and the attribute on whose values the restructuring is based as *source attribute* (SA). As criteria deciding about the utility of the join (or, alternatively, about the similarity of the TA values to be joined), various measures based on empirical conditional probabilities of SA values given TA values have been used: the common idea of all of them (Fisher's category utility, information gain, MML measure, Hellinger's divergence) is the effort to maximize internal similarity and minimize mutual similarity of TA values in terms of SA values; those TA values should be joined which correspond to similar distributions of SA values, so that the join does not substantially increase the average "impurity". Different SA values are however treated as having always uniform (i.e. maximal) *extrinsic dissimilarity*, with the exception of the CLASSIT system [GenLaF89], which utilizes a standard-deviation-based measure to handle continuous source attributes.

In our work, we want to address the problem of a larger class of source attributes, namely attributes with *hierarchically* structured and/or *linearly* ordered domain. For this, we need an objective function capable of reflecting extrinsic dissimilarity unique for each pair of SA values. We demonstrate the concept of dissimilarity among values on a simple example of military grades (see Fig. 30). The full set of military grades is linearly ordered; in addition, there are (contiguous) groups of grades which share many properties, and can be viewed as abstract values in a hierarchy. A heuristic dissimilarity measure should comprise both factors; we suggest the following form, without claiming that it is necessarily the best one:

$$Diss(x,y) = HDiss(x,y).LDiss(x,y)$$

where

- $HDiss(x,y)$ is *hierarchical* dissimilarity:

$$HDiss(x,y) = 1 - \frac{(\|e(x)\| + \|e(y)\|) \,.\, \|e(x) \cap e(y)\|}{2.\|e(x)\|.\|e(y)\|}$$

[66] This is the bottom-up view which we keep throughout this paper. In top-down view, we want to split the domain of values instead; the problem of construction of the objective function, which is focal in this section, is however independent of the search method. Finding a hierarchical structure optimal wrt. such measure typically requires a combination of joining and splitting steps - see [Fisher96] for an overview.

where $e(x)$ is the set of edges in the hierarchy which belong to some path leading from *any* to x (each edge appears only once); $\|.\|$ denotes set cardinality.[67]

- $LDiss(x, y)$ is *linear* dissimilarity:

$$LDiss(x, y) = \frac{\mid Ord(x) - Ord(y) \mid}{Total - 1}$$

where $Ord(x)$ is the ordinal number of x in the linear sequence and $Total$ is the total number of values; it is thus the relative distance of values in the ordering. This kind of dissimilarity measure has been used previously in ML research (e.g. in [KrePoS95]).

The whole $Diss(x, y)$ as well as its components are symmetric and take values from $[0, 1]$; $HDiss(x, x) = LDiss(x, x) = Diss(x, x) = 0$ for any x. Given the structure of military grades above, we can e.g. compute hierarchical dissimilarity between the values "corporal" and "lieutenant" as 2/3 and linear dissimilarity as 2/5, thus $Diss(corporal, lieutenant) = 0.267$. If a clustering algorithm is to build a hierarchy of TA values based on the values of the source attribute "grade" it should strive to join together not only TA values which correspond to the same grade, but also TA values which correspond to different but similar grades. It must then take into account the dissimilarity of SA values, which is extrinsic to the clustering itself.

As a measure expressing the degree of internal *cohesion* of a dataset in terms of SA values, we use the following formula:

$$Coh(A) = \frac{\Sigma_{i=1}^{n} \Sigma_{j=1}^{n} \|a_i\| \, \|a_j\| \, (1 - Diss(a_i, a_j))}{n^2} \tag{6}$$

where n is the number of values of source attribute A and $\|a_i\|$ is the frequency of value a_i in data. This measure roughly expresses the average degree of instance dissimilarity within the whole data. To evaluate the utility of a partition of target attribute B into m values, we can further compute aggregated cohesion given B, in a way analogical to information-theoretic or probabilistic approaches ($\|./.\|$ stands for conditional frequency):

$$Coh(B, A) = \Sigma_{k=1}^{m} P(b_k) \frac{\Sigma_{i=1}^{n} \Sigma_{j=1}^{n} \|a_i/b_k\| \, \|a_j/b_k\| \, (1 - Diss(a_i, a_j))}{\|b_k\|^2} \tag{7}$$

For multiple source attributes, the overall cohesion can be calculated as the average value of (7).[68] This typically occurs in hierarchization of the class attribute given a set of input attributes.

Exploitation of constructed hierarchies, with each non-leaf node representing merely a union of observable values (without a semantic meaning of its own) differs from exploitation of man-made hierarchies: the former merely enable *generalization* while the

[67] An advantage of this measure is its applicability to non-tree dags.

[68] Different weights (importances) of attributes can be taken into account if this form of background knowledge is available.

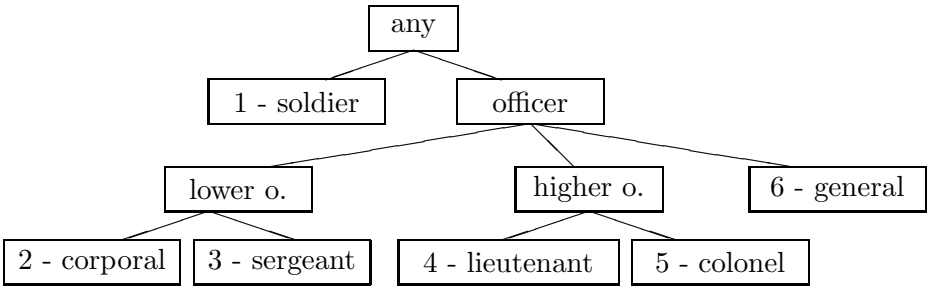

Figure 30: Example hierarchy of military grades

latter enable *abstraction*, which has stronger impact on comprehensibility. As the hierarchization step is separated[69] from the rule-learning step, they should (in fully-blown applications) be interleaved with another step - *interpretation* of constructed values (unions) by an expert, and possibly rejection of some obviously meaningless unions.[70] The risk of accidential formation of value unions is higher for input attributes, where the source information is limited to one (i.e. class) attribute; it seems that for plausibility of such hierarchization, it is necessary that the the single source attribute has at least a richer value structure (hierarchical and/or linear with multiple values, as illustrated above). An example of tentative interpretation of conjoined values can be found in section 5.5.3.

As a predecessor to our *hierarchization* technique, we can see the on-line value-unioning operation suggested also by [Nunez91], as well as other techniques exploiting the so-called internal disjunction. The difference lays in the off-line position of hierarchization, which is prone to be followed by interpretation of unions, prior to learning; unlike [Nunez91], our technique is adapted to nominal as well as ordered attributes. From another point of view, our technique is closely akin to *conceptual clustering* (of classes) [Fisher87], [GenLaF89], [OliBaW96] and to *discretization* (of input attributes) [Catl91], [LeeShi94]. Its difference from the bulk of clustering techniques consists in the account of extrinsic-dissimilarity, as presented in section 5.2.4; among the previous clustering systems, it is only CLASSIT which considers extrinsic dissimilarity but only in the specific sense of continuous attribute values. The same holds for discretization techniques, among which the one proposed by Lee and Shin [LeeShi94] is most similar to our hill-climbing clustering in the search method (but not in the objective function).

[69]This distinguishes our approach from that of Núñez, where the value-merging operation (of input attribute values) is performed online, during the construction of a decision tree. [Nunez91] mentions the use of existing hierarchies and the construction of unions as substantially different operations.

[70]This interpretation problem is mentioned e.g. by Catlett [Catl91], in the context of finding intervals of continuous attributes.

5.3 Integrity constraints

In models of human reasoning, two important types of statements can be identified: positive and normative[71]. Both types of statements can appear as a conclusion of a sort of rule. This implies that also two types of "rules" can be distinguished; let us call them *inference rules* and *integrity constraints*.

An inference rule infers, from the validity of a certain statement, the validity of another statement; an integrity constraint, in contrast, infers, from the validity of a certain statement, the *requirement* of validity of another statement. In this respect, an inference rule and an integrity constraint can deal with the same matter; in an extreme case, they may even look quite similar. Consider the following example:

If Mr. X works as private lawyer then he must have a university degree.

The rule can be read as an inference rule: from the profession of Mr. X, you can deduce that he (almost) certainly has the degree. If you are a journalist from a local newspaper, you can e.g. trust that the article on legal issues he sent you is sound and should be published.

The same rule can also be read as an integrity constraint: a university degree is required for the profession and, if Mr. X lacks it, there is a spot on his "moral integrity". If you are the same journalist but this time wanting to scandalize Mr. X, you are seeking for negative evidence of his degree...

In sections 2.2.2 and 2.1.2 we have come across multiple feature of the MOBAL system. MOBAL also uses integrity constraints in the form of disjunctive Horn clauses - these are syntactically richer than inference rules, for which a single literal is allowed as head. If the left-hand side of the constraint is satisfied then the right-hand side is checked; if it evaluates false, a message is output ("Integrity constraint violation in...") and an inconsistency-solving task is added to the agenda of tasks. The principles of integrity constraints of this type are probably borrowed from the field of deductive databases [Bry93]. There, the goal is to keep a database consistent through multiple updates.

Nevertheless, integrity constraints, even in a simpler form, can be also used for guiding empirical induction from categorial data. They prevent the inclusion of "forbidden" combinations into the set of potential left-hand sides of rules to be learned. If we recall once more the distinction of bias types from section 2.2.1, we see that the impact of integrity constraints can be characterized as *representational* bias, similarly to hiearchies of attribute values. This time, however, the bias *reduces* the hypothesis space.

[71] This is a general topic which appears in numerous disciplines such as philosophy, psychology, law etc. For the sake of brevity, we will however treat it rather pragmatically, i.e. merely in the sense in which it "protrudes" into the applied artificial intelligence research.

5.4 Implementation

5.4.1 Hierarchies and ESOD

We have implemented separately the learning and classification algorithms[72] from section 5.2.3 and a hierarchization algorithm based on similarity measures from section 5.2.4. The former are grafted upon the ESOD method of learning a set of weighted rules from observational data (including the compositional mechanism used for classification of new instances). ESOD (and its recent implementation named KEX) has been described in more detail e.g. in [IvaSte88], [BerIva94] and section 3 of this work. Its learning algorithm was shown before, in Fig. 18. It consists in systematic search through the space of categorial rules, in the decreasing order of coverage; only those rules are accepted which "bring new information" wrt. the pseudobayesian composition of simpler rules accepted before. From the beginning, ESOD was viewed as a method for learning from data *without* expert. This was considered as a major advantage, as the method could be used even in domains where experts are not available, and on raw data rather than on selected examples only - the "Feigenbaum's bottleneck" (see section 1.1.2) could thus be avoided.

The motivation for using ESOD as the base-level learner and classifier was twofold[73]:

- the systematic search method of ESOD can be augmented to work with hierarchies of input attributes without any change to its principles;
- the compositional method of classification enables to quantify the plausibility of each alternative conclusion, which can be beneficial for sensitive evaluation of explanations[74].

To incorporate value hierarchies into the ESOD algorithm, two simple extensions are needed:

- to include into the CAT list also *abstract* values (categories), and
- to replace the notion of *subrule*[75] (rule whose left-hand side contains a subset of attribute-value pairs from the original rule) with the notion of *more general rule*, namely a rule with more general combination at the left-hand side: a combination C is more general (or equal) than D iff for every literal $A_i = v_i$ from C, there is a literal $A_i = v_j$ from D such that either $v_i = v_j$ or there is a path leading from v_i to v_j in the hierarchy defined on attribute A_i.

The way of handling class hierarchies in ESOD-based *learning* is obvious, as the macro-learner in Fig. 28 does not impose any requirements on the embedded learner; a set of

[72] The algorithms have been first described in [Svatek96].

[73] A third, less explicit reason might be the fact that ESOD has been developed at the author's workplace, the sum of unpublished know-how concerning its use being immediately available.

[74] Especially in the case of multiple explanations in a non-tree hierarchy, which has not been investigated yet.

[75] This is the reason why we have underlined it in the algorithm at Fig. 18.

weighted rules is induced for each non-leaf node of the hierarchy (hierarchical rulebase), as well as without recourse to the hierarchy (flat ruleset). The *classification* element of ESOD composes, separately for each class, the weights of rules which match the given instance, and returns the list of classes ordered according to composed weight. For testing purposes, we can assume that the class with highest weight - *best-weight class* - is always chosen[76]. This enables to integrate the ESOD classifier into the macro-classifier from Fig. 29.

The current version of the *hierarchization* algorithm is based on simple bottom-up hill-climbing clustering (similar to the discretization technique suggested by Lee and Shin [LeeShi94]), the objective function being the extrinsic-similarity-based cohesion introduced in section 5.2.4. At the beginning, each value of the target attribute corresponds to one cluster. In each step, a pair of clusters is joined so that the loss of overall cohesion is minimal. In the end, the resulting binary tree is heuristically pruned - some nodes for which the loss of cohesion was too small are removed. In order to process continuous values, a *discretization pre-processor* was also developed. It uses the same objective function as the hierarchization algorithm, but the search method is one-shot search for local maxima of the objective function (cf. e.g. [Kock95]). As the search techniques of hierarchization/discretization are likely to be replaced by more sophisticated ones in future versions, we do not describe them formally and in detail.

5.4.2 Integrity constraints and ESOD

The ESOD method is a generate-and-test method based on systematic search; with an increasing number of attributes and their values, the combinatorial complexity may become prohibitive. Essentially, there are two ways how to fight it:

- to impose *fixed limits* of search, such as the maximum number of left-hand-side literals or absolute/relative frequency of combinations to be considered,
- to introduce explicit restrictions on combinations to be investigated - *integrity constraints*, so as to eliminate impossible or uninteresting combinations.

The first group of techniques is widely used in machine learning; they rely upon the assumption of appropriate heuristics being available in a given domain. Their use is almost inevitable when applying ESOD-like methods on medium- and large-size datasets, as they enable a substantial cutoff of computational cost. They may, on the other hand, incur a serious loss of information present in data.

The second approach is much less reported, as it requires an external source of domain knowledge, which is not always available. In our work, we assume that in many domains,

[76]Note that when evaluating an explanation, we need not cling to the boolean concept of fit (1 if the next element in the explanation is returned as the best-weight class in the given step, 0 otherwise) but can take into account e.g. the weight difference between the best-weight class and the successor in explanation. The choice of explanation-evaluation schemes would deserve a thorough study, which has not been undertaken yet.

it is at least possible to point out *impossible combinations* of attribute values, which are ruled out by the fundamental logic of the domain.

For simplicity, let us consider only integrity constraints in the form:

$$a_i : (V_{i1}, \ldots, V_{im}) \Rightarrow a_j : (V_{j1}, \ldots, V_{jn})$$

or

$$a_i : (V_{i1}, \ldots, V_{im}) \Rightarrow \neg(a_j : (V_{j1}, \ldots, V_{jn}))$$

where a_i, a_j are attributes and $(V_{i1}, \ldots, V_{im})$, $(V_{j1}, \ldots, V_{jn})$ are subsets of their domains. They can be read as: if a_i has a value from $(V_{i1}, \ldots, V_{im})$[77] then a_j must (or must not, respectively) have a value from $(V_{j1}, \ldots, V_{jn})$.

The use of constraints in an ESOD-like algorithm is quite simple: when combinations are generated, those ruled out by the constraints are eliminated.[78] Integrity constraints may be particularly useful in conjunction with hierarchical attributes - it is possible to specify constraints at a higher level of abstraction, without recourse to concrete values present in data. Such constraints remain valid even if the concrete language of data changes.

It is obvious that in systematic-search-based methods like ESOD, most computer time is spent on computing the frequencies of combinations and goal concepts in data. There is, therefore, a tradeoff between evaluating combinations wrt. integrity constraints and wrt. data. We can deduce that:

- with increasing number of objects, the use of constraints saves more time spent on evaluation wrt. data, but
- at the same time, the number of combinations considered may rapidly grow and if only a small proportion of these are covered by constraints, the time spent on evaluation wrt. constraints increases.

The computation-time *cutoff* can be roughly expressed as

$$Coff = m_{IC} \,.\, n \,.\, t_o \;-\; m \,.\, t_{IC} \tag{8}$$

where

- m_{IC} is the number of generated combinations which are counterdicted by integrity constraints,
- n is the number of objects,

[77] Or a value which is subordinated to a value from $(V_{i1}, \ldots, V_{im})$, given a_i is a hierarchical attribute. The same holds for the right-hand side of the constraint.

[78] Presently, we do not consider the problem of mutual logical consistency of constraints, and left it to their provider.

- t_o is the time needed to verify whether an object satisfies a combination,
- m is the total number of generated combinations,
- t_{IC} is the time needed to verify whether a combination violates an integrity constraint.

This formula is naturally very simplified and does not account for many variable features present in the ESOD algorithm.

An important point is whether and when the number of combinations actually grows with increasing data size. Typically, we set a minimal frequency threshold for categories, which keeps the number of combinations very low for small data sizes; if we increase the number of objects, the number of combinations grows until all "common" combinations are present; since then it stagnates. The utility of constraints thus strongly depends on the size of the combinatorial space of attribute values - they will be most suitable for large datasets described by a low number of attributes with small domains. First experiments on using constraints are reported in section 5.5.1.

5.5 Experiments

The empirical experiments aimed at finding out about the impact of hierarchies and constraints on accuracy, comprehensibility and computation time. Three problems (and datasets) have served for this purpose so far: the control-site description problem formulated by the author of this work himself [Svatek96], the glass identification problem from the UCI repository of machine learning databases, and the problem of per-share revenue assessment from the Czechoslovak voucher privatization.

5.5.1 The control-site description (CSD) problem

An orienteering competition is a sport event, in which every competitor has to visit a sequence of control sites, in a given order; he uses a map and a compass to navigate in the terrain. Every control site is described by a set of pictograms which provide additional information. These are available long before the race (unlike the map which is obtained as late as at the start) and thus can be thoroughly analyzed. Each description consists of several items. The most important ones (which have only been considered for our purposes, as input attributes) describe:

- the object near which the control flag is situated (such as a hill, an outstanding tree or a stream)
- the position of the flag wrt. the object (such as on the top, at the south-west corner, or at the bend)
- additional property of the object (e.g. deep, sandy or deciduous)
- distinction of the object among several objects of the same type in the control area (e.g. southern, upper or middle).

Our goal attribute was the difficulty of finding the control flag.[79] Three classes have been heuristically suggested: 1 means no trouble finding the control flag, 2 means a small loss of time (up to a few tens of seconds), 3 means a greater loss of time, i.e. a serious navigation failure. It should be forestated that the difficulty of controls depends on many aspects not present in the descriptions, especially on other objects nearby, which can ease the navigation. In addition, the success of an individual competitor may be influenced negatively by his/her lowered concentration in the particular moment (even the easiest control can thus be missed), and positively by following another competitor. This is a reason why (noise-robust) methods for learning from observational data seem to be appropriate.

As a source data file, we have taken the protocol compiled by the author of this paper himself during 17 orienteering competitions across 3 years. The file contained 244 data objects corresponding to control sites. Of them, 186 were of class 1, 40 of class 2 and 18 of class 3. The observable values for (the four above mentioned) input attributes have been encoded according to standards of the International Orienteering Federation [ConDes90], and the attribute hierarchies have been constructed along rather obvious criteria[80]. In Fig. 31 we show two fragments of the hierarchy on attribute `obj_type`; each of them addresses a different way of abstraction. Class hierarachies have not been considered. Table 3 summarizes the numbers of observable and abstract values.

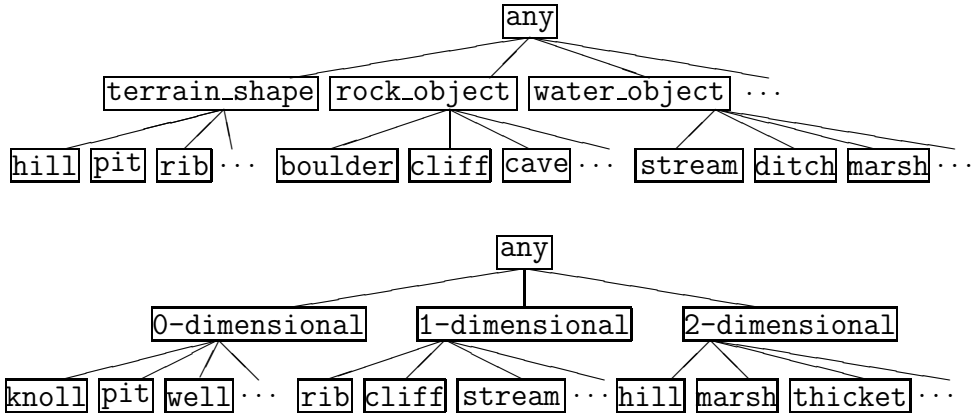

Figure 31: Fragments of the hierarchy on `obj_type` in the CSD task

[79] Recalling the categorization of *classification* problem solving tasks from section 1.2.3, we can see that our task can be most naturally viewed as *prediction*, as the properties of the control *cause*, with a certain likelihood, some entailments for the competitor, in the *future*.

[80] The knowledge we have used when building up the hierarchies can be, in the "scarcity" dimension from section 2.2.1, characterized as "common-sense", as it mostly deals with common names of terrain objects. Some terms and their relations (e.g. "thicket" or the difference between a hill and a knoll) may however not be clear to a non-initiated person; they can be viewed as "rudimentary" domain knowledge.

attribute	observable values	abstract values
obj_type	59	18
flag_pos	18	4
add_prop	11	2
which_of	6	3

Table 3: Numbers of attribute values in the CSD task

	with hierarchies	without hierarchies
number of rules	24	5
number of literals	42	2
accuracy in source data	76.64%	76.23%
average weight difference	0.107	0.135

Table 4: Results of experiments with hierarchies in the CSD task

We ran the learner with attribute hierarchies and without them, both times with a fixed frequency threshold (minimum) of 10 objects, to avoid overfitting. Then we ran the classifier with the rulesets induced, on the source dataset. The results are summarized in Table 2. In addition to accuracy obtained by choosing always the best-weight class, we have computed the average difference between the composed weight of the best-weight class and the composed weight of the actual class of the object; it can be viewed as a kind of "average error".

The rulebase learned without hierarchies contained almost only rules with empty left-hand side, as the frequency of most observable values was too low to allow generalization (due to the fixed frequency threshold as well as to the properties of the embedded statistical test). Its classification capability corresponded to *majority rule* approach, i.e. to assigning all objects to the most frequent class overall (which was class 1). The only "non-empty" rules induced were (weight in parentheses):

```
obj_type=boulder --> class = NOT 1 (0.75)
obj_type=boulder --> class = 3 (0.78)
```

which can be interpreted as "controls at boulders are rather difficult to find".

The rulebase learned with hierarchies contained substantially more rules but the accuracy did not significantly improve. This is not surprising because the data were extremely noisy due to external factors. However, most rules with abstract literals were meaningful and would definitely provide more information than rules with empty left-hand sides:

```
obj_type=1-dimensional --> class = 1 (0.61)
obj_type=0-dimensional --> class = NOT 1 (0.61)
```

```
obj_type=in_forest AND flag_pos=not_known --> class = NOT 1 (0.59)
obj_type=rock_object --> class = NOT 1 (0.71)
obj_type=man_made --> class = 1 (0.77)
obj_type=1-dimensional AND flag_pos=foot_of --> class = NOT 1 (0.77)
...
```

These rules are quite plausible: in the orienteering community, it is generally agreed that e.g. controls at linear or man-made objects are easy to find, unlike controls in rocky areas or at the foot of small ("0-dimensional") objects.

We have also experimented with integrity constraints. We have formulated the following ones:

```
obj_type:(0-dimensional) ==>
flag_pos:(unknown,top,side,edge,middle,foot)

obj_type:(1-dimensional) ==>
flag_pos:(unknown,slope,end,bend,crossing,fork,top,inner_corner,
          outer_corner,foot)

obj_type:(2-dimensional) ==>
flag_pos:(unknown,top,side,edge,middle,part,inner_corner,outer_corner,
          tip,foot)

obj_type:(vegetation) ==>
add_prop:(no,coniferous,deciduous,overgrown)
```

The first three claim that the dimension of the object (point, linear or planar) delimits the variety of possible positions of the control flag. The remaining one, analogically, lists possible properties of vegetation objects.

We have performed a series of tests, with data size growing[81] from 25 to 200, both with constraints and without them. The parameters were otherwise set as follows: frequency threshold (minimum) 5, combination length (maximum) 3, significance level 0.05. The results are summarized in Table 5. The columns of the table correspond, in turn, to the:

1. number of objects[82],
2. total number of combinations generated by the learner,
3. number of combinations eliminated by constraints (and thus not tested on data),

[81] For simplicity, the objects have been added in the natural order, not picked by random.
[82] We refer to the notation introduced in equation (8) in section 5.4.2.

Figure 32: Change of relative cutoff wrt. data size

4. computation time with constraints[83],
5. computation time without constraints,
6. difference between $t_{w/o}$ and t_{with} (the cutoff),
7. percentage of time cut off by constraints (i.e. $Coff/t_{w/o}$) - see also the graph at Fig. 32.

From the table we can see that the number of combinations (m) grows approximitely linearly with increasing data size (n), which, together with the increase of n itself, accounts for the increase of computation time. At the beginning, the constraints do not save any time (as there is no combination violating them). Subsequently, the constraints apply and eliminate impossible combinations such as "bend of a pit" or "flat tree". As the combinatorial space is rather large in the CSD task, we did not reach the stagnation point of m within acceptable data size; nevertheless, the utility of the constraints eventually seemed to (at least) outweigh their cost. The relative cutoff was constantly increasing (starting from negative value), with a single exception (at data size 150), which we could not explain so far[84]. The rapid increase of cutoff for largest data sizes should not be overestimated, as it may be implementation-dependent and related not only to the time complexity but also to the *space complexity* of the algorithm: it seems that, without constraints, the size of the database containing the combinations exceeds the limits of conventional memory, and time-consuming swapping is applied.

In general, the experiments above suggest that, in particular for tasks with combinatorial properties more favourable than in the CSD task, integrity constraints could lower

[83]The times are in the form hh:mm:ss. The high numbers are partly due to the slowness of the computer used for testing (PC386SX) and of the Prolog implementation.

[84]The preliminary hypothesis is that the newly accepted combinations were mostly rather specific, and thus "hard" to verify by constraints and "easy" to verify in data, in terms of position in the lattice of combinations.

n	m	m_{IC}	t_{with}	$t_{w/o}$	$Coff$	$Coff$ (%)
25	255	0	0:19:08	0:16:57	-0:02:11	-12.88
50	322	0	0:57:49	0:53:39	-0:04:10	-7.77
75	494	8	2:14:49	2:08:19	-0:06:30	-5.07
100	675	23	4:03:37	3:57:38	-0:05:59	-2.52
125	843	42	6:47:27	6:49:21	0:01:54	0.46
150	1001	65	9:55:07	9:55:59	0:00:52	0.15
175	1130	68	13:09:34	13:22:19	0:12:45	1.59
200	1402	107	20:10:04	21:27:51	1:17:47	6.04

Table 5: Results of experiments with integrity constraints in the CSD task

the computation time. In order to establish rules deciding whether to apply them in a particular situation, more thorough research would be needed.

5.5.2 Glass identification problem

The GLASS database is one of those recently added to the UCI repository of machine learning databases[85]; the original source was the British Home Office Forensic Science Service. The problem was formulated in the context of criminological investigation, which involved the need to identify the type of glass, found in the place of crime, based on optical and chemical analysis. The dataset has 214 instances described by values of 9 continuous-valued attributes indicating the refraction index and the content of various chemical elements. There are 7 classes of glass (of which only 6 are present in data); the documentation includes the hierarchy from Fig. 33, which has 5 non-leaf values.

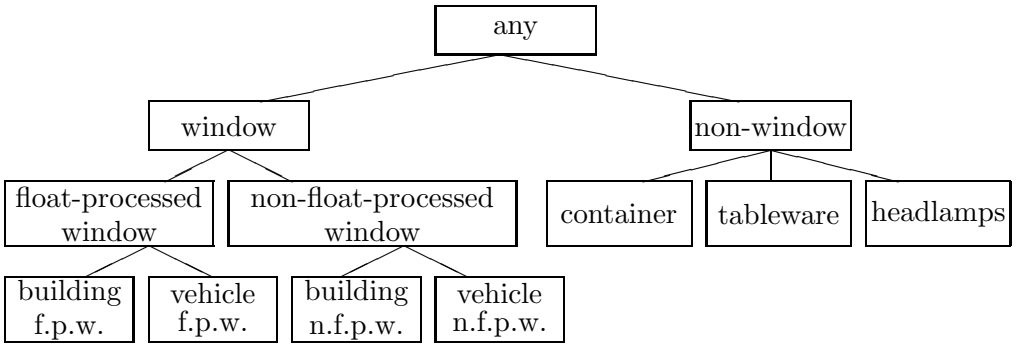

Figure 33: Class hierarchy in the glass domain

[85]The public repository at the University of California Irvine contains a large collection of datasets that have been used in ML research and applications and are constantly used for testing and comparison of new systems. See [UCI-rep95].

no.of rules of FR	93
avg.no.of rules per node in HR	25
acc.of FR in src.data	81.8%
acc.of HR in src.data	77.1%
perfect-fit explan.(PFE)	88.8%
acc.of FR / PFE	84.2%
acc.of FR / ¬PFE	62.5%

Table 6: Summary of results from the glass domain

We have first discretized and then hierarchized[86] the input values with respect to class distributions, by means of techniques presented in section 5.2.4; as class hierarchy, we took the one present in documentation, as it was. Then we induced both the flat ruleset (FR) and the hierarchical rulebase (HR), and tested them on source data. The results are summarized in Table 6.

We can see that the average number of rules in partial rulesets of the hierarchical rulebase was significantly lower than the number of rules of the flat ruleset. We can thus assume that individual refinement steps within the hierarchy could be more transparent than the classification made merely with the flat ruleset. The hierarchical rulebase had an acceptable performance by itself; moreover, it provided a high number of *perfect-fit explanations* to the classification decisions produced by the flat ruleset. An important observation (see the last two lines in the table) was that the accuracy of classification was substantially higher for decisions with perfect-fit explanations than for the others. This seems to suggest that the lack of explanatory justification signals a lower degree of *confidence* in the conclusion.

To illustrate the outlook of a hierarchical explanation, we present the trace for instance no.1 from the source dataset (Fig. 34). First, the input values are listed. Then, the classification by the flat ruleset is described, in terms of rules which decided about the class assignment.[87] Finally, each individual step of the explanation is listed in the same way. We can see that the explanation decomposes the process of classification: the content of magnesium decides between window/non-window glass, while the content of silicium indicates that the glass has been float-processed; finally, as none of the "exception" rules for vehicle glass is fired, it is assumed by default that the instance belongs to class 1 (float-processed building window glass).

[86]For the sake of brevity, we list only the results achieved with input value hierarchies. For comparison, we have also proceeded with non-hierarchized values; the results were slightly but insignificantly worse in terms of accuracy.

[87]Not all rules applied are listed but only those satisfying a heuristic criterion of display; the problem of selection of rules to be *displayed* in explanation is omitted here, as it is rather specific for the compositional paradigm of ESOD.

```
OBJECT:
   RI=1.52101, Na=13.64, Mg=4.49, Al=1.10,
   Si=71.78, K=0.06, Ca=8.75, Ba=0.00, Fe=0.00.

CLASSIFICATION:
   1 - Building windows float processed glass
      default: 1 (0.33)
      rule: Mg > 2.695 --> 1 (0.63)
      rule: Si from 71.3 to 71.785 --> 1 (0.86)
      composed: 1 (0.93) - best weight by 0.34

HIERARCHICAL EXPLANATION:
   FIT 3 of 3
   LEVEL 1: choosing between
         win - Window glass
         nonwin - Non-window glass
      default: win (0.76)
      rule: Mg > 2.695 --> win (0.94)
      composed: win (0.99) - best weight by 0.98
   LEVEL 2: choosing between
         fp - Float-processed window glass
         nfp - Non-float-processed window glass
      default: fp (0.53)
      rule: Si from 71.3 to 71.785 --> fp (0.91)
      composed: fp (0.94) - best weight by 0.88
   LEVEL 3: choosing between
         1 - Building windows float processed glass
         3 - Vehicle windows float processed glass
      default: 1 (0.80)
      composed: 1 (0.80) - best weight by 0.60
```

Figure 34: Example of hierarchical explanation in the glass domain (object no.1)

5.5.3 Per-share revenue problem

The third experimental dataset, eventually, was an *economic* one. During the last years, the domain of economics and business is becoming more and more fruitful for ML applications. As a fairly distant goal, we can view the suggestion by Jaime Carbonell, made in 1992: "I propose several different challenges that could be within reach of present-day researchers from different ML paradigms, borrowing liberally from discussions with my colleagues: ..(parts left out).. ML IN FINANCE. The first machine learning program that learns in some aspect of investment (e.g. the equities market), producing an investment strategy that earned the investors *over one million dollars in a one-year period* when the strategy was followed to the letter without human intervention..." [Carbon92]. Our experiments were, naturally, much more humble.

In the Czechoslovak voucher privatization (1992-4), the stock of large companies previously owned by the state has been distributed to masses of shareholders by means of a "roulette-like" investment game. For the investors it was important to assess, in advance, the future per-share revenue. The dataset we used in our experiments is a pre-processed version of information provided by the investors by a public-information company; the pre-processing, which consisted in discretization of numeric values in dialog with expert, has been done within a master-thesis project [Kovar95]. The dataset contains 581 objects (companies) described by 18 input attributes and a class attribute with 8 ordered values - the actual per-share revenue. Among the input attributes, two were binary, two were multivalued nominal (region and industry), and the remaining ones were discretized ordered (various economic indicators). After the removal of incompletely specified objects[88], 509 entered the actual experimentation.

First, we have hierarchized all but the binary input attributes, based on the class attribute. Despite the simplicity of the hill-climbing clustering algorithm, some meaningful value unions arose, which were interpretable even without recourse to an expert. For the "region" attribute, for example, the top split of the hierarchy separated the eight Czech regions from the three Slovak ones (it should be noted that the dataset was collected before the political splitting of Czechoslovakia!); the next split divided the Czech regions into sets of five and three, the latter being exactly the western regions which neighbour with EC countries (Germany and/or Austria). The class attribute has been also submitted to hierarchization, which yielded a simple partition into the upper three and the lower five values.

The rule learning tasks were performed in the same way as in the glass domain. The results are in Table 7. As the exact correspondence of real/proposed class may be too strict a requirement in the case of eight ordered classes, we have also computed "tolerant-matching" accuracy, such that each object was considered as correctly classified even if the proposed class was the one immediately above/below the real class. The results are similar as for the glass domain, except that the hierarchical rulebase achieves higher accuracy that the flat ruleset. This may be attributed to the high number and thus low

[88]Techniques for handling unknown values in ESOD learning exist but they have not been incorporated into the hierarchical version yet.

no.of rules of FR	97	
avg.no.of rules per node in HR	38.67	
Testing	*strict*	*"tolerant"*
acc.of FR in src.data	35.8%	68.8%
acc.of HR in src.data	43.0%	77.8%
perfect-fit explan.(PFE)	58.7%	———
acc.of FR / PFE	47.5%	78.9%
acc.of FR / ¬PFE	19.0%	54.3%

Table 7: Summary of results from the privatization domain

frequency of leaf classes, which prevents some correct rules from being accepted in flat learning, due to insufficient coverage; rules at the *any* node, deciding between two large class unions only, are more likely to pass. This phenomenon definitely deserves a deeper study.

5.5.4 Summary interpretation

Based on the preliminary experiments described in this section, we have formulated several hypotheses, which should be verified by more thorough testing:

- abstraction hierarchies on input attributes can improve comprehensibility of rules learned; the impact on accuracy is not significant (at least, not in source data);
- hierarchies of classes can improve comprehensibility of classification, via providing reconstructive explanations; they can be also used to (roughly) estimate the confidence in the classification decision; under certain circumstances, hierarchical classification can be even more accurate than classification without class hierarchy;
- class-sensitive clustering can discover interesting unions of attribute values; these unions should be subject to post-interpretation in order to bring benefits in terms of rule comprehensibility;
- integrity constraints can lower the computation time if they cover each a significant amount of the combinatorial space, and the number of objects is high enough to make up for the overhead of constraints testing.

5.6 Problems and perspectives of learning with hierarchies and constraints

We have described a set of implemented methods for dealing with value hierarchies within the learning process, both for input attributes and for the class attribute. The main underlying motivation is gaining better comprehensibility, which seems to have been achieved

to a certain extent: abstract values of input attributes improve comprehensibility of *individual rules*, while the structure of classes gives a deeper insight into the *classification process*. We also study the related task of *hierarchy building*, and suggest an objective function for clustering with hierarchical and/or linear source attributes. Furthermore, integrity constraints can decrease the learning time, under specific circumstances related to the combinatorial complexity of the attribute-value search space.

Much work should be done to *verify* the hypotheses formulated so far more thoroughly, and to *increase* the power of the methods. The former will require to complement source-data testing with *cross-validation*, which is one of the central points of the original ESOD method, but has not been considered in our version yet; also, learners and classifiers based on other than compositional paradigms should be embedded into the hierarchical macro-algorithms.

For the methods themselves, multiple extensions are at hand, of which some are to be realized in the near future:

- comparing multiple explanations (in non-tree dags of classes);
- handling non-leaf (i.e. partial "no-care") values in data[89];
- replacing hill-climbing in the hierarchization algorithm with a more sophisticated search technique (cf. [Fisher96]).

[89]This phenomenon is natural for *case libraries*, where the expert's decision is often made based on higher-level input values (cf. e.g. [Lenz95]).

6 Conclusions

In this work, we have analyzed the problem area of *learning from data with prior knowledge*, from multiple viewpoints. Following an investigation of the state-of-the-art (which yielded a brief overview, in section 2), we have concentrated on the problems of:

- learning with prior *problem-solving* knowledge (in the form of rules with/without weight);
- learning with prior *static domain knowledge* (in the form of abstraction hierarchies and integrity constraints).

For each of these, novel techniques have been devised, which solve the problem to a certain extent. Nevertheless, open problems remain which would require more thorough work. Some of the techniques are not general enough yet to be applicable to real-world problems. In this section, we summarize the outcomes of the subprojects and outline the perspectives of each approach.

The research on learning with prior decision rules *without weight* has resulted in formulation of a knowledge-integration scheme (section 4.3), which takes into account the correctness as well as the importance of individual rules; the integrated ruleset is, in a certain sense, minimal and consistent. It can be applied straightforwardly in a propositional framework, where the generality relation can be determined easily; in a richer (e.g. first-order) language, the application would be much more difficult. Moreover, the technique addresses only the *integration* segment of the *"bypass" model* (cf. section 4.2); attention should be paid to the *induction* segment as well, as the utility of application depends on the synergy of techniques realizing both segments.

The research on integration of *weighted* rules from expert and from data (section 4.4) has opened many problems. We have suggested a simple heuristic technique for interpolation of expert's and empirical weights of a rule; however, it is not clear how this can be adapted to a set of multiple interrelated rules. The expected difficulties related to the complexity of consistent weight adjustment have proven real, and led to a partial failure of this subproject, especially with respect to the implementation within the existing ESOD method. Some preliminary ideas have, however, been formulated (in section 4.4.4), which could give rise to more successful solutions in the future (e.g. use of qualitative weights). This would of course require a substantial amount of further research, both in the field of theory and that of experimental endeavour.

In the context of learning with prior *static domain knowledge*, we have suggested and implemented a set of methods for dealing with value hierarchies and integrity constraints within the learning process. The hierarchies can be defined on the domains of both the input attributes and the class attribute.

The main underlying motivation was to gain better *comprehensibility*, which seems to have been achieved to a certain extent. First experiments suggest that abstract values of input attributes improve the comprehensibility of *individual rules*, as they are closer

to human thinking (they alleviate the abstraction step which would be necessary for interpretation of "data-level" rules with detailed values). Potentially, such rulebases can be also more *concise*, as a single rule with abstract values can replace multiple rules with detailed values. The structure of classes, on the other hand, gives a deeper insight into the *classification process*, thanks to hierarchical explanation.

Moreover, abstract values of input attributes have higher *coverage* than detailed values, and thus are more likely to satisfy sample-size biases of a given learning method. Somewhat analogically, an impact of class hierarchies can be that of increased *reliability* of the classification results, as hierarchical classification complements direct classification; if there is an agreement, the result is (more) reliable, otherwise it is (more) dubious. In some situations, hierarchical classification can "focus" on the right part of the classification domain an thus provide, by itself, better accuracy than direct classification.

Furthermore, we have studied the related task of *hierarchy building*, and suggested an objective function for clustering with hierarchical and/or linear source attributes (section 5.2.4). The generic part of the function introduces sensitivity to *extrinsic dissimilarity* (i.e. dissimilarity among different values of the same source attribute), which seems to be a novel idea in the machine learning research.

The impact of *integrity constraints* (in terms of computation time cutoff) depends on the proportion of combinations covered by them and on the number of objects (section 5.4.2). As soon as the data objects saturate the combinatorial space, further increase of data should rapidly increase the cutoff. However, in one of our experiments, the utility of constraints has outweighted their cost after a certain sample size has been reached, even though the number of combinations grew approximately linearly.

The present thesis has addressed several problems considered as important by machine learning research. It has attempted to categorize and characterize some (most prominent) existing approaches to learning with prior knowledge, and to contribute to the research in this field by novel methods and techniques. These methods have been, at least partially, evaluated in a theoretical as well as experimental manner.

The author envisages to elaborate further on (at least some of) the outcomes of the work in his next project(s), giving priority to application-ripen methods, especially in connection with analysis of economic data.

æ

References

[AamPla94] Aamodt, A. - Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, Vol.7, Nr.1, 1994, 29-38.

[AlmAkK95] Almuallim, H. - Akiba, Y. A. - Kaneda, S.: On Handling Tree-Structured Attributes in Decision Tree Learning. In: Proceedings of the Twelfth International Conference on Machine Learning (ML-95). Morgan Kaufmann, 12-20.

[AroPrB96] Aronis, J. M. - Provost, F. J. - Buchanan, B. G.: Exploiting Background Knowledge in Automated Discovery. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, 1996 (KDD-96).

[Berka93a] Berka, P: Knowledge EXplorer: a Tool for Automated Knowledge Acquisition from Data. Austrian Research Institute for AI, Vienna, TR-93-3, 1993.

[Berka93b] Berka, P.: A Comparison of Three Different Methods for Acquiring Knowledge about Virological Hepatitis Tests. Austrian Research Institute for AI, Vienna, TR-93-10, 1993.

[Berka93c] Berka, P.: Discretization of Numerical Attributes for Knowledge Explorer. LISp Technical Report 93-03, Prague University of Economics, Prague 1993.

[BerIva94] Berka, P. - Ivánek, J.: Automated Knowledge Acquisition for PROSPECTOR-like Expert Systems. In: ECML'94, European Conference on Machine Learning. Lecture Notes on Artificial Intelligence, Springer Verlag 1994, 339-342.

[Berka95] Berka, P.: Knowledge Explorer. Habilitation thesis. Prague University of Economics, Faculty of Informatics and Statistics, 1995.

[Birnb91] Birnbaum, L.: Rigor mortis: a response to Nilsson's "Logic and artificial intelligence". *Artificial Intelligence*, 47 (1991), 57-77.

[BluEhH87] Blumer, A. - Ehrenfeucht, A. - Haussler, D. - Warmuth, M. K.: Occam's Razor. *Information Processing Letters*, 24 (1987), 377-380.

[BraCeK95] Bratko, I. - Cestnik, I. - Kononenko, I.: Attribute-Based Learning. In: State of the Art in Machine Learning. Special Issue of *MLnet News, the Newsletter of the European Network of Excellence in Machine Learning*, September 1995, 6-11.

[BraJor93] Brazdil, P. - Jorge, A.: Exploiting Algorithm Sketches in ILP. In: ILP'93 - Third International Workshop on Inductive Logic Programming, Bled 1993.

[BraTor90] Brazdil, P. - Torgo, L.: Knowledge Acquisition via Knowledge Integration. In: Current trends in Knowledge Acquisition. IOS Press, 1990.

[BroUtt95] Brodley, C. E. - Utgoff, P. E.: Multivariate Decision Trees. *Machine Learning*, Vol.19, No.1, 1995, 45-77.

[Bruha91] Bruha, I.: Machine Learning: Empirical Methods. In: SOFSEM'91 - 18th Intl. Winter School on Current Trends in Theory and Practice of Informatics, Jasná p. Chopkom 1991, 7-51.

[Bruha95] Bruha, I.: Unknown Attribute Value Processing Utilizing Expert Knowledge on Attribute Hierarchy. In: MLnet Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases, Heraklion 1995, 130-135.

[Bry93] Bry, F.: Towards Intelligent Databases. In: ISMIS'93 - Seventh Intl. Symposium on Methodologies for Intelligent Systems, Lecture Notes on Artificial Intelligence, Springer Verlag 1993, 116-131.

[Cain91] Cain, T.: The DUCTOR: A Theory Revision System for Propositional Domains. In: Proceedings of the Eighth National Conference on Machine Learning, 1991, 485-489.

[Camac95] Camacho, R.: Using Machine Learning to extract Models of human control skills. In: AIT'95 - The 2nd International Workshop on Artificial Intelligence Techniques, Brno 1995, 143-160.

[Carbon92] Carbonell, J.: Machine Learning: A Maturing Field. Machine Learning 9 (1992), 5-8.

[Carry94] Carry, C.: Validation de connaissances apprises. In: JFA-94, Neuvièmes Journées francophones sur l'Apprentissage, Strasbourg 1994, D1-D15.

[Catl91] Catlett, J.: On changing continuous attributes into ordered discrete attributes. In: Machine Learning - EWSL'91, European Working Session on Learning, Springer Verlag 1991, 164-178.

[Clanc85] Clancey, W. J.: Heuristic Classification. *Artificial Intelligence*, 27 - 3, 1985, 289-350.

[Clanc88] Clancey, W. J.: Acquiring, representing, and evaluating a competence model of diagnostic strategy. In: The Nature of Expertise, Lawrence Erlbaum Press 1988, 343-418.

[ConDes90] Control descriptions. IOF 1990.

[ClaNib89] Clark, P. - Niblett, T.: The CN2 induction algorithm. Machine Learning, 3, 261-283, 1989.

[ClaMat93] Clark, P. - Matwin, S.: Using Qualitative Models to Guide Inductive Learning. In P. Utgoff, editor, Proc. Tenth International Machine Learning Conference (ML-93), 49-56, 1993.

[ClaMat94] Clark, P. - Matwin, S.: Using Qualitative Models to Guide Inductive Learning. In: Machine Learning - ECML'94, European Conference on Machine Learning, Catania 1994. Lecture Notes on Artificial Intelligence, Springer Verlag 1994, 360-365.

[ClePro90] Clearwater, S. - Provost, F. J.: RL4: A Tool for Knowledge-Based Induction. In: Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence, IEEE C.S. Press, 24-23.

[CraSle90] Craw, S. - Sleeman, D.: Automating the Refinement of Knowledge-Based Systems. In: ECAI'90 - European Conference on Artificial Intelligence, Stockholm 1990, 167-172.

[CraSle91] Craw, S. - Sleeman, D.: The Flexibility of Speculative Refinement. In: Proceedings of the Eighth National Conference on Machine Learning, 1991, 28-32.

[CraSlB94] Craw, S. - Sleeman, D. - Boswell, R. - Carbonara, L.: Is Knowledge Refinement Different from Theory Revision?. In: MLnet Workshop on Theory Revision and Restructuring. Catania 1994, 33-35.

[DecMer94] Decaestecker, C. - van de Merckt, T.: A Unifying Framework for Analysing Biases in Similarity-Based Learning.In: MLNet Workshop on Declarative Bias, Catania 1994, 5-33.

[DeJMoo86] DeJong, G. - Mooney, R.: Explanation-Based Learning: An Alternative View. *Machine Learning*, 1 (1986), 145-176.

[DeHMaW94] De Hoog, R. - Martil, R. - Wielinga, B. - Taylor, R. - Bright, C. - van de Velde, W.: The Common KADS model set. Delivrable KADS-II/M1/DM1.1b/UvA/018/6.0/FINAL

[DeRLaD93] De Raedt, L. - Lavrač, N. - Džeroski, S.: Multiple Predicate Learning. ILP Intl. Workshop, Bled 1993, 221-240.

[DudGas79] Duda, R. O. - Gasching, J. E.: Model Design in the Prospector Consultant System for Mineral Exploration. In: Michie, D. (ed.), Expert Systems in the Micro Electronic Age, Edinburg University Press.

[Duval91] Duval, B.: Abduction and Induction for Explanation-Based Learning. In: Machine Learning-EWSL'91, Porto 1991, 348-360.

[Fensel94] Fensel, D.: What it makes difficult to apply Machine Learning in Model-Based Knowledge Acquisition. In: MLNet Workshop on Learning and Knowledge Level, Catania 1994, 1-4.

[Fensel93] Fensel, D.: The Reconciliation of Symbol and Knowledge Level. Technical Report, Angewandte Informatik und Formale Beschreibungsverfahren, University of Karlsruhe (Nr. 266, 1993.)

[FenHar94] Fensel, D. - van Harmelen, F.: A comparison of languages which operationalize and formalize KADS models of expertise. The Knowledge Engineering Review, Vol.9:2, 1994, 105-146.

[FerCoJ93] Ferreira, J. - Correia, J. - Jamet, T. - Costa, E.: An Application of Machine Learning in the Domain of Loan Analysis. In: Machine Learning - ECML'93, European Conference on Machine Learning, Vienna 1993. Lecture Notes on Artificial Intelligence, Springer Verlag 1993, 414-419.

[Fisher87] Fisher, D. H.: Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning* 2, 139-172.

[Fisher96] Fisher, D. H.: Iterative Optimization and Simplification of Hierarchical Clusterings. *Journal of Artificial Intelligence Research* 4 (1996), 147-179.

[Flach93] Flach, P.: Predicate invention in inductive data engineering. In: Machine Learning - ECML'93, European Conference on Machine Learning, Vienna 1993. Lecture Notes on Artificial Intelligence, Springer Verlag 1993, 83-94.

[GanThL93] Ganascia, J. G. - Thomas, J. - Laublet, P.: Integrating Models of Knowledge and Machine Learning. In: Machine Learning - ECML'93, European Conference on Machine Learning, Vienna 1993. Lecture Notes on Artificial Intelligence, Springer Verlag 1993, 396-401.

[Garden88] Gärdenfors, P.: Knowledge in Flux - Modeling the Dynamics of Epistemic States. MIT Press, Cambridge, MA, 1988.

[GenLaF89] Gennari, J. - Langley, P. - Fisher, D. H.: Model of Incremental Concept Formation. Artificial Intelligence Journal, Vol.40, 11-61.

[GinWeP88] Ginsberg, A. - Weiss, S. M. - Politakis, P.: Automatic Knowledge Base Refinement for Classification Systems. *Artificial Intelligence*, 35 (1988), 197-226.

[GorDeJ95] Gordon, D. - desJardins, M.: Evaluation and Selection of Biases in Machine Learning. *Machine Learning*, 20, 5-22 (1995).

[HajHav85] Hájek, P. - Havránek, T.: Automatická tvorba hypotéz: mezi analýzou dat a umělou inteligencí. [Automated Hypotheses Formation. Between Data Analysis and Artificial Intelligence.] In: Metody umělé inteligence a expertní systémy II. ČSVTS - FEL ČVUT, Praha 1985, 84-98.

[HajHaC83] Hájek, P. - Havránek, T. - Chytil, K. M.: Metoda GUHA. Automatická tvorba hypotéz. Praha, Academia 1983. 316 pp.

[HamHiC96] Hamilton, H. J. - Hilderman, R. J. - Cercone, N.: Attribute-Oriented Induction Using Domain Generalization Graphs. In: Proceedings of the Eighth IEEE International Conference on Tools with Artificial Intelligence, IEEE C.S. Press, Toulouse 1996, 246-253.

[Haussl88] Haussler, D.: Quantifying Inductive Bias: AI Learning Algorithms and Valiant's Learning Framework. Artificial Intelligence, 36 (1988), 177-221.

[HelSom95] Helsper, E. M. - van Someren, M. W.: Interactive Refinement for Structured Knowledge-Based Systems. In: MLnet Workshop on Knowledge Level Modelling and Machine Learning, Heraklion 1995, I.3.1-I.3.13.

[Herm94] Herrmann, J.: Multiple Biases that Control the Different Learning Strategies in COSIMA. In: MLNet Workshop on Declarative Bias, Catania 1994, 41-44.

[IvaSte88] Ivánek, J. - Stejskal, B.: Automatic Acquisition of Knowledge Base from Data without Expert: ESOD. In: Compstat'88, Physica-Verlag, Heidelberg 1988, 175-180.

[Ivanek94] Ivánek, J.: Minimum knowledge base search from categorial data. In: 3^{rd} Workshop on Uncertainty Processing in Expert Systems, Třešť 1994, 77-86.

[KADS95] KADS-II Project, state in Dec95. Internet, WWW, `http://swi.psy.uva.nl/projects/CommonKADS/home.html`.

[KAW95] KAW Mailing List Archive, Internet, `ftp://swi.psy.uva.nl/pub/mailing-lists/kaw/`.

[KliMoR95] Klingspor, V. - Morik, K. - Rieger, A.: Learning Concepts from Sensor Data of a Mobile Robot, *Machine Learning*, 1995.

[Kock95] Kočková, S.: Evaluation Functions in the Multi-Interval Discretization Context. In: MLnet Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases, Heraklion 1995, 222-227.

[Kodrat94] Kodratoff, Y.: Industrial Applications of ML: Illustrations for the KAML Dilemma and the CBR Dream. In: Machine Learning - ECML'94, European Conference on Machine Learning, Catania 1994. Lecture Notes on Artificial Intelligence, Springer Verlag 1994, 3-19.

[KodMic90] Kodratoff, Y. - Michalski, R. (eds.): Machine Learning: An Artificial Intelligence Approach. Vol.III, Morgan Kaufmann 1990.

[Kolodn93] Kolodner,J.: Case-Based Reasoning. Morgan Kaufmann, 1993, 668 pp.

[Kovar95] Kovaříčková: Automatic knowledge acquisition from economic data. [Master thesis, in Czech], Prague University of Economics, 1995.

[KrePoS95] Kretowski, M. - Polkowski, L. - Skowron, A. - Stepaniuk, J.: Data Reduction Based on Rough Set Theory. In: MLnet Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases, Heraklion 1995, 210-215.

[Kubat89] Kubát, M.: Floating approximation in time-varying knowledge-base. In: *Pattern Recognition Letters* (Vol.10), 1989, 223-227.

[KukPop94] Kuklová, J. - Popelínský, L.: On Biases in Inductive Data Engineering. In: MLNet Workshop on Declarative Bias, Catania 1994, 45-51.

[LaiRoN86] Laird, J. E. - Rosenbloom, P. S. - Newell, A.: Chunking in Soar: The Anatomy of a General Learning Mechanism. *Machine Learning*, 1:11-46, 1986.

[LinVal91] Ling, X. - Valtorta, M.: Revision of Reduced Theories. Proceedings of the Eighth National Conference on Machine Learning, 1991, 519-523.

[DecBia94] MLNet workshop on Declarative Bias. Working Notes. Catania 1994.

[LavDze94] Lavrač, N. - Džeroski, S.: Inductive Logic Programming. Ellis Horwood 1994, 293 pp.

[LeeShi94] Lee, C. - Shin, D. G.: A Context-Sensitive Discretization of Numeric Attributes for Classification Learning. In: European Conference on Artificial Intelligence, ECAI-94, 428-432.

[LenBuc91] Leng, B. - Buchanan, B. G.: Constructive Induction on Symbolic Features: Introducing New Comparative Terms. In: Proceedings of the Eighth National Conference on Machine Learning, 1991, 163-167.

[Lenz95] Lenz, M.: CABATA - A hybrid CBR system. In: Proc.First European Workshop on Case-Based Reasoning (EWCBR-93), Kaiserslautern 1993.

[LeRoux94] Le Roux, B.: De l'Expertise de Modélisation. In: JAC-94, Cinquièmes Journées Acquisition des Connaissances, Strasbourg 1994, A1-A14.

[UCI-rep95] Machine Learning Databases' Repository, Internet, WWW, `http://www.ics.uci.edu/AI/ML/Machine-Learning`.

[MarVlc92] Mařík, V. - Vlček, T.: Some Aspects of Knowledge Engineering. In: Advanced Topics in AI. Lecture Notes in AI 617, Springer 1992, 316-337.

[Math91] Matheus, C. J.: The Need for Constructive Induction. In: Proceedings of the Eighth National Conference on Machine Learning, 1991, 173-177.

[MicCaM83] Michalski, R. S. - Carbonell, J. G. - Mitchell, T. M. (Eds): Machine Learning. An Artificial Intelligence Approach. Palo Alto, Tioga Press 1983. 572 pp.

[Michal83] Michalski, R. S.: A Theory and Methodology of Inductive Learning. *Artificial Intelligence*, 20 (1983), 111-116.

[MicCaM86] Michalski, R. - Carbonell, J. - Mitchell, T. (eds.): Machine Learning: An Artificial Intelligence Approach, Vol.II. Morgan Kaufmann 1986.

[MicKod86] Michalski, R. S. - Kodratoff, Y.: Research in Machine Learning: Recent Progress, Classification of Methods, and Future Directions. In:[KodMic90], 3-29.

[Mitch77] Mitchell, T. M.: Version spaces: a candidate elimination approach to rule learning. In: Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77), 1997, 305-310.

[Mitch82] Mitchell, T. M.: Generalization as search. *Artificial Intelligence*, 18, 203-226, 1982.

[MitKeK86] Mitchell, T. M. - Keller, R. M. - Kedar-Cabelli, S. T.: Explanation-Based Generalization: A Unifying View. Machine Learning, 1 (1986), 47-80.

[MooOu91a] Mooney, R. J. - Ourston, D.: Theory Refinement with Noisy Data. Tech.Rep.AI91-153, AI Lab, Univ. of Texas at Austin, March 1991.

[MooOu91b] Mooney, R. - Ourston, D.: Constructive Induction in Theory Refinement. In: Proceedings of the Eighth National Conference on Machine Learning, 1991, 178-182.

[MooOur93] Mooney, R. - Ourston, D.: A Multistrategy Approach to Theory Refinement. In: Michalski, Tecuci (eds.): Machine Learning: A Multistrategy Approach. Morgan Kaufman 1993, 141-164.

[MorWrK93] Morik, K. - Wrobel, S. - Kietz, J. U. - Emde, W.: Knowledge Acquisition and Machine Learning: Theory, Methods and Applications. Academic Press, London, New York, 1993.

[MugFen90] Muggleton, S. - Feng, C.: Efficient induction of logic programs. In: Proceedings of the Workshop on Algorithmic Learning Theory, 1990.

[MurKaS93] Murthy, S. - Kasif, S. - Salzberg, S. - Beigel, R.: OC1: Randomized induction of oblique decision trees. In: Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI'93), 322-327.

[Nakh95] Nakhaeizadeh, G.: Is KDD something more than the classical data analysis? In: AIT'95 - The 2nd International Workshop on Artificial Intelligence Techniques, Brno 1995, 69-72.

[NedCau92] Nédellec, C. - Causse, K.: Knowledge Refinement using Knowledge Acquisition and Machine Learning methods. In: Current developments in Knowledge Acquisition: EKAW'92, 171-190.

[Nedel95] Nédellec, C.: Integration of Machine Learning and Knowledge Acquisition. *The Knowledge Engineering Review*, Vol.10:1, 1995, 77-81.

[Nedel96] Nédellec, C.: Modeling ML and Comprehensibility. In: IJCAI'95 Workshop on Comprehensibility in Machine Learning, Montreal 1995.

[NedRou94] Nédellec, C. - Rouveirol, C.: Specifications of the HAIKU system. Rapport de Recherche no.928, L.R.I., Université de Paris Sud 1994.

[Newel82] Newell, A.: The knowledge level. *Artificial Intelligence* 18 (1982), 87-127.

[Nilss91] Nilsson, N. J.: Logic and artificial intelligence. *Artificial Intelligence*, 47 (1991), 31-56.

[Nunez91] Núñez, M.: The Use of Background Knowledge in Decision Tree Induction. *Machine Learning*, 6, 231-250 (1991).

[OliBaW96] Oliver, J. J. - Baxter, R. A. - Wallace, C. S.: Unsupervised Learning Using MML. In: Proceedings of the Thirteenth International Conference on Machine Learning (ML-96), Morgan Kaufmann, 364-372.

[Pfahr94] Pfahringer, B.: Controlling Constructive Induction in CIPF: An MDL Approach. In: Machine Learning - ECML'94, European Conference on Machine Learning, Catania 1994. Lecture Notes on Artificial Intelligence, Springer Verlag 1994, 242-256.

[PosScZ96] Posthoff, C. - Schlosser, M. - Zeidler, J.: The Integration of Rule and Example Sets. In: FGML-96, Annual Meeting of the Special Interest Group Machine Learning, Chemnitz 1996.

[Przym88] Przymusinski, T. C.: On the semantics of stratified deductive databases. In: J.Minker (ed.): Foundations of Deductive Databases and Logic Programming. Morgan Kaufmann, Los Altos 1988, 193-216.

[Quinl86] Quinlan, J. R.: Induction of Decision Trees. *Machine Learning* 1: 81-106, 1986.

[Quinl87] Quinlan, J. R.: Simplifying decision trees. *Int. J. Man-Machine Studies* (1987), No.27, 221-234.

[Quinl93] Quinlan, J. R.: C4.5 Programs for Machine Learning. San Mateo, Morgan Kaufmann 1993.

[QuiCam93] Quinlan, J. R. - Cameron-Jones, R. M.: FOIL: A Midterm Report. In: Machine Learning - ECML'93, European Conference on Machine Learning, Vienna 1993. Lecture Notes on Artificial Intelligence, Springer Verlag 1993, 3-20.

[Quinl94] Quinlan, J. R.: The Minimum Description Length Principle and Categorical Theories. In: Proceedings AAAI'94, 233-241.

[RenRag93] Rendell, L. - Ragavan, H.: Improving the Design of Induction Methods by Analyzing Algorithm Functionality and Data-Based Concept Complexity. Proc. IJCAI'93, Chambéry 1993, 952-958.

[RicMoo95] Richards, B. L. - Mooney, R. J.: Automated refinement of first-order Horn-clause domain theories. *Machine Learning*, Vol.19, No.2, 95-131, 1995.

[RouAlb94] Rouveirol, C. - Albert, P.: Knowledge Level Modelling of Generate and Test Learning Systems. In: MLNet Workshop on Learning and Knowledge Level, Catania 1994.

[Saitta95] Saitta, L.: Shift of Bias in Machine Learning. In: State of the Art in Machine Learning. Special Issue of MLnet News, the Newsletter of the European Network of Excellence in Machine Learning, September 1995, 38-44.

[SchGra84] Schlimmer, J. C. - Granger Jr., R. H.: Incremental Learning from Noisy Data. Machine Learning, vol.1, no.3, 1984, 317-354.

[ScmAit95] Schmalhofer, F. - Aitken, J. S.: Beyond the Knowledge Level: Behavior Descriptions of Machine Learning Systems. MLnet Workshop on Knowledge Level Modelling and Machine Learning, Heraklion 1995, III.1.1.-III.1.15.

[Schr95] Schreiber, T.: Knowledge-level modelling and the new knowledge engineering cycle. In: MLnet Workshop on Knowledge Level Modelling and Machine Learning, Heraklion 1995.

[ScoClG91] Scott,A.C. - Clayton,J.E. - Gibson,E.L.: A Practical Guide to Knowledge Acquisition. Addison-Wesley, 1991.

[Shap83] Shapiro, D.: Algorithmic Program Debugging, MIT Press, Cambridge 1983.

[Shortl76] Shortliffe, E. H.: Computer-Based Medical Consultations: MYCIN. New York, Elsevier 1976, 264 pp.

[ShaDie90] Shavlik, J. W. - Dietterich, T. G. (eds.): Readings in Machine Learning. Morgan Kaufmann 1990, 853 pp.

[SikSha92] Sikora, R. - Shaw, M.: The Evolutionary Model of Group Problem-solving: A Computational Study of Distributed Rule Learning. Technical Report UIUC-BI-AI-DSS-92-03, University of Illinois at Urbana-Champaign, 1992.

[Sleem95] Sleeman, D.: Knowledge Base Refinement and Theory Refinement. In: State of the Art in Machine Learning. Special Issue of MLnet News, the Newsletter of the European Network of Excellence in Machine Learning, September 1995, 45-54.

[Slodz94] Slodzian, A.: Configuring decision tree algorithms with KresT. In: ML-Net Workshop on Learning and Knowledge Level, Catania 1994.

[SmyGoH90] Smyth, P. - Goodman, R. M. - Higgins, C.: A Hybrid Rule-based Bayesian Classifier. In: Proc. European Conf. on Artificial Intelligence, Stockholm 1990, 610-615.

[Sommer94] Sommer, E. et al.: MOBAL 3.0 User Guide. German National Research Center for Computer Science (GMD), St. Augustin 1994.

[Svatek94a] Svátek, V.: Automatic Knowledge Base Construction using Expert Knowledge and Data. In: Kybernetika a umelá inteligencia [Cybernetics and Artificial Intelligence], Herl'any 1994.

[Svatek94b] Svátek, V.: Compositional Rule Learning from Expert and Data. In: AIT'94, International Workshop on Artificial Intelligence Techniques, Brno 1994.

[Svatek95] Svátek, V.: Integration of rules from expert and rules discovered in data. In: Statistics, Machine Learning and Knowledge Discovery in Databases. Ed.: Kodratoff, Y. - Nakhaeizadeh, G. - Taylor, C., Heraklion, FORTH 1995, 124-129.

[Svatek96] Svátek, V.: Learning with Value Hierarchies in the ESOD framework. In: AIT'96 - 3rd International Workshop on Artificial Intelligence Techniques. Ed.: Žižka, J., Brno, TU Brno 1996, 102-109.

[Svatek97] Svátek, V.: Exploiting Value Hierarchies in Rule Learning. In: van Someren, M. - Widmer, G. (Eds.): ECML'97, 9th European Conference on Machine Learning. Poster Papers. Prague 1997, 108-117.

[TanShi93] Tangkitvanich, S. - Shimura, M.: Learning from an Approximate Theory and Noisy Examples. In: Proceedings of the 11th National Conference on Artificial Intelligence (AAAI'93), 465-471.

[TanHay93] Tansley, D. - Hayball, C.: Knowledge-Based Systems Analysis and Design. A KADS Developer's Handbook. Prentice Hall 1993.

[TecKod90] Tecuci, G. - Kodratoff, Y.: Apprenticeship learning in imperfect domain theories. In:[KodMic90], 515-551.

[ThoLaG93] Thomas, J. - Laublet, P. - Ganascia, J. G.: A Machine Learning Tool Designed for a Model-Based Knowledge Acquisition Approach. In: EKAW-93, European Knowledge Acquisition Workshop, Lecture Notes in Artificial Intelligence No.723, N.Aussenac et al. (eds.), Springer-Verlag, 1993, 123-138.

[Thomas94] Thomas, J.: Learning within a Problem Solving Method: An Empirical Evaluation. In: Proceedings of the ECAI workshop on Integration of Machine Learning and Knowledge Acquisition, Amsterdam, August 1994, 51-57.

[Thrun91] Thrun, S. B. et al: The Monk's Problems. A Performance Comparison of Different Learning Algorithms. Carnegie Mellon University 1991, 154 pp.

[TinLow97] Ting, K. M. - Low, B. T.: Model Combination in the Multiple-Data-Batches Scenario. In: van Someren, M. - Widmer, G. (Eds.): ECML'97, 9th European Conference on Machine Learning. Lecture Notes on Artificial Intelligence, Springer Verlag 1997, 250-265.

[Torgo93] Torgo, L: Controlled Redundancy in Incremental Rule Learning. In: Machine Learning - ECML'93, European Conference on Machine Learning, Vienna 1993. Lecture Notes on Artificial Intelligence, Springer Verlag 1993, 187-195.

[TorKub91] Torgo, L. - Kubát, M: Knowledge Integration and Forgetting. In: AI'91 International Workshop, Prague Technical University, 1991, 169-182.

[Utgoff88] Utgoff, P. E.: ID5: An Incremental ID3. Proceedings of the Fifth National Conference on Machine Learning, Michigan 1988, 107-120.

[Vali84] Valiant, L.: A theory of the learnable. *Communications of the ACM*, 27 (11), 1134-1142.

[VMeDec95] Van de Merckt, T. - Decaestecker, C.: About Breaking the Trade Off Between Accuracy and Comprehensibility in Concept Learning. In: Proceedings of the Workshop on Comprehensibility in Machine Learning, IJCAI-95, Montreal.

[VVelde94] Van de Velde, W.: Towards Knowledge Level Models of Learning Systems. In: MLNet Workshop on Learning and Knowledge Level, Catania 1994.

[VDoVSo94] Van Dompseler, H. J. H. - Van Someren, M. W.: Using Models of Problem Solving as Bias in Automated Knowledge Acquisition. In: ECAI'94 - European Conference on Artificial Intelligence, Amsterdam 1994, 503-507.

[VSom89] Van Someren, M. W.: Reducing the Description Space for Rule Learning. In: Kodratoff, Hutchinson (eds.): Machine and Human Learning. Michael Hornwood Limited, London, 1989, 173-183.

[VSom93] Van Someren, M. W.: Report on the Learning and Problem Solving Familiarization Workshop in Blanes 1993. *MLnet News*, Vol.2, No.1, 6-7.

[VSom95] Van Someren, M. W.: A perspective on Machine Learning and Comprehensibility from Knowledge Acquisition. In: Proceedings of the Workshop on Comprehensibility in Machine Learning, IJCAI-95, Montreal.

[Vitas97] Vitásek, J.: Building an expert system in MOBAL environment. [Master thesis, in Czech], Prague University of Economics, 1997.

[Wick94] Wick, M. R.: Explanation as a Primary Task in Problem Solving. *The Knowledge Engineering Review*, Vol.9:1, 1994, 18-82.

[Wider86] Widerhold,G.: Knowledge versus Data. In: (Brodie, Mylopoulos eds.) On Knowledge Base Management Systems. Integrating Artificial Intelligence and Database Technologies. Springer Verlag, 1986.

[WidKub93] Widmer, G. - Kubát, M.: Effective Learning in Dynamic Environment by Explicit Context Tracking. In: Machine Learning - ECML'93, European Conference on Machine Learning, Vienna 1993. Lecture Notes on Artificial Intelligence, Springer Verlag 1993, 227-243.

[WieScB92] Wielinga, B. - Schreiber, T. - Breuker, J.: KADS: a modelling approach to knowledge engineering. *Knowledge Acquisition*, 4(1), 1992, 5-53.

[Wilk90] Wilkins, D. C.: Knowledge base refinement as improving an incorrect and incomplete domain theory. In: Kodratoff, Michalski (eds.): Machine Learning: An Artificial Intelligence Approach. Vol.III, Morgan Kaufmann 1990, 493-513.

[WiBe93] Winkelbauer, L. - Berka, P.: ALEX: Towards Adaptive Learning in a Multi-Algorithm Environment. LISp Technical Report 93-02, Prague University of Economics, Prague 1993.

[Winst75] Winston, P. H.: Learning structural descriptions from examples. In: Winston (ed.): The Psychology of Computer Vision. New York, McGraw Hill 1975.

[Winst92] Winston, P. H.: Artificial Intelligence. Addison-Wesley 1992, 735 pp.

[Wrob93] Wrobel, S.: On the proper definition of minimality in specialization and theory revision. In: Machine Learning - ECML'93, European Conference on Machine Learning, Vienna 1993. Lecture Notes on Artificial Intelligence, Springer Verlag 1993, 65-82.

[Wrob94a] Wrobel, S.: Concept Formation and Knowledge Revision. Contributions to a computational theory. Kluwer Academic Publishers, Dordrecht 1994.

[Wrob94b] Wrobel, S.: Heuristic Control of Minimal Base Revision in KRT Using a Two-Tiered Confidence Model. In: MLnet Workshop on Theory Revision and Restructuring. Catania 1994.

[Wrob94c] Wrobel, S.: Concept Formation During Interactive Theory Revision. *Machine Learning*, 14 (1994), 169-191.

æ

Glossary of Abbreviations

AI	artificial intelligence
CLT	concept learning tool
CSD	control-site description
DBS	database system
D/E	data/examples
EBG	explanation-based generalization
EBL	explanation-based learning
ESOD	Expert System from Observational Data
FR	flat ruleset
GTM	generic task model
HR	hierarchical rulebase
ILP	inductive logic programming
I/O	input/output
KADS	Knowledge Acquisition and Design System
KAW	Knowledge Acquisition Workshop
KBS	knowledge-based system
KEX	Knowledge Explorer
KRT	knowledge revision tool
MDL	minimal description length
MML	minimal message length
MK	meta-knowledge
ML	machine learning
PSK	problem-solving knowledge
RDT	rule discovery tool
SA	source attribute
SDK	static domain knowledge
TA	target attribute
TDIDT	top-down induction of decision trees
UCI	University at California-Irvine

æ