The Role of Qualitative Knowledge in Machine Learning*

Johannes Fürnkranz
juffi@ai.univie.ac.at
Austrian Research Institute for Artificial Intelligence
Schottengasse 3
A-1010 Vienna
Austria

November 31, 1992

Abstract

This paper analyzes the important role qualitative knowledge plays in Machine Learning. For this purpose important results from research in the fields approximate theory formation, automated qualitative modeling, learning in plausible domain theories and learning with abstractions are reviewed. The analysis of these approaches shows several of the benefits the use of qualitative knowledge can bring to Machine Learning and also points out important problems that have to be dealt with. The need for qualitative knowledge to keep learning tractable is illustrated with examples from the domain of chess. Finally we make some suggestions for further research based on the shortcomings of previous approaches.

*This research is sponsored by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant P8756-TEC. Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry for Science and Research.

Contents

1 Introduction

Knowledge representation is one of the oldest and probably the most important issue in Artificial Intelligence. In the early years of AI research it was widely believed that most problems can be solved with fairly general problem solving methods, given that an appropriate formalization and representation can be found. The most outstanding example of this optimistic point of view was Newell and Simon's "General Problem Solver" [Newell *et al.*, 1960], which operates in a simple state space representation of the world.

Later on research mostly concentrated on trying to find better and more efficient inference algorithms, and the knowledge representation side was more and more neglected. Research in Computer Chess is a typical example of this lack of appropriate knowledge representation. A chess computer's knowledge is encoded in an evaluation function that assigns a numerical value to each position. The only design criterion for evaluation functions is that they have to include basic chess concepts in a very efficient way to speed up the search process. Consequently, chess playing programs are very hard to understand and to debug, as detecting the parts of the knowledge that are responsible for a bad move is very complicated. In this respect the basic ideas of today's chess programs have not changed considerably since the proposal of the MiniMax algorithm by Claude Shannon in 1950 [Shannon, 1950]. Improvements in hardware [Hsu, 1987] and in the efficiency of the MiniMaxing method — especially through the use of Alpha-Beta Pruning [Knuth and Moore, 1975] — has led to the chess playing programs of today that are able to challenge Grandmasters and maybe in the near future even the World Champion. Thus research in Computer Chess has led to the development of very efficient search algorithms, but has contributed very little to research on human chess playing. It is significant that none of today's leading programs are able to learn from their faults.[1]

[1] In Deep Thought the weights of the parameters of the evaluation function have been tuned using a simple reinforcement algorithm [Hsu *et al.*, 1990], but the values of the parameters are not changed during tournament play. Bebe can avoid playing a losing

Later on it was proposed that large amounts of knowledge rather than efficient and sophisticated general reasoning procedures are the key to intelligent performance of a computer system, the so-called *Knowledge Principle* [Lenat and Feigenbaum, 1991]. This philosophy has also led to several chess programs that tried to make use of explicit representation for plans and goals. But these approaches were only successful in limited subdomains of chess, as e.g. tactical combinations [Wilkins, 1982], pawn endgames [Berliner and Campbell, 1984] etc. The main problem was the complexity of the domain. First of all it is very hard to formulate precise rules or plans for chess as adding or removing only one piece on the board can have a significant influence on the outcome of the game. Second, even if there would be a set of rules that could be used for good play, it is quite likely that the size of this set would be too big to allow a game within reasonable time limits.

We think that the slow progress in knowledge based chess systems (and in many other complex domains) is due to the lack of appropriate representation formalisms that allow efficient reasoning in complex domains. Classical representations in first-order logic are not sufficient to deal with complex real-world domains in a tractable way. In recent years this problem has been addressed independently in many research areas in AI and in particular in Machine Learning. The purpose of this paper is to give an overview of the research done in learning of and in the presence of various forms of "qualitative" background knowledge. In this paper the term "qualitative" will be used for all kinds of knowledge that attempt to overcome the intractability of complex domains by allowing uncertainty.

When forced to attempt a definition of what we mean by "Qualitative Knowledge", the following negative definition would be the result:

Definition 1.1 (Qualitative Knowledge) Qualitative Knowledge *is any kind of knowledge that does not always allow a correct and consistent match between the represented objects and the real world, but can nevertheless be used to get approximate characterizations of the behavior of the modeled domain.*

---

line twice by saving the board positions occurring in a game in a long term memory [Scherzer *et al.*, 1990], but it does not generalize in any way. It will still make the same *type* of mistake over and over again.

Various fields of research can be labeled as using *qualitative knowledge* according to this definition. Among them are

- Explanation-Based Learning with approximate theories
- Automated Qualitative Modeling
- Learning of theories with different abstraction levels
- Plausible Reasoning

In chapter 2 we show the need for what we call *Qualitative Knowledge* to facilitate learning in various complex domains. Although chess — being the favorite domain of the author — will appear quite often in this paper, we want to stress that the general principles exemplified by this domain are valid for all complex, intractable domains (e.g. reasoning about physical systems [Weld and deKleer, 1990], medical domains [Bratko *et al.*, 1989], music [Widmer, 1992b], chess [Tadepalli, 1989], [Flann, 1989] etc.).

The next three chapters give a review of previous work in Machine Learning on the topics of approximation, learning with qualitative and plausible theories and abstraction. Very loosely these chapters correspond to the three different kinds of abstraction as seen in [Doyle, 1986]:

Approximation is the simplification of the domain theory through assumptions that some condition holds or some constraint is satisfied.

Qualitization is the mapping of a continuous description of a domain into a discrete one.

Aggregation is the subsumption of complex structures under simpler structures.

Doyle's notion of *abstraction* closely corresponds to our notion of *qualitative knowledge*. We will use abstraction whenever a change of the representation level occurs during learning. So while the next two chapters focus on the characterization and representation of approximate and qualitative domain theories, chapter 5 concentrates on the benefits of different ways of moving from one level of detail to a more approximate, abstract representation level.

The three chapters are also ordered according to the amount of uncertainty that is explicitly captured in the representation languages. While

approximate theories (chapter 3) can be expressed in any representation language by simply not formulating rules that are not necessarily correct, qualitative and plausible theories (chapter 4) need special language constructs to capture various uncertainties of the domain. Qualitative modeling, for instance, introduces discrete characterizations of various continuous behaviors in physical domains. Finally chapter 5 deals with deep theories that have various levels of detail in their representations.

2 The need for Qualitative Knowledge

We will now look at the question of why we need qualitative knowledge at all. For this purpose several properties of complex domains that cause problems for classical knowledge representation methods are discussed shortly. From this section the need for other, qualitative knowledge representation becomes apparent. We use the domain of chess as our running example, because it is sufficiently complex to exhibit all problems that representations of real-world domains usually have to solve, but it still is a closed domain, relatively easy to survey, where the quality of a system's knowledge can be judged quite accurately. For these reasons chess has been called "the drosophila[2] of AI" alluding to the important role this comparably simple animal played as the object of early research in genetics [McCarthy, 1990].[3] In particular chess has become a standard example [Tadepalli, 1986, Tadepalli, 1989, Flann, 1990] for an *intractable domain*.

2.1 Knowledge is intractable

Many applications of rule-based production systems assume that the underlying theory is complete and consistent. However, real-world domains usually can't be formalized in a nice and neat way [Rajamoney and DeJong, 1987]. It is often the case that not all information needed is available or that the system's knowledge is not correct. Even in domains where it would be possible to have complete and correct knowledge about the environment, the size of the knowledge base or of the implied search space would be too big to be useful. Complex complete information games like chess are good examples of such *intractable domains* [Mitchell *et al.*, 1986]. Theoretically, a chess

[2] fruitfly

[3] As most experts nowadays consider a silicon chess world champion to be just a matter of time, a trend has developed that the game of Go should take up that role, because it is not that accessible to brute-force searching algorithms and thus imposes stronger challenges to the proper formalization of knowledge (see Part V of [Marsland and Schaeffer, 1990]).

playing system has all the information it would need to determine whether a given position (in particular the starting position) is a win, a loss or a draw. Every medium advanced programmer could write a program that could play perfect chess, if it were not for the limitation that it would take longer than the universe exists for the program to make its first move. Thus, as complete and correct domain theories usually are unavailable or intractable, ways have to be found to deal with theories that are incorrect, incomplete or both.

2.2 Knowledge is inconsistent

One may of increasing the tractability of a domain may be to allow not necessarily correct and consistent domain theories. Especially *over-generalization* seems to be a powerful method to obtain simple, but powerful rules. People use over-general heuristics like "People from the same country speak the same language" in common-sense reasoning. Simple rules that are correct for *most* of a large amount of data are often preferred over rules that cover a whole, but small data set (see also section 2.5).

To stick with the chess example, it is reasonable to assume that winning a knight is an advantage sufficient for a win. Material advantage is a very important factor in the evaluation of a chess position, but unfortunately, this knowledge is too general to be correct. In many positions a piece can be sacrificed for compensation of some other form, e.g. mobility.[4] Attempts to incorporate the intuitive knowledge of "compensation for sacrifices" into chess playing program have failed so far, mostly because an exact formalization of the term "compensation" is not available in the chess literature and probably can't be found.

Needless to say that incorrect theories impose major problems for reasoning as well as for learning. Multiple explanations may arise for a fact, a fact might even be disproven and proven with the same theory, thus resulting in inconsistency. New forms of — plausible — inferences have to be found to cope with this problem.

[4]Note that I'm not necessarily talking about sacrifices that result in a mating attack or at least win the sacrificed material back in a couple of moves. Many sacrifices can not be justified by analysis, it is part of a grandmaster's knowledge to "feel" when a position gives compensation for sacrificed material *in the long run*.

2.3 Knowledge is incomplete

Another way to deal with the intractability of many domains is to try to reason with incomplete knowledge. In general not all preconditions for the proof of a fact are known. What should be done in cases like this is to look for *plausible* explanations for the missing parts of the proof. If a fact cannot be proved to be true, but arguments to strengthen the belief in it can be found, it is plausible to assume its truth. The extent to which the plausible explanation should be trusted, depends on several factors including the relevance of the fact for the overall goal and the number and strength of supporting (and disagreeing) arguments.

In chess a significant amount of thinking time is used for trying to figure out your opponent's plan and then to either prevent it or to simply ignore it and to follow your own plan (which in that case had better been stronger than your opponent's). This process can be viewed as trying to find a "proof" for your best move, which includes to find out `plan( opponent, X)`. No exact proof for any plan `X` can be given, as this information is in general not available. Nevertheless the current situation on the board and the history of the game may provide evidence for or against certain values of `X`.

2.4 Knowledge has various representation levels

Human experts in a domain mostly solve common problems in that domain without much reasoning. Apparently experts can immediately and intuitively associate the appropriate action with the current situation. On the other hand, when they are faced with unfamiliar situations, they can rely on a profound knowledge about the domain and are capable of solving the problem with more elaborate methods.

In chess playing this multi-level representation of knowledge becomes obvious at speed or blitz chess. Here the players each have five minutes or less to play the whole game. Whoever uses up his allotted time first, loses (unless the game is decided by conventional means first). This game allows no deep computations of move sequences, players have to decide on their moves almost immediately. Thus they have to rely on their "intuition" to make the right moves. Presumably very efficient pattern recognition methods associate very few candidate moves with the position at hand and one of them is chosen with a very limited look-ahead.

Nevertheless, the quality of play in Blitz Chess is not that different from the quality of play in regular chess as one might expect, given the differences in the amount of time spent for analyzing the positions [Church and Church, 1983]. The reason for this is that even in regular chess, experienced players neither examine considerably more moves nor calculate deeper variations than beginners. The difference is that the knowledge they use in the problem solving process allows them to compentently and efficiently select the "right" moves for further examination [deGroot, 1965].

Thus it seems to be the case that chess players have at least two levels of knowledge representation:

- a very efficient knowledge that immediately produces approximate results (good candidate moves)
- a more elaborate model about weaknesses and strengths of positions and moves which has to be searched with more expensive problem solving methods

Apparently both forms of knowledge are closely connected and it is probably hard to assign pieces of knowledge to either of the two categories, so that it seems to be a plausible assumption that both categories represent the same kind of knowledge at a different level of detail. "Shallow", efficient rules suggest candidate moves which are then submitted to a "deep", causal theory for closer evaluation (unless time constraints prevent that).

2.5 Knowledge is simple and approximate

It has already been stressed in sections 2.2 and 2.4 that simple rules should be preferred over complicated ones even at the cost of lack of precision. The principle that simpler rules, all other things being equal, are more likely to be predictive for future data — known as *Occam's razor* — has been thoroughly examined [Blumer *et al.*, 1987] and widely applied as a preference criterion for rule formation in Machine Learning algorithms [Michalski, 1983]. Nevertheless, the importance of this principle is still underestimated. In most Machine Learning algorithms a correct, complicated rule is preferred over a simple, approximate rule. Humans seem to use slightly over-generalized rules in their commonsense reasoning (cf. also the examples in section 2.2).

The concept of a "fork" is one of the first pieces of chess knowledge a player acquires. Although the concept "fork" is easy to understand, it is hard to formalize. The intuitive definition — a piece attacks two or more pieces such that the player can gain material by capturing one of them — can hardly be converted into a general rule, as there are many ways for the opponent to prevent loss of material (e.g. capture the piece that forks; move one of the forked pieces and give check, so you can save the other one in the following move; threaten other pieces etc.). Many learning algorithms would end up with a set of separate recognition rules for highly specialized concepts like "knight fork where the knight is not attacked and one of the attacked pieces is the opponent's king" etc. Human chess players, on the other hand, seem to base recognition of forks on very simple visual patterns. Information about whether the expected gain of material will actually take place, is checked seperately in every case the recognition rule fires and says "Look at this, there *might* be a fork".

So it is not only desirable to learn correct rules, but also to learn approximate, but simple rules that, although incorrect in some exceptional cases, combine high predictivity with efficiency. Simple, but powerful rules are desirable, even if they are imprecise.

2.6 Knowledge is noisy

Very related to the topic of approximate and incomplete knowledge is the fact that knowledge is often *imprecise* and *noisy*. Very often data needed by a system is automatically acquired by various input mechanisms, which are subject to errors and imprecisions.

An intelligent agent interacting with the real world has to rely on its sensory data. Although errors in sensing devices usually are small, they are nevertheless there. Numerical methods for reducing the errors like e.g. averaging over multiple readings exist, but they can't guarantee precise measurements either. Errors in discrete sensing devices are even harder to deal with.

Mechanisms for representing uncertainty and noise in data are needed. The effects of noise on input data has been widely studied (see e.g. [Angluin and Laird, 1988, Clark and Niblett, 1987]), but the problem of distinguishing noise in the data from rare instances of the target concept has been addressed only recently [Srinivasan *et al.*, 1992] and is mostly based

on statistical heuristics. Qualitative knowledge about the approximate behaviour of the environment might help to recognize noise, because of its lack of plausibility.

As the environment of a chess playing system (the chess board) is rather simple, errors because of noise in data are rather unlikely to occur. Nevertheless, human chess players make mistakes that could be interpreted as resulting from noisy input data. E. g. a common mistake is that when calculating a sequence of moves you move a piece twice, the second time forgetting that it already has moved. "Hanging a piece", also a mistake that occurs more often than chess players like, results of incomplete information about the current situation on the board — a threat has been overlooked. Note that this kind of error is different from not knowing about a certain kind of threat. Human players obviously do not perform a systematic "forward chaining" search for threats, otherwise pieces wouldn't hang that often. Besides, a complete search for threats would be far too inefficient for the "human hardware".

2.7 Knowledge is acquired in various ways

Another important point to mention, although not directly related to knowledge representation, is that knowledge is acquired and used in a variety of ways. Chess players gain much by continuing practice that presumably automates ("operationalizes") their theoretical knowledge. This knowledge in turn has most often been taught by friends, teachers or books. A player can prepare his opening repertoire by studying books and memorizing moves or by continuous practice, but most efficiently with both.

Operationalizing already acquired knowledge is nothing else than Explanation-Based Learning, while learning openings by mere practice is done by induction. Learning from instruction is often done by analogy, as prototypical, instructive instances are presented that will be remembered in similar situations.

Closely related to the different ways of knowledge acquisition is that knowledge can be used in several ways. We already have discussed issues of efficient "intuitive" knowledge and inefficient "deep" models in section 2.4. The former apparently is immediately associated with certain features of the environment (a fork is recognized without any processing) while the latter has to be searched more systematically.

3 Rule Approximation

As we have seen in chapter 2, in sufficiently complex domains there is a need for the use of approximations in order to increase the tractability of the domain knowledge. An approximation of a theory intuitively is a simpler theory in the same representation language that can be used to derive most of the results required in a more efficient way.

We will try to analyze approximations in the next section. After that, several approaches to learning of, and in connection with approximate theories will be introduced.

3.1 Approximations and Simplifying Assumptions

Consider the following approximate definition of the concept fork in chess: "A *fork* is when a piece threatens two of his opponent's pieces (either of higher value or en prise) at the same time. This wins material, as the other player can only move one of the pieces" The given definition is exact in board positions where the opponent has no defending resources like taking the threatening piece, moving out of the fork and simulatneously giving a counterthreat, etc. Thus the approximation might be viewed as an exact theory of a subdomain of the original problem where several additional assumptions simplify the problem.

Thus most authors [Doyle, 1986, Ellman, 1988] define approximation in a way similar to this:

Definition 3.1 (Approximation) *An* approximation T_A *of a domain theory* T *may be viewed as making* useful *simplifying assumptions about the domain of* T *and then reformulating* T *in the new simplified subdomain.*

Note that this definition does not define the quality of an approximation. There are many subdomains of a domain, and clearly most of them are not appropriate for forming approximations. Identifying different types of *useful* simplifying assumptions will be the purpose of the next section.

We also define

Definition 3.2 (Simplifying assumption) *A* simplifying assumption *is a mapping of a domain D into a subset D_S of itself.*

Domain simplifying assumptions may be explicitly stated or implicitly assumed. By simplifying the domain theory they can improve performance in reasoning and learning. It is much easier to check for the mere pattern of a knight fork than to analyze whether it will actually be successful or not. This gain in efficiency and simplicity unfortunately must be traded off with precision and consistency. One of the major problems that arises when using approximations is that approximate theories may be *inconsistent*. In problems outside of the simplified subdomain, explanations may contradict each other. The fork in chess promises material gain, but when the assumption of no appropriate defensive resources does not hold, a simple look-ahead of one ply might tell the player that it will lose material, when e.g. the forking piece is *en prise*. Thus two contradicting results can be derived. Of course, multiple non-contradicting explanations are possible as well.

To keep the loss in precision small, it is advisable to use assumptions that are true in most typical cases. Ideally approximations should also yield a bias towards a correct solution in cases where the underlying assumptions are not true. The approximate definition of a "fork" is — despite the danger of producing inconsistencies — still useful, as it tells the chess player that there might be a chance of winning material, thus biasing his search to moves and plans that may prevent the threat. The player thus can evaluate the opportunity by examining the position at a finer level of detail (see also sections 2.4 and chapter 5).

So among the problems that have to be dealt with when using approximate theories are

- finding the right approximations
- reasoning and learning in incomplete or inconsistent theories
- multiple explanations

3.2 Learning of Approximations

Learning approximate theories has mostly been studied in connection with Explanation-Based Learning. In [Mitchell *et al.*, 1986] it has already been

noted that imperfect theories are a major drawback of EBL. Several authors have given the problem a closer examination and several approaches were proposed to extend explanation-based learning to be able to use inconsistent, incomplete or intractable domain theories (e.g. [Rajamoney and DeJong, 1987, Tadepalli, 1986, Danyluk, 1989, Sebag and Schoenauer, 1992]). This chapter will be concerned with the learning of approximate rules. As we have seen in the preceding section, we have to identify useful simplifying assumptions for finding good approximate theories.

3.2.1 Approximation Hierarchies

Ellman [Ellman, 1988] was the first to stress the importance of identifying appropriate simplifying assumptions for approximate theories. He views finding the right simplifying assumptions as the main goal of explanation-based learning of approximate theories. Ellman's system POLYANNA thus considers approximation as a search through the space of possible simplifying assumptions.

For this purpose assumptions are encoded in generic functions representing a hierarchy of increasingly specialized implementations of the same function. This hierarchy induces a similar hierarchy on the space of approximate theories created by combining various functions at various levels of abstraction. Thus it forms a lattice according to the relation that a theory is more special than another when it contains a generic function that is more special than its equivalent in the other theory. This space can be systematically searched, starting at the root (the simplest theory) and moving down the hierarchy to more complex theories only when simpler ones are contradicted by training examples. The algorithm takes an error threshold as input and stops at the simplest — i.e. the first — theory that meets the specified error rate, thus finding an acceptable trade-off between shortening the process of explanation building and correctly explaining many examples, i.e. finding a useful approximation. Error rates are estimated empirically by using a teacher-provided set of training examples.

The system was applied to derive heuristics for the game of Hearts and a uniprocessor scheduling task [Mostow *et al.*, 1990].

3.2.2 Incremental Approximation

Lazy Explanation-Based Learning [Tadepalli, 1989] is a very radical approach to learning in intractable two person games like chess. One of the major problems in these domains is that — in order to prove a move to be correct — all possible moves of the opponent have to be considered. Generalizing this into general concepts is hard to do, as in a slightly different situation on the board your opponent might have a totally different set of responses and counter-plans. Lazy EBL's philosophy is to attack this problem by paying no attention to the possible refutations of a move. The system learns a set of over-general, optimistic plans called *o-plans*. All moves in the plan contribute to the achievement of either the goal itself or of one of its preconditions. During actual play a planning module combines all o-plans that seem to be relevant for the current board situation into so-called *c-plans*. Over-general plans can be refined by learning a new o-plan that will be indexed in the original plan as a counter plan. Explanation-Based Learning is invoked each time the system encounters an unexpected plan failure. Then the c-plan that has been constructed from previously learned o-plans is generalized into a new o-plan that only considers moves that have either been suggested by its c-plan or that were actually performed on the board. Thus only a subset of possible moves is considered. Tadepalli calls this simplifying assumption *Omniscience Assumption*: The planner assumes that it knows all the o-plans necessary to explain all the relevant alternative moves by both the players. The hope is that the more o-plans are learned the more alternative moves for both players will be suggested and thus the better the play will be.

The work described in [Chien, 1989] is very similar to Lazy EBL. While Lazy EBL is designed mainly for the intractability arising from the need to consider all the opponent's possible moves in two-person games like chess, Chien's extension to EBL incrementally approximates and corrects plans in the domain of STRIPS-like planning. Here intractability arises through the unknown side-effects of the application of some operators. In the initial learning phase an approximate plan is learned by considering only *immediate* effects of operator applications. As in Lazy EBL, each time an expectation failure occurs, the system tries to explain the failure and to correct it with plan refinement (in the case of unexpected goal failure) or with specializing over-general failure explanations (in the case of unexpected goal achievements). The inaccuracies in the learned knowledge arise because — although

the system performs a complete update of all *immediate* consequences of an operator application — it does not consider possible *inferred* effects. As the systems assumes everything that has become true to be true until explicitly contradicted, failures may arise when unnoted inferred effects clash with preconditions of later operators. This clearly is another instantiation of Tadepalli's *Omniscience Assumption*.

A difference in the two systems, however, lies in the way they refine over-general schemas. While Lazy EBL learns new counter-plans to correct over-general plans, Chien's system simply specializes the preconditions of the existing approximate rules.

Although Lazy EBL and Chien's Incremental EBL seem to differ considerably from Ellman's POLYANNA, the algorithms are actually quite similar. Both search a space of approximate theories, moving on to more complicated theories when errors in the simpler theories have been detected. Thus they find the simplest correct theory. The main difference is that POLYANNA learns from a batch of examples provided by a teacher, while Lazy EBL detects errors in its theories by expectation failures in actual play. Lazy EBL thus incrementally changes its theory, while POLYANNA relies on having a representative set of examples in advance.

One of the common characteristics of both Lazy EBL and Incremental EBL is that they change their theories as soon as the approximation proves to be incorrect in some cases. Theory approximation is thus treated as a means to converging towards a correct and consistent theory. This has led to the definition of *Probably Approximately Tractable* (PAT) Learning [Tadepalli, 1990] which tries to extend the notion of Probably Approximately Correct (PAC) Learning as introduced in [Valiant, 1984] to capture the additional constraint of learning a tractable theory. In their way of incrementally approaching a target concept the systems are quite similar to approaches that repair plans from failure [Hammond, 1990].

3.3 Approximate Background Knowledge

The algorithms of the last section try to make simplifying assumptions to speed up the learning process. In this section approximate background knowledge is used for narrowing down the search space for inductive learning algorithms.

ML-SMART [Bergadano and Giordana, 1988] is an algorithm for inducing concept description in first-order logic from specified positive or negative instances. In addition background knowledge can be specified that may be incorrect or incomplete. ML-SMART searches the rule space to find a rule that is consistent with the given examples for the target concept. It starts a best first general to specific search and specializes the hypothesis that everything is an instance of the target concept by successively choosing one of the non-operational predicates and expanding one of the rules of the background knowledge that fire on this predicates. If no such rule can be found, but the current hypothesis is still too general, it is specialized by inductive means. Too specialized rules can be detected by a heuristic that watches the number of covered examples of each hypothesis and can be generalized by dropping conditions. All hypotheses are organized in a tree structure such that the most promising hypothesis — according to a heuristic that trades off completeness and consistency against the ratio between the number of operational and non-operational predicates in the rule — can be chosen for further modification. The algorithm can also be reformulated in an incremental way as has been shown in [Widmer, 1989a].

A related approach can be found in [Pazzani and Kibler, 1992]. FOCL (First-Order Combined Learner) tries to combine the virtues of the First-Order Inductive Learner FOIL [Quinlan, 1990] and Explanation Based Learning. It allows the user to specify extensionally defined background knowledge as well as intensionally defined training examples. Background knowledge can be in the form of typing constraints for predicates, information about the variabilization of the predicate's arguments (e.g. all different) and partially learned or even incorrect rules in an either operational or non-operational way. Non-operational rules will be operationalized by expanding the clause that promises the highest information gain, thus performing a heuristic hill-climbing search, instead of ML-SMART's best first search. This is continued until operational predicates that can be checked or learned are reached on the leaves of the 'proof tree'. The various forms of background knowledge allow FOCL to reduce the search space considerably and provide means for a more goal-directed search. Even the specification of incorrect rules gives a better performance than the no background knowledge at all, as long as the trainings examples are informative enough to allow for an inductive patching of the errors.

3.4 The Utility Problem

Although Tadepalli shows most of the algorithms of the section 3.2 to be probably approximately tractable, it is quite conceivable that most of the systems — when faced a complex learning task — will learn huge theories. [Minton, 1984] reports the problem of learning too many too specialized rules with explanation-based learning in several game domains including chess. This experience has motivated his research with PRODIGY where he tried to make sure that "the cumulative benefits of applying the knowledge must outweigh the cumulative costs of testing whether the knowledge is applicable" [Minton, 1990]. This has been known as the *utility problem* (see also [Cohen, 1992] and section 3.5.2).

If approximations incrementally approach a correct and consistent domain theory as in the last section, mistakes will always be corrected by either learning new rules as in Lazy EBL or by further specializing the preconditions of learned rules as in Incremental EBL. No criterion is applied that allows to stop learning and rely on different kinds of problem solving.

In section 2.1 we have tried to suggest that domains like chess are inherently intractable. No complete and consistent formalization of it can be tractable. Approximate knowledge that guides problem solving towards a solution can help to increase the tractability of the domain. But in order to make use of them, a criterion has to be found to assess the quality of an approximation. If approximate rules are incrementally refined every time they do not produce the right result, the problem arises that more and more rules will be learned for more and more specialized concepts, so that problem solving with the acquired knowledge becomes intractable again because of the huge number of rules that have to be processed.

In the domain of chess we have seen that approximate rules are mainly used to guide the search into a certain direction. When a chess player recognizes the pattern of a "fork", this is only the start of a complex, but tractable problem-solving process that checks if the fork actually will result in a gain of material. Forks that do not end up in a gain of material will not cause a further refinement of the rules. Approaches using abstraction hierarchies seem to attack this problem in a similar fashion: Concepts recognized at a high level of abstraction do not necessarily have to be true after examining the situation with a more detailed domain theory. This is known as the "false proof" problem (see [Plaisted, 1981, Tenenberg, 1987] and section 5.2).

3.5 The Multiple Explanation Problem

Reasoning with inconsistent theories has not been thoroughly investigated yet. [Roos, 1992] gives a framework for using logic in inconsistent domain theories, but no large-scale implementation of a system for reasoning with inconsistent knowledge is available so far. Nevertheless several authors have faced the problem of dealing with multiple, possibly inconsistent explanations (see also section 2.2) in Machine Learning.

Multiple inconsistent explanations can arise from incomplete domain theories, when some information is missing and must be assumed, or from incorrect domain theories that contain one or more over-general rules.

3.5.1 Selecting the best explanation

An obvious way to deal with multiple explanations is to try to rule out impossible or implausible explanations until only one explanation is left which clearly is the best. [Rajamoney and DeJong, 1987] have designed a system that is able to design experiments to gather enough information to decide which explanations cannot hold for the given situation. Similarly, experiments can be conducted to find the only completion of an incomplete proof.

Another approach is tried in [Fawcett, 1989]. Even for explanation-based learning in complete and consistent domains, the importance of selecting the "best" explanation to derive the most accurate rule has been recognized [Pazzani, 1988]. Fawcett's work tries to transform this approach to EBL systems in incomplete theory domains. Abduction is used to guess the missing parts of a partial proof. Among all possible abductive completions of the proof the system chooses the most plausible partial explanation for the training instance. In this process heuristics for concise rules, rules that account for many known attributes of the examples while leaving as few antecedents as possible unproven, and for more specific rules are computed into a single integer value that accounts for the plausibility of the partial explanation. Then a new, maximally specific rule covering the training instance is asserted. Some primitive generalizations are performed, but the generated rules for unproved facts should be fed to an inductive generalization procedure, as soon as enough of them have been collected.

The GEMINI system [Danyluk, 1989, Danyluk, 1987] relies on previous experience in completing partial proof trees in incomplete domains. The

author proposes to use knowledge of previous proof trees to bias the induction needed to fill in the gaps of incomplete proofs. Several heuristics are suggested to guide the induction, e.g. eliminating features that already appear in other parts of the proof of an example and eliminating features that generally appear with a high frequency.

In [Keller, 1988] the METALEX system [Keller, 1987] has been extended to deliberately introduce approximations. Two operators — `Truify` and `Falsify` — replace arbitrary subexpressions of initially correct concept descriptions with the atoms `True` and `False`. The operators might as well be applied to search control decisions, e.g. unsolved subgoals might be considered as already solved by application of the `Truify`-operator to the predicate `solved`.

Starting with an initially correct description of the target concept, the system performs a steepest ascent hill-climbing search. The system attempts to apply the `Truify` or `Falsify` operator that maximizes increase in efficiency, while minimizing decrease in effectiveness. The first operational definition that meets certain predefined performance criteria will be used. Experiments have shown that METALEX was able to improve its efficiency due to its ability to explicitly reason about costs and benefits of approximating.

A similar approach to increasing a problem solver's efficiency while maintaining its effectiveness by dropping conditions can be found in the works of [Zweben and Chase, 1988] and [Chase *et al.*, 1989]. The ULS system transforms rules generated by EBL into approximate rules using statistical measures. Rules are generalized by dropping conditions that are true most of the time and do not introduce new variable bindings. For this purpose counters maintain the number of times a condition is true over the number of times a condition is tested, thus estimating the conditional probability that a condition is true given that the already tested conditions of the rules have been true. ULS also keeps a copy of the original rule that can be used to test whether the generalization was bad in case the new rule repeatedly misclassifies examples.

3.5.2 Using a set of explanations

In [Cohen, 1992] the multiple inconsistent explanation problem is addressed by repeatedly marking a rule that gives the highest ratio of newly explained examples to rule size, thus performing a greedy search to cover the set of

examples with a minimum set of rules. Rules covering negative examples are eliminated before this process. The approach is successfully applied to the domain of finding suitable opening bids for the domain of Bridge.

Subsequent work [Cohen, 1990] addressed both, the utility problem (section 3.4) and the *multiple inconsistent explanations problem* by combining induction with explanation-based learning. The utility of a rule is its contribution to performance improvement and as such is directly proportional to the coverage of the rule and inversely proportional to the match cost of the rule. The system thus chooses the explanation which maximizes the ratio of newly explained examples to the size of the rule. With this method multiple, possibly inconsistent explanations can be dealt with (see [Cohen, 1992]).

In addition to this heuristic that — quite similarly to the PRODIGY system [Minton, 1990] — increases the convergence rate of learning, the system also tries to learn approximations of expensive rules as another method to avoid *swamping*. For this purpose approximations of rules are formed by throwing out all but a bounded number of literals of each rule. As the process is exponential on the literal bound, only few literals can be kept. In all but one test domains substantial performance improvement occurred when learning only rules of a length ≤ 2.

[Sebag and Schoenauer, 1992] propose a different approach to this problem. The basic idea of the authors is to learn meta-rules that control the application of the original rules. In a first phase (possibly inconsistent) rules are induced from a set of examples. The instance descriptions are then reduced to two predicates for each rule of the already learned knowledge. One predicate is true when the rule has fired, the other when the rule has not fired. After all examples have been reduced like this, an inductive learning mechanism uses these new predicates to learn preference rules for the original rules. This process of meta-rule generation can be succesively applied to rules, meta-rules, meta-meta-rules etc. Experiments in a propositional learning framework [Sebag and Schoenauer, 1990] have shown that three iterations succesively improve predictive accuracy, while further steps may decrease performance on unseen examples, possibly through overfitting. Another possibility is to use the meta-rules *instead* of the original rules, which yields promising results too.

4 Plausible Theories

So far we have considered approaches that approximate theories by simply relaxing the constraint that the learned rules have to be correct. Simple, but powerful over-generalizations can serve as useful approximations in many intractable domain theories. However, the qualitative nature of the knowledge learned in the systems of chapter 3 is often kept implicit. In many of them it is hard to distinguish between correctly learned and approximate knowledge.

In this chapter we will consider a different approach to the learning of and with qualitative knowledge: the introduction of explicit language constructs to capture weak inferences and approximations to make the "qualitativeness" of the domain explicit. The work of Collins and Michalski [Collins and Michalski, 1989] gives a formal framework for human plausible reasoning. Although the presented formalism has been implemented in several versions [Dontas and Zemankova, 1988, Baker *et al.*, 1987] it has not yet been directly used in Machine Learning research.[5] Nevertheless, many of the ideas presented in this chapter are directly influenced by or based on ideas of this fundamental work.

Explicitly representing weaker forms of inference should increase understandability and expressive power. Analogously to [Doyle, 1986] we will call this process *qualitization*. But while Doyle restricts his definition of the mapping of a continuous domain description onto a discrete representation, we want to stress that our definition is meant to capture all forms of language constructs that capture some kind of intuitive, plausible reasoning.

It is to be expected that several of the problems that can occur with using approximate rules may as well appear in reasoning with plausible theories. In particular several of the explanation-based learning approaches of section 4.2 are quite similar to approaches of chapter 3. While the research reported there tried to overcome intractability by allowing inconsistent and incomplete domain theories, other authors deal with the problem by extending

[5] Michalski's recent *Inferential Theory of Learning* [Michalski, 1992] views some of the basic ideas of from the standpoint of Machine Learning.

explanation-based schema formation to a powerful generalization algorithm by replacing implication in the domain theory with some weaker forms of inference [Kodratoff and Tecuci, 1989, Widmer, 1992a].

4.1 Qualitative Reasoning about Physical Systems

4.1.1 Qualitization

In the middle of the 80's a new direction of research in qualitative knowledge representation has started: *Qualitative Reasoning about Physical Systems*. The basic idea behind this is to map the continuous physical systems (mostly modeled by differential equation systems) into a discrete representation that captures the qualitative behavior of the systems, but disregards quantitative details. [Doyle, 1986] calls this process *qualitization*. Among the number of qualitative knowledge representation formalisms that have emerged over the years [Weld and deKleer, 1990, AI-, 1984, AI-, 1991] three formalisms have now become "standard":

- Qualitative Process Theory [Forbus, 1984]
- Qualitative Simulation [Kuipers, 1986]
- Reasoning with Confluences [deKleer and Brown, 1984]

All of the above systems support some kind of *qualitative algebra* [Williams, 1988] that allows them to reason with qualitative function descriptions. In physical systems *qualitative differential equations* (QDEs) are very important and thus form central notions of most representation languages. We will shortly introduce the three systems following the more elaborate discussion in [Coiera, 1992]. For a detailed description consultation of the original papers is recommended.

The work described in [deKleer and Brown, 1984] is primarily concerned with modeling the structure of complex objects. The basic philosophy of this approach is that no information about a model's function should be included into the representation. A physical model should be constructed by forming a modular description of the model components and their interactions. The behavior of the entire system can then be simulated and the function of the model supposedly can be derived from it. This is called the *No Function in Structure* (NFIS) principle.

A set of confluences, i.e. QDEs, is used to describe the behavior of device component. Connections to other components allow a simulation of the device model, called *envisionment*. Theoretically the modularity of this approach would allow huge libraries of components to be built up that could be put together to form more complex objects. While this approach has been proven useful in structured domains as e.g. circuitry, it has its difficulties in representing continuously varying systems.

In the Qualitative Process Theory [Forbus, 1984], on the other hand, the central notion is a *process*. Objects consist of a *view* — the processes and their relations to each other — and a set of *influences* that are used to model simple qualitative relationships of functions, such as monotonicity, derivation, addition etc. Views have entry conditions that allow the same object to have different views depending on the context in which the object is used. QPT is a very general approach to qualitative reasoning about physical systems. This of course has the disadvantage of requiring quite some time to specify a model.

A discussion of the relation between the formalisms and their advantages and disadvantages can be found in [Coiera, 1992].

We will discuss a simplified version of the QSIM approach to qualitative reasoning in a little more detail in the next section, because of its simplicity and its wide use for Machine Learning research (e.g. [Coiera, 1989a, Bratko *et al.*, 1991, Varšek, 1992]).

4.1.2 QSIM

Qualitative SIMulation (QSIM) [Kuipers, 1986] is the most recent approach to qualitative reasoning. Its main purpose was to allow simulation — envisionment — of processes. Thus it is mainly concerned with the time varying behavior of qualitative representations of functions which then can be used for simulating the behavior of the system. The main difference between QPT (developed for modeling) and QSIM (developed for simulation) is stressed by the Qualitative Process Compiler (QPC) [Crawford *et al.*, 1990], an algorithm that allows transformation of QPT influences into QSIM constraints.

In QSIM a model of a physical system is described by a set of continuously differentiable, real valued functions of time. These functions are qualitized by mapping their domain onto an ordered, finite set of discrete symbols, like

e.g. `high`, `low` and `medium`, called *landmarks*. In addition to a landmark, the value of a function consists of a symbol indicating the direction of change, either `inc` (increasing), `dec` (decreasing), or `std` (steady).

The model specification basically consists of a set of *qualitative constraints* that must hold between different functions, very similar to QPT's notion of influences. 6 different types of constraints can be used:

`ADD(f,g,h)` $\iff \forall t : f(t) + g(t) = h(t)$

`MULT(f,g,h)` $\iff \forall t : f(t) \cdot g(t) = h(t)$

`MINUS(f,g)` $\iff \forall t : f(t) = -g(t)$

`DERIV(f,g)` $\iff \forall t : \frac{d}{dt} f(t) = g(t)$

`M+(f,g)` $\iff \forall t : f(t)$ is *monotonically increasing* with $g(t)$

`M-(f,g)` $\iff \forall t : f(t)$ is *monotonically decreasing* with $g(t)$

The domain of each function, as already mentioned, is a set of landmarks. A list of real valued *corresponding values* can be used to specify particular points of each of the relations as an additional constraint, but as our interest lies in the qualitative part of the reasoning procedure we will pay no further attention to this possibility.

In the next sections we will first deal with the automated acquisition of qualitative knowledge and then with its use as background knowledge.

4.1.3 Learning of Qualitative Models

An important problem in automated qualitative modeling — referred to as the *system identification problem* — is to find a model of a physical system that is consistent with given examples of its behavior. In section 4.1.1 we have introduced the NFIS principle. Following this principle, many qualitative modeling theories only require to specify the structure of the modeled objects and the behavior of each individual component. The overall behavior of the system can then be inferred from that by simulation.

A natural place for learning in this framework is to induce the qualitative behavior of the components themselves from provided examples. In [Mozetič, 1987a] a system is described which learns rules for the function of

the model components from instances of their behavior. The initial background knowledge consists of a structured model of the heart from the KARDIO project [Bratko *et al.*, 1989] and some instances of component behavior.

Initial data-driven learning generalizes these instances into first-order logic rules. It does so by generalizing a true fact and subsequently specializing it by adding literals and instantiating or unifying variables until no negative instance is covered. This procedure is quite similar to systems of the AQ-family [Michalski, 1980] with the difference that it is using non-recursive PROLOG-like clauses as a representation language. It can thus be considered as an early *Inductive Logic Programming (ILP)* system [Muggleton, 1991, Quinlan, 1990], very similar to the Model Inference System (MIS) [Shapiro, 1981].

As in MIS [Shapiro, 1982], the initial hypotheses can be incrementally debugged. For this purpose the model with the induced functions of the components is simulated and the result is compared with the intended model behavior. In the case of the derivation of a false fact, exactly one hypothesis is blamed and the negated instance of the hypothesis is added as a negative exception, thus preventing future derivations of the false fact. If a true fact cannot be derived, the debugger proposes positive, yet uncovered instances for some component behaviors until the model behavior can be derived. Generating positive exceptions for the rules is a non-trivial task which, due to its extensive use of abstractions [Mozetič, 1987b], is described in more detail in section 5.5. The proposed negative and positive exceptions can be validated by the user. When he thinks that enough exceptions for component behaviors have been accumulated, the incremental learning procedure can be started again to modify the rule set for the components' behavior.

GenModel [Coiera, 1989a] is another early system for automated qualitative modeling. As a simplified version of the QSIM qualitative domain theory is used as background knowledge, the learning algorithm is again quite similar to recent approaches in ILP. Input to the algorithm are the names of the functions that should model the components and several examples for their behavior.

After generating landmarks for each function by simply taking the union of all values occuring in the examples, the most specific set of constraints consistent with the first training instance is generated and then successively generalized. Exhaustive search is used for the generation of this clause, which

is too expensive for bigger models.

In a simple example domain, modeling a qualitative description of the U-tube, GENMODEL ends up with 14 constraints, instead of the correct solution of 6 constraints. This and other experiments show that the system has a tendency to over-specialize. Further examples would be needed to generalize the learned model. This specific-to-general search strategy is dual to Mozetič's general-to-specific search for forming the initial hypotheses as described above.

A more detailed discussion together with a suggestion for the use of negative training examples can be found in chapter 9 of [Coiera, 1989b]. [Coiera, 1993] has pointed out that the clause generation process in GENMODEL is nothing else than constructing the *relative least general generalization* (rlgg) in ILP terminology [Plotkin, 1970, Plotkin, 1971, Buntine, 1988].

A well-known ILP system was then actually used in [Bratko *et al.*, 1991]. GOLEM [Muggleton and Feng, 1990] learns first-order clauses in a very efficient way. A definition of the QSIM theory in first order Horn clauses allows it to be used as background knowledge. Qualitative constraints are expressed by logic predicates, similar to the approach we have used in section 4.1.1. Thus they can be used for learning a description of the target concept. The constraint predicates are compiled into tables of ground facts as needed by GOLEM. GOLEM is able to induce the correct description of a legal state in the U-tube system by using only 10 examples and 5408 ground facts as background knowledge, although the correctness of Golem's description was not immediately obvious, as it introduced a new variable into the description of the model, which would have been impossible with GENMODEL. The application of other ILP systems — FOIL [Quinlan, 1990] and LINUS [Lavrač *et al.*, 1991] — is reported to be less successful on this particular problem.

A very interesting approach to the same problem domain using a *genetic algorithm* can be found in [Varšek, 1992]. Qualitative Models are represented in hierarchical binary trees. Genetic operators — crossover and mutation — change the population of these tree structures and only the fittest models survive. The fitness of a model is estimated by computing a raw fitness function of the model's behavior depending on the percentage of covered positive and on the percentage of negative examples that have been recognized as illegal states. This raw fitness is defined to be the same among similar candidate

models[6] to prevent early convergence and permit the formation of subpopulations that exploit different niches. Thus a very broad search for a global fitness maximum is performed.

Several experiments in simple QSIM models, among them the U-tube, exhibit good learning progress. The algorithm finds five different minimal solutions with 6 constraints in about 20 generations. Background knowledge could be incorporated into this approach by introducing "building blocks" into the trees of the initial population that is currently generated randomly.

4.1.4 Chunking of Qualitative Behavior

The research described in section 4.1.3 tries to induce the behavior of single components in a structured model of a physical system in order to be able to simulate the behavior of the entire system. The approaches in this section assume that a structured model of the physical system is available and the behavior of the various components has already been specified. As simulating the behavior of the system can be quite expensive (see section 2.4), several authors have considered to learn efficient rules for associating the correct system behavior with the input without having to simulate the whole process.

Preliminary work in this area has again been done in the KARDIO project [Bratko *et al.*, 1989]. As we have seen in section 4.1.3 an MIS-like learning algorithm was able to induce and debug component behaviors. When the debugging process has come to an end, i.e. the user has enough confidence into the behavior of the learned qualitative model, it can be used for simulation and problem solving. A natural approach would be to operationalize and generalize every problem solution with an Explanation-Based Generalization algorithm [Mitchell *et al.*, 1986]. However, in the KARDIO project another approach was used [Mozetič, 1986]: The generated qualitative model is simulated on all possible inputs and thus a complete case base of model behavior is obtained. These instances are then used as training examples for generating efficient decision rules with a predecessor of the AQ15 [Michalski *et al.*, 1986] induction algorithm. The induced rules have proved to be not only efficient, but also quite meaningful to human domain experts.

A similar approach has been taken in [Pearce, 1988]. By systematically

[6] Similarity of two models is estimated with the number of examples on which the models agree.

failing all model components, a qualitative model is used to induce rules for diagnosing faults. Having a complete and consistent set of examples — all possible faults have been simulated — the induction algorithm AQR [Clark and Niblett, 1987][7] generates a set of rules that covers 100% of the data. Thus learning is once more used for data compression. A comparison with a handcrafted model for the same task (diagnosing faults in a satellite's power subsystem) showed that the construction of diagnosis rules by induction is cheaper, more precise and easy to verify.

Pearce's method using AQ-type induction, however, was not able to diagnose faults affected by the history of certain components, a type of failure that actually can occur in the satellite's power subsystem.[8] [Feng, 1991] shows that this problem can be solved by using learning algorithms with greater expressive power, i.e. algorithms that can induce first-order logic descriptions of the faults. The author uses the ILP algorithm GOLEM [Muggleton and Feng, 1990] in the task mentioned above. GOLEM succeeds in 29 out of 33 induction experiments. 4 faults could not be detected by induced rules in certain conditions, e.g. a solar panel in eclipse.

4.1.5 Learning with Qualitative Domain Models

Several approaches exist for learning with qualitative background theory. In [Forbus and Gentner, 1986] a theoretical framework is given by Ken Forbus and Dedre Gentner as an attempt to combine *Qualitative Process Theory* [Forbus, 1984] and *Structure Mapping Theory* [Gentner, 1983]. [Forbus and Gentner, 1986] describe a coupling of the two systems: QPT is used to model portions of people's physical knowledge, and SMT is used for reasoning by analogy, i.e. it gives a theory of how a known solution to a problem can be mapped onto a "similar" problem.

Dedre Gentner views analogy as a mere comparison between two structural representation without any functionality in the mapping process [Gentner and Landers, 1985]. This loosely corresponds to the NFIS principle we have discussed in section 4.1.1. The process of comparing two knowledge

[7] AQR is a variant of Michalski's AQ [Michalski, 1980] that is able to deal with conflicting classifications of examples.

[8] In [Pearce, 1988] this problem did not arise as examples for failures that could not be distinguished from a normal state with respect to the observable indicators had been removed from the set of example behaviors used for induction.

structures in order to find relevant similarities to the current situation merely falls out as a map between the relationship between two physical processes. No information about the goal of the mapping or about the problem to solve is involved.[9] Learning occurs by mapping a qualitative explanation of a physical phenomenon onto a new domain.

Falkenhaimer's research on the STRUCTURAL MAPPING ENGINE (SME) [Falkenhainer *et al.*, 1989] and its use in PHINEAS [Falkenhainer, 1990] is an attempt to implement the Structural Mapping Theory developed in [Forbus and Gentner, 1986].

PHINEAS is an implemented integration of Gentner's SMT and Forbus' QPT. Whenever the system is not able to explain an encountered physical process it tries to find a solution by analogical inference. First SME is used to retrieve a previous experience matching the current situation. This prior situation is explained using the domain theory and the explanation is mapped to the new situation with SME. The consistency of the resulting models can be verified by comparing their predictions to the current situation. In [Falkenhainer, 1987] a time based planning system can then be used to extend the consistent theory through further experimentation. When the proposed experiments show that the current theory is inconsistent, it can be refined using model refinement techniques or a new analogy can be looked for in order to find a new explanation. The learned theories are thus evaluated by their ability to predict observed physical phenomena.

This verification-based refinement technique supposedly applies equally well to any form of theory formation in which conjectures of uncertain validity are made. In this respect PHINEAS is quite similar to GENMODEL [Coiera, 1989a] — where inconsistent rules are deleted and a new, correct rule is learned — and to the incremental theory refinement approaches of section 3.2.2.

4.1.6 Directed Dependencies

The monotonicity operators `M+` and `M-` used in QPT and QSIM (see section 4.1.2) are very widely used in qualitative knowledge representations. They are also referred to as *qualitative proportionalities* [Forbus, 1984] or *directed dependencies* [Collins and Michalski, 1989].

[9] For a criticism of this view see [Hammond *et al.*, 1991].

The use of directed dependencies requires an ordering relation on the set of qualitative values. When `M+( f, g)` holds, this means that when the values of `f` move into a certain direction (upwards or downwards), the values of `g` are likely to change into the same direction. So e.g. `M+( PieceAdvantage, ChancesToWin)` specifies that the more pieces a player wins in a chess game, the higher are his chances to win. `PieceAdvantage` as well as `ChancesToWin` are functions over a domain of ordered symbols, such as `PawnUp`, `ExchangeUp` or `RookDown` and $-+$, $\mp$, $=$, $\pm$ or $+-$.[10]

Plausible Inferences — i.e. educated, abstract guesses at why a given proposition is likely to be true — are proposed in [DeJong, 1989]. Two problems appear: Plausible inferences are uncertain and imprecise. The problem of imprecision is solved by a procedure outside of the EBL-scope that watches and remembers the amount of changes in the quantitative values of a qualitative variable and uses them for interpolation in future problems. Uncertainty is decreased by the fact that derivations of explanations from plausible background knowledge by using an existing observation for biasing the search adds credibility to the inferred rule. The interpolation function mentioned above can be used for detecting inconsistencies, especially when smoothness constraints are used in addition to continuity constraints. DeJong's notion of Plausible Inference differs considerably from Collins' Theory of Plausible Reasoning as imprecision is wholely contained in the background knowledge and does not reflect fundamental domain fuzziness.

One of the shortcomings of DeJong's approach is the lack of ability to rate the plausibility of explanations and that it is unable to incrementally modify learned concepts in the face of new evidence. Based on the plausible EBL approach of DeJong, [Widmer, 1992a] uses plausible explanations for better generalizations in plausible proof trees.

Widmer's system uses estimated degrees of plausibility to assess the value of an explanation, quite similarly to the approaches of section 3.5.1. If possible, only generalizations that are safe (i.e. preserve the validity of the inference drawn in the proof tree) will to be considered for generalization. Also several types of explanatory links are less important or less plausible (by definition or by their estimated degree of plausibility) than others, which

[10] The latter five symbols are commonly used in collections of chess games denoting (in that order) decisive advantage for black, minor advantage for black, equal chances, minor advantage for white and decisive advantage for white.

means that generalizing or dropping them is less dangerous. Information from heuristics like these is combined into one value, indicating how likely it is that a generalization of the explanation is justified.

When facing an unexpected behavior the algorithm heuristically decides whether a new example should be used to deduce a new rule or whether an existing rule should be incrementally modified. New rules are learned by using plausible explanation trees which are stored for further use by incremental induction. This process is described in detail in [Widmer, 1991]

While the approaches described above extend Explanation-Based Learning with the use of plausible inferences using qualitative background knowledge, [Clark and Matwin, 1993] present an approach using directed dependencies to guide the search for correct rules in inductive learning. The background knowledge consists of a qualitative model of a physical system represented with a directed graph whose nodes represent numeric parameters and whose arcs represent directed dependencies between the parameters. The inductive learning algorithm CN2 [Clark and Niblett, 1989] is then used to learn rules that are consistent with the qualitative model. The conditions of the learned rules are restricted to be tests on the numeric parameters. A rule is defined to be consistent with the model when it follows a tree in the model graph where each test in the condition corresponds to a node in the graph. The rules predict an increase or decrease of the numeric values correponding tothe leaves of the tree. The restriction on CN2 only to learn rules that are consistent with the qualitative model resulted in a better explainability of the results as well as in an increase predictive accuracy. Constraining the search space by using qualitative background knowledge seems to have a positive effect on the learning behavior as it reduces the danger of learning incorrect rules that by chance fit the sample data.

4.2 Determinations

4.2.1 Introduction

Determinations are an attempt to formalize a piece of common-sense knowledge that is too weak to be captured with logical implication. A determination denotes a dependency between two expressions, without explicitly stating what this dependency is. A well-known example for this is that one's

native language depends on the country where one is born, i.e. the country determines the language.

In the relatively simple chess endgame King and Pawn vs. King (KPK) it might be plausible, even for novices that the position of the white king relative to its pawn might determine the outcome of the game. Nevertheless without further analysis, it is not at all plausible that the king must be in front of his pawn — possibly even obstructing the pawn's path to the eighth rank (where it can promote to a queen, ensuring an easy win) . In fact the mistake of pushing the pawn forward as fast as possible, protecting it with the king from behind, can be frequently found in beginners' games. But when the player realizes that the position of the king relative to his pawn is very relevant to the outcome of the game, the player needs to see only one game where the king in front of the pawn wins the endgame and he will derive the rule "The king has to be in front of the pawn", a rule that is known by the majority of chess players. Many more examples of determinations in common-sense knowledge can be found in [Russell, 1986a].

As we have seen, the mere knowledge that there *is* a functional relationship between the position of king and pawn and the outcome of the game was sufficient to learn a simple, powerful rule. We will know try to formalize this kind of reasoning.

4.2.2 Definition

A logical expression $P(x, y)$ is said to determine another expression $Q(x, z)$ when any two things x that have the same value y on property P also have the same value z on property Q. As determination is a property of predicates, we also can say P determines Q. The important difference to implication is that without knowledge of one or more ground instances of $P(x, y)$ and $Q(x, z)$ nothing can be inferred about the relationship between P and Q. If P implies Q then knowledge about instances of P is sufficient for inferring that Q is also true. If P determines Q however, nothing can be inferred about Q from the knowledge of P only. We need pairs of examples that unify with $P(x, y)$ *and* $Q(x, z)$ to establish the relationship between them. It is easy to see that if P implies Q it also determines Q, but the reverse is not true. Thus logical implication can be viewed as a special case of determination.

[Davies and Russell, 1987] give the following formal definition of determinations:

Definition 4.1 (Determination) *A predicate schema P with sets of free variables x and y* determines *(denoted as $\succ$) a predicate schema Q with sets of free variables x and z when the following holds:*

$$P(x,y) \succ Q(x,z) \Longleftrightarrow \forall y,z[(\exists x P(x,y) \wedge Q(x,z)) \Rightarrow (\forall x P(x,y) \Rightarrow Q(x,z))]$$

This states that P determines Q — with which it shares the variables x — whenever it holds that after seeing ground instances of P and Q that unify the variables x to the same constants any further instantiation of P can be used to derive another instantiation of Q. Note that the variable sets y and z are bound *outside* of the derived implication.

From the chess determination

```
king_pawn_relation( CurrentPosition,  Relation)  ≻
   classification( CurrentPosition,  Class)
```

we can thus derive the following rules after seeing one instance of each of the two cases:

```
classification( CurrentPosition, won) :-
   king_pawn_relation( CurrentPosition, in_front).
classification( CurrentPosition, draw) :-
   king_pawn_relation( CurrentPosition, behind).
```

4.2.3 Learning with Determinations

Determinations have originally been proposed as a "a logical approach to reasoning by analogy" [Davies and Russell, 1987]. Up to this work analogy has mainly used heuristic similarity measures [Russell, 1986b, Carbonell, 1986] for retrieving previous instances and mapping their features to the current situation. By using determinations a reasoning system now has the chance to find appropriate generalizations from single examples, without having the rules it will generate be implicitly specified from the start as most explanation-based systems do.

In subsequent work [Russell, 1987] points out that single-instance generalization as in EBL and reasoning by analogy both can have explanation-based as well as determination-based justifications. For this purpose both — single-instance generalization and analogy — are put in a framework using

determinations [Davies and Russell, 1987]. It is shown that strong domain theories, as used traditionally in EBL, overly constrain learning. For any implicative rule there is a corresponding, less specific determination which, when combined with an instance, yields the same result. The training instance is then needed as a source of information, while in EBL with strong domain theories it merely serves as a focus.

4.2.4 Weaker Relatives of Determinations

Determinations are still relatively strong statements, as after seeing one instance a universal rule is deduced, but if this example were one of the notorious "exceptions to the rule", the wrong rule would be deduced. When a chess novice encounters an endgame that is won by keeping the king behind the own pawn, he might not be aware of the fact that his opponent must have made mistakes. He might thus generalize to the false rule that endgames always can be won by keeping the king behind the pawn.

For this reason many authors use some form of determinations with a weaker semantics. Russell himself calls this form of knowledge *partial determinations* [Russell, 1986a]. Directed dependencies (section 4.1.6) can be viewed as a kind of determination as well, only used for ordered sets of qualitative values.

The system POSTHOC [Pazzani, 1989] uses so-called *influences*. Influences are determinations, that do not necessarily have to be correct. After seeing a training example POSTHOC forms a hypothesis by finding an influence on the target concept and assuming that all factors that influence the target imply it. Thus a generalization from a weak form of inference (influences) to a stronger form (deduction) is performed. Whenever the system goes wrong with one of its rules, it tries to correct the error. In the case of errors of omission[11] a new clause is added that covers the example. In the case of errors of commission[12] POSTHOC tries to specialize one of the clauses by adding a new literal such that the example is no longer covered.

As influences are much weaker statements than determinations, one example is no longer enough to find valid generalizations. POSTHOC needs a variety of them to converge towards the right concept definition. Thus it may be viewed as a version of the Incremental EBL algorithms (see section 3.2.2)

[11] The system does not recognize an object as an instance of the object's definition.

[12] The system erroneously classifies an object to be an instance of the target concept.

where the approximation is guided by a weak form of background knowledge, instead of merely trying to guess the relevant features.

As the ILP algorithm ML-SMART uses its background knowledge "only" as a powerful means for guiding the search for an inductive hypothesis (see section 3.3), the rules specified as background knowledge need not be complete or consistent. For this reason [Bergadano *et al.*, 1989] present an extension of the basic algorithm, where in addition to first-order logic rules, so-called *dependencies* can be specified. Dependencies are nothing else as a method which enables the user to give hints about what predicates might be relevant for a concept definition. Very often, no exact definition of the background knowledge is available, but the user has some idea of which predicates might be used in the target concept description. The inductive discrimination algorithm of ML-SMART uses these hints to constrain the search for a correct hypothesis as described above.

Dependencies may be viewed as a formalism to denote possibly over-specialized rules. Too many conditions are specified in the concept descriptions. Induction is used to select the correct set of predicates to form a consistent rule. The system WHY [Saitta *et al.*, 1991] on the other hand introduces a special 0-arity predicate Ω that can be used to explicitely specify the incompleteness, i.e. the over-generality, of rules. Here a induction process, very similar to the one used in ML-SMART can be used to fill in the gaps.

4.2.5 Learning of Determinations

As we have seen, determinations can be quite useful for analogical and explanation-based learning. Representing less restrictive pieces of knowledge than implications, they are also easier to specify. Nevertheless automatic acquisition of determinations is a topic that still has to be explored.

In [Russell and Grosof, 1987] an example is given where the transitivity relation that holds between determinations is exploited to infer new determinations. In short, if we have $P \succ Q$ and $Q \succ R$ we can infer $P \succ R$.

Chapter 8 of [Russell, 1986a] also attacks the problem of inductively acquiring determinations. The author suggests two methods to learn determinations from either a set of universally quantified expressions or from a set

of instances. The first method inverts the reasoning involved in using determinations as guidelines for analogies. Input are rules as in the example in section 4.2.2, while the output is the determination saying that the relative position of king and pawn determines the outcome of the game.

The more challenging kind of induction is the one from examples. Russell develops an algorithm that induces determinations from facts. A problem that arises here is that in real-world domains it will seldom be the case that a determination will be satisfied by every possible instantiation. In the chess endgame example we have already seen that it is possible to win the endgame with keeping the king behind the pawn, if the opponent also makes mistakes. Russell therefore tries to compute a *degree* of determination, i.e. a value between 0 and 1 indicating the strength of the determination. He thus more or less measures the degree of determination in *influences* similar to those introduced in section 4.2.4. The basic method for this is to randomly select known examples from the joint domain of P and Q and then to compute the proportion of those pairs matching on P that also match on Q [Russell, 1986a]. The resulting certainty factor supposedly measures the percentage of examples satisfying the implication that would result by using the induced determination for an analogy. Russell calls these weaker determinations *partial determinations*.

Of course these certainty factors could as well be used to measure the degree of confidence in the analogy. Using certainty factors could at least express the fact that the inference might be faulty. Calculating useful certainty factors, of course, is a difficult problem. Russell's method relies on a uniform distribution of examples over the space of possible instances. If one randomly generates examples of chess pawn endgames, the majority of them will be one with the king behind the pawn, because of weak play by the opponent. When looking at games of expert players, however, all of them will be following the right rule.

It might seem that inducing determinations from facts is not very useful, as the implications resulting from subsequent analogies could have been induced in the first place, without the detour into determinations. This does not consider the fact that determinations represent an entire family of implications. When seeing only examples of games with king in front of pawn, a determination can be induced that might later on — after seeing an illustrating example — be used for deducing that all other games are lost.

5 Deep Theories and Abstractions

5.1 Introduction

As we have seen in section 2.4, knowledge may be represented at various levels of detail. We have seen in chapters 3 and 4 that knowledge can be approximated by either learning almost correct, but simple rules or by introducing new language constructs that explicitly capture the uncertainty of the learned rules. In both approaches the problem how to deal with inconsistency arises. We have seen several straightforward solutions including incremental refinement of domain theories, re-learning of inconsistent rules or selecting the most plausible explanation. Another approach to deal with this and related problems is to have simple and efficient knowledge to find an approximate solution and then refine it with more elaborate and time-consuming problem-solving methods. This seems to be the case in human chess play, where players usually have very general ideas and plans that guide their search for a move sequence that serves their goal. Chess players usually know "What to do" and have to think a while to find out "How to do it" quite contrary to chess computers, who usually have no idea what they are doing, but are very good in how to do that without making bad mistakes.

5.2 Deep Models

One way to represent these ideas in knowledge based systems is the use of the so-called *deep and shallow models*. Shallow models consist of rules that are directly applicable to the input data and immediately produce an answer.

Deep models can be used to generate shallow level knowledge as has been shown in the KARDIO-project [Bratko *et al.*, 1989] or in [Pearce, 1988]. In Mozetič's work on learning in the KARDIO project described in section 4.1.4 the qualitative background knowledge is organized into a hierarchical, deep knowledge base. Experiments prove that this kind of knowledge representation is much more effective for learning than a flat organization that directly

relates instances with their classifications.

The advantages of shallow knowledge representation is its efficiency, but disadvantages are the inflexibility, as rules typically correspond to a few, highly specialized cases, and the large number of rules that have to be maintained. Deep models on the other hand are highly structured and modular. Reasoning in models of this kind is much more transparent and understandable, but much less efficient as it requires a more sophisticated control structure.

[Klein and Finin, 1987] give an operational definition of the `deeper-than` relation for domain models:

Definition 5.1 (Deepness) *Model M_1 is* deeper *than model M_2 if there exists some implicit knowledge in M_2 which is explicitly represented or computed in M_1.*

Note that according to this definition two models can both be deeper than the other. In this case the two models cannot be compared.

In the next section we will look at a mechanism called *abstraction* that allows to change from a deeper model to a more shallow model and vice versa. Thus shallow, efficient knowledge can be used to generate an approximate solution to a problem, while deeper, more costly knowledge is only used to refine and correct the result.

5.3 Abstractions

Definition 5.1 gives a definition of a deepness-relation between models. Now we need a mechanism that allows us to change between various levels of details. Intuitively, abstraction can be described as the process of mapping a representation of a problem into a new representation which reduces complexity and preserves certain desirable properties like e.g. that a solution in the abstract space must be useful for solving the ground problem as well [Giunchiglia and Walsh, 1989]. Several formal definitions try to capture this intuitive idea. [Plaisted, 1981] defines a class of functions called *abstraction mappings* for this purpose.

Definition 5.2 (Abstraction) *An* abstraction *is a mapping of a clause C onto a set of clauses $f(C)$ such that f — the* abstraction mapping *— has the properties*

1. *If R is a resolvent of S and T then at least one element of $f(R)$ is subsumed by a clause that is the resolvent of two elements of $f(S)$ and $f(T)$.*

2. *If S subsumes T then $f(S)$ subsumes $f(T)$.*

3. $f(\emptyset) = \emptyset$

Plaisted also shows that a function that maps clauses literal by literal is an abstraction mapping if it preserves complements ($f(\neg S) = \neg f(S)$) and instances (if S is an instance of T then $f(S)$ is an instance of $f(T)$).

A serious problem that may arise with Plaisted's formalization of abstraction is the *false proof problem*, i.e. consistent theories might become inconsistent after abstraction.

[Tenenberg, 1987] introduces *restricted predicate mappings*, a subset of abstraction mappings, to deal with this problem. Predicate mappings are functions that map predicate symbols from one first order language to those of another, where they possibly may result in a common symbol, representing a superconcept of the original predicates. The intuition behind restricted predicate mappings is that the interpretation of a predicate in the abstract theory should be the union of the interpretations of each of the predicates in the original theory that map to it. This is obtained by removing all axioms from the original theory that serve to distinguish the relations that are conflated at the abstract level. In addition it is permissible to include mapped positive clauses from the original theory into the abstract theory. More precisely Tenenberg's restricted predicate mappings can be obtained by adding the following condition to definition 5.2:

Definition 5.3 (Restricted Predicate Mapping) *An abstraction mapping is a restricted predicate mapping when the following condition is satisfied:*

4. *$S \in f(T)$ for some T that is part of the axiomatization of the domain theory. Then either S must be a positive clause or every clause R such that $S \in f(R)$ can be proven in the domain theory.*

Interesting in this definition is that it involves the notion of a proof. Thus Tenenberg's approach is a leap from a mere structure-matching abstraction function to a semantic definition.

An important property of abstraction mappings is the *upward-solution property*: Every solution in a problem space has a corresponding solution in an abstracted space. The opposite, however, is desirable in most problem-solving applications: A solution found in an abstract space should be usable in deeper representation levels as well. Tenenberg's restricted predicate mappings achieve this goal by replacing Plaisted's *upward-solution property* with a *downward-solution property*, i.e. each solution at an abstract level corresponds to a solution at a detailed level. Solutions at abstract levels may not exist though, because they may require some of the details that are ignored at that level. [Giunchiglia and Walsh, 1989] show a close relationship between the upward-solution property and the false-proof problem: The authors prove that for every mapping between two theories[13] that has the upward-solution property, there is always a set of consistent axioms whose abstraction is inconsistent, i.e. that they cause the false proof problem.[14]

Abstraction hierarchies have been widely used in Artificial Intelligence research, especially in the fields of problem-solving and planning (see e.g. [Sacerdoti, 1974a], [Sacerdoti, 1974b], [Korf, 1980]). A historical account of the use of abstractions in various domains can be found in [Giunchiglia and Walsh, 1992a]. Our main concern in the next sections will be the use of abstractions in the context of learning.

5.4 Abstraction by Approximation

Knoblock's work on a problem solver that uses hierarchies finally led to the development of an algorithm that allows the system to automatically generate the hierarchies needed for efficient problem-solving [Knoblock, 1989].

[13] They do not restrict themselves to predicate mappings, but prove their results for general functions that map representation languages, axioms and/or inference rules. As most systems use some form of first-order predicate calculus as a representation formalisms, Plaisted's definition given above is sufficient for our purposes. For a more general account of abstractions see [Giunchiglia and Walsh, 1992b].

[14] [Giunchiglia and Walsh, 1989] propose a solution for this problem by introducing an ordering according to relative "weakness" of abstraction spaces. This sequence is then searched from strongest to weakest abstractions to either identify the first abstract theory where the theorem in question is not a theorem (in which case it cannot be a theorem in the ground space either) or to map the proof of the weakest abstraction to the ground space to construct a proof of the analogous theorem.

Knoblock's system ALPINE constructs plans for several domains — among them the Tower of Hanoi puzzle, a robot planning domain and a machine-shop scheduling domain — on increasingly less detailed abstraction levels. In the Tower of Hanoi domain a plan to move the biggest disk is generated on the first level.[15] This plan cannot be executed immediately as in general the disk will lying beneath some other disk. Thus a plan for moving this disk has to be created without violating the prior planning result (movement of the biggest disk). Knoblock calls the latter requirement — that the refinement of an abstract plan leaves the truth value of every literal in the abstract plan unchanged — the *ordered monotonicity property* [Knoblock, 1990b], which is a special case of the downward solution property. The hierarchy of abstraction spaces in ALPINE is formed by removing successive classes of literals, such that each abstraction space is an approximation of the original problem space. The abstraction mapping thus is reminescent of some of the ideas of chapter 3.

ALPINE automatically learns its abstraction hierarchies by generating an ordered graph with the literals used in the STRIPS-like operators [Fikes and Nilsson, 1971] as nodes. The directed edges of a graph are inserted among literals of the add- and delete-lists, and between them and the literals of the precondition of the same operator. A directed edge from one literal to another thus says that the former must be in the same or on a higher level of abstraction as the latter. The strongly connected components of the resulting graph then represent a class of literals that must be in the same abstraction level. The connection between different literal classes can only be one way (otherwise they would be strongly connected) and thus a partial ordering of literal classes is found — the abstraction hierarchy. Experimental results prove the problem-solving using the learned abstractions to solve more problems than versions using EBL or PRODIGY.

PABLO [Christensen, 1990] is — as a hierarchical planner — quite similar to ALPINE. It also learns a hierarchy of abstraction spaces, but uses a different strategy. It performs a regression of goals through the operators, thus being able to determine the number of steps required to achieve each goal. The system then solves a problem in successive abstraction levels by first working on the parts of the problem that require the most steps. ALPINE on the other hand tries to first solve the parts of the problem that can be left unchanged

[15] For more details on the planning aspects of ALPINE see [Knoblock, 1990a].

while the remaining subproblems are solved.

In [Unruh and Rosenbloom, 1989] possibilities for the use of abstraction in the SOAR system [Laird *et al.*, 1987] are suggested. As ALPINE and PABLO, SOAR uses a method for abstracting the search space by simply removing some aspects of the problem. While in the systems discussed above this is done a priori in a static way, in SOAR abstraction occurs dynamically during problem solving. Whenever SOAR encounters and *impasse* during search, it usually resolves this by proposing a new subgoal [Laird *et al.*, 1987]. [Unruh and Rosenbloom, 1989] suggest that instead of further problem solving in another level of subgoals the impasse may as well be removed by ignoring the sub-goal. This is very similar to the approach taken in [Keller, 1988] (see section 3.5.1). Learning occurs in SOAR by converting the subgoal-based search into shallow rules [Rosenbloom and Newell, 1986, Rosenbloom and Laird, 1986]. Thus abstracted subgoals are simply ignored and approximate search control rules are learned.[16]. This approximation mechanism naturally extends to form an abstraction hierarchy out of the dynamic subgoal hierarchy that SOAR creates during problem solving. When an operator is finally chosen after a search using abstraction, its preconditions have to be fulfilled before the operator can actually be applied. Thus the formerly abstracted subgoal has to be searched now. During the search for an operator that establishes this goal, abstraction might be useful again, the new abstraction being on a more detailed level than the original one. This incremental deepening continues until a complete plan is generated.

The proposed method is a very general technique that can be applied without domain specific abstraction knowledge. However, additional background knowledge can be used to guide the abstraction (e.g. to choose the subgoals that should be abstracted). Experimental results in a robot toy domain and a computer configuration domain show that abstraction helps more the more search is required for a solution of the problem.

[16] Abstraction in SOAR is limited to search control as this ensures that unjustified abstractions will not lead to incorrect, but only to inefficient behavior

5.5 Abstraction by Enhancing the Representation Language

While the approaches to abstraction of the last section made use of rule approximation techniques similar to those of chapter 3, this section will deal with approaches that simplify learning by dynamically changing to a more expressive representation language.

Qualitization as introduced in chapter 4 is one form of simplifying learnning in many domains. In [Mozetič, 1987b] an approach to automated qualitative modeling (see section 4.1.3) is extended with the use of abstractions. In section 4.1.3 we have described how initial hypotheses for the component behavior are formed. The model is then simulated on new instances. Whenever an expectation failure occurs, a debugging module similar to MIS [Shapiro, 1982] is invoked. In the case of incomplete hypotheses — a true fact is not covered — a new rule is added to the current hypothesis. While generating this new rule, the search space is reduced by considering an abstraction of the uncovered instance, as behavior at the specific level in the model must be consistent with the behavior at the abstract level, i.e. the *upward solution property* must hold in the abstraction hierarchy. So the basic idea is that instead of querying the user for possible values of a variable as it is the case in MIS, appropriate values can be automatically generated by abstracting the expression and considering only specializations of the more abstract concept.

For this purpose the qualitative model of the heart that is used as background knowledge is represented at several layers of detail. Abstractions can be formed in one of the following ways:

- Constants of the same type[17] can be collapsed into a single, abstract value, thus forming a hierarchy of values.
- Functions can be renamed and some (or all) of their arguments deleted.
- Predicates can be renamed and some of their arguments deleted.

Thus a hierarchy of representation languages can be formed that will be used for model refinement.

[17] Variables and constants are associated with types, such that variables of a certain type can only be instantiated by constants of the same type.

In subsequent work [Mozetič and Holzbaur, 1991] this abstraction mechanism is put into an explanation-based learning framework. In classical EBL non-operational predicates are unfolded in the generalized proof tree until the concept is defined in operational predicates only [Mitchell *et al.*, 1986]. The authors extend this mechanism by adding the abstraction methods used above. Concrete instantiations of these abstraction schemas can be specified by the user and will be applied to all literals in the proof tree. This mechanism allows abstraction to occur during explanation-based generalization. Classical EBL falls out as a special case of this framework, as the operationality criterion can be formulated by adding a predicate abstraction operator for each operational predicate that renames the predicate to itself and deletes none of the arguments.

In [Drastal *et al.*, 1989] abstraction is used to bias induction in an attribute-value domain language. A domain theory defining more abstract descriptors in terms of simpler ones is used for a simple kind of forward chaining performed on each instance until a set of descriptors is reached that cannot be further abstracted. The set of all such descriptors is used as the language in which the target concept will be induced.

[Giordana and Saitta, 1990] use this idea in the inductive relational learning system ML-SMART [Bergadano and Giordana, 1988]. Here a first-order logic abstraction theory quite similar to [Mozetič and Holzbaur, 1991] can be specified. In the first abstraction phase ML-SMART generates instance representations at different levels of abstraction as an input to the inductive learning algorithm. This differs from the approach of [Drastal *et al.*, 1989] where a set of maximally abstracted descriptors is used. Various experiments with induction on different abstraction levels confirmed that using the most abstract representation level minimizes the combined costs of abstraction and induction. Besides, the results achieved at the most abstract level appeared to be most understandable for human domain experts.

[Giordana *et al.*, 1991] put this work into the ILP framework of *Inverse Resolution* [Muggleton and Buntine, 1988]. A non-generalizing absorption operator, a similar variant of inter-construction, and an additional mechanism, term abstraction, are introduced for this purpose. The latter is a combination of absorption and intra-construction and serves the purpose of defining compound terms, similar to abstract data types in programming.

The approaches discussed above show that the abstraction mechanism can be axiomatized by means of a theory, and abstraction itself can be obtained by deduction. This approach is further pursued in the system PLACE described in [Flann, 1989]. The author observes several difficulties with common approaches to abstraction via approximation (see section 5.4) in solving formation problems[18]. Flann claims that abstraction by approximation is ineffective in formation problems, as it is very often the case that the reintroduction of a constraint that has been abstracted away interacts with the results of the problem-solving on the abstract level. Thus the results obtained at the approximate, abstract level are very often invalid at the specific level, i.e. the ordered monotonicity property is violated. Removing pieces from a chess position can cause drastic changes in the evaluation of this position. We have discussed this problem in section 2.5 and come to the conclusion that although abstract concepts like a "fork" cannot be defined precisely, an approximation with a simple visual pattern as human chess players apparently use can be a useful guide for the search for good moves.

Flann attacks this problem along similar lines, although he stresses that the abstractions his program finds are not approximations: PLACE is able to recognize many typical, abstract concepts (e.g. `my-king-in-check`) and associate plans and goals with them. The goals are completely instantiated and highly specialized, thus constraining the search. The use of abstract concepts reduces the search as well, as PLACE does not have to consider all moves in a position, but reasons with abstract operators (e.g. `move-my-king-out-of-check`).

The problem of goal interaction is dealt with through a process called *visualization*. PLACE looks for operators that maintain, destroy or achieve a goal. Complex expressions for the achievement of multiple goals are analyzed and compiled by doing an exhaustive case analysis [Flann, 1990]. This analysis is able to generate geometrical constraints that can be used as a recognition pattern for the abstract concept.

[18] Formation problems differ from derivation problems in that the goal is not specified in the same structural description language as the initial state, but through a functional description. Typical formation problems are optimization problems. Flann views chess as a formation problem with the goal to optimize the outcome of the game.

6 Multistrategy Learning

We have seen in section 2.7 that knowledge can be acquired and used in a variety of ways. Initial research in Machine Learning has mostly concentrated on acquiring powerful methods for each principle learning method separately. Nowadays fairly powerful algorithms for learning by induction, deduction or analogy exist. The logical next step is therefore to try to integrate several different methods into a single system. Research on *Multistrategy Learning* has been quite active for several years.

As we are mainly concerned with the underlying knowledge representation and not so much with the learning strategies, a complete summary would be beyond the scope of this paper. We will be content to give a few references and to shortly introduce a prototypical system, MTL, that includes weak, qualitative forms of inferences.

Multistrategy Learning has been mostly investigated in the context of integrating inductive and deductive learning methods. Well-known systems that enhance Explanation-Based Learning methods with an inductive learning component are UNIMEM [Lebowitz, 1986] and OCCAM [Pazzani, 1990]. Other approaches can be found in [Flann and Dietterich, 1989] and [Widmer, 1989b]. In Inductive Logic Programming, the systems FOCL [Pazzani and Kibler, 1992] and ML-SMART [Bergadano and Giordana, 1988] try to integrate background knowledge into inductive learning methods.

DISCIPLE [Kodratoff and Tecuci, 1987] is another integration of inductive and deductive learning methods. In subsequent work learning by analogy was added as an additional means of dealing with imperfect domain theories [Tecuci and Kodratoff, 1990]. From this work the Multistrategy Task-adaptive Learning framework (MTL) [Tecuci and Michalski, 1991] has emerged.

MTL is a Multistrategy approach to learning in the Plausible Reasoning framework of [Collins and Michalski, 1989]. The approach is adaptive in the sense that it is able to apply different learning strategies for different learning

tasks. When provided with some input information the system builds up a Plausible Justification Tree, a structure quite similar to a proof tree (as used in EBL), but allowing different kinds of inferences — deduction, induction, abduction and analogy — to occur at the edges of the tree. The inferences can be formalized by using constructs from the Plausible Reasoning theory by [Collins and Michalski, 1989], as e.g. determinations (see section 4.2). During the construction and the following generalization of the tree, different kinds of new knowledge can be inferred: new concept definitions, new rules, new facts, abstractions etc. In certain base cases the MTL method reduces to EBL, abductive learning, empirical induction or analogical reasoning. Further research has to be done to allow inconsistent information in the knowledge base.

Other systems try to integrate Explanation-Based Learning and Neural Nets [Shavlik and Towell, 1989] or EBL and Case-Based Learning [Wilkins, 1990]. For more on multistrategy learning systems and a more detailed account of the general framework see [Michalski and Tecuci, 1992].

7 Conclusion

The main purpose of this paper was to analyze the important role that qualitative knowledge in various forms can play in Machine Learning research and give a review of previous research in this field. The need for simple, approximate, noise- and error-tolerant knowledge representations has been stressed not only for Machine Learning. Various approaches towards the use of qualitative knowledge for applications in Machine Learning have been discussed in this paper. The static representation of domain knowledge can be approximated by explaining only a subset of the domain (chapter 3) or by finding less detailed, but more understandable and simpler representation formalisms (chapter 4). Both methods can be used to support abstraction, i.e. dynamically changing from one representation level to a more approximate one in order to allow more efficient problem solving (chapter 5). Although in all areas several papers have been published (we have tried to include the most important ones, but could not have been complete) there is still a lot to be done.

The relatively new area of *Inductive Logic Programming* now allows to efficiently induce concept descriptions formulated in first-order logic (see e.g. [Muggleton, 1991] or [Quinlan, 1990]). This gain in expressiveness over traditional propositional learning algorithms has already contributed significantly to learning in qualitative domain theories [Giordana *et al.*, 1991, Bratko *et al.*, 1991, Feng, 1991, Coiera, 1993].

The logical next step is to further extend the representational power of the induced concepts by allowing qualitative language constructs or by relaxing the correctness constraint in order to generate more efficient rules. Preliminary research in [Giordana and Saitta, 1990] and [Drastal *et al.*, 1989] has suggested that finding an appropriate level of abstraction for the representation of the target concept can facilitate learning.

The *false proof problem* is an important issue when using abstractions for more efficient problem-solving behavior. Many systems try to solve this problem by restricting the representation languages in a way that does not al-

low the problem to arise. The *downward solution property* [Tenenberg, 1987] or the *ordered monotonicity property* [Knoblock, 1990b] are examples of this approach (see section 5.2). But to gain even more benefit from abstraction techniques, these constraints on the representation language and/or the abstraction mapping should be relaxed and other methods for dealing with this problem should be developed. Research on using rule approximations as reported in [Fawcett, 1989, Knoblock, 1989, Keller, 1988, Cohen, 1990, Tadepalli, 1989] and on qualitative extensions of the representation language [DeJong, 1989, Widmer, 1992a, Mozetič and Holzbaur, 1991, Bennett, 1989] in traditional Explanation-Based Learning environments shows that a tradeoff between correctness and efficiency is an important issue. Applications of this finding to induction are still waiting to be investigated.

Early research in Machine Learning has mostly concentrated on developing learning algorithms for inductive or deductive generalization. The insight that each of the generalization methods has their specific strengths and that a learning system should exploit different strategies dependent on the current learning situation, has led to the development of a theory for Multistrategy Learning (chapter 6). Similarly research is currently concentrated on enhancing their learning systems with one of the discussed methods of introducing qualitative knowledge. Each of the knowledge representation methods discussed in this paper has their strengths and weaknesses and a powerful learning system should be able to find the appropriate method by itself.

References

[AGA, 1990] *Working Notes of the AAAI Workshop on Automatic Generation of Approximations and Abstractions*, Boston, Massachusettes, 1990.

[AI-, 1984] *Artificial Intelligence, 24*, 1984. Special Volume: Qualitative Reasoning About Physical Systems.

[AI-, 1991] *Artificial Intelligence, 51*, 1991. Special Volume: Qualitative Reasoning About Physical Systems II.

[Angluin and Laird, 1988] D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[Baker *et al.*, 1987] Michelle Baker, Mark H. Burstein, and Allan M. Collins. Implementing a model of human plausible reasoning. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 185–188, Milano, Italy, 1987.

[Bennett, 1989] Scott W. Bennett. Learning approximate plans for use in the real world. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 224–228, Ithaca, New York, 1989.

[Bergadano and Giordana, 1988] F. Bergadano and A. Giordana. A knowledge intensive approach to concept induction. In *Proceedings of the 5th International Conference on Machine Learning*, pages 305–317, Ann Arbor, Michigan, 1988.

[Bergadano *et al.*, 1989] F. Bergadano, A. Giordana, and S. Ponsero. Deduction in top-down inductive learning. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 23–25, Ithaca, New York, 1989.

[Berliner and Campbell, 1984] Hans Berliner and Murray Campbell. Using chunking to solve chess pawn endgames. *Artificial Intelligence*, 23:97–120, 1984.

[Blumer *et al.*, 1987] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

[Bratko *et al.*, 1989] I. Bratko, I. Mozetič, and N. Lavrač. *KARDIO: a Study in Deep and Qualitative Knowledge for Expert Systems*. MIT press, 1989.

[Bratko *et al.*, 1991] Ivan Bratko, Stephen Muggleton, and Alen Varšek. Learning qualitative models of dynamic systems. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 385–388, Evanston, Illinois, 1991.

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

[Carbonell, 1986] Jaime G. Carbonell. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach, Vol. II*, pages 371–392. Morgan Kaufmann, Los Altos, California, 1986.

[Chase *et al.*, 1989] Melissa P. Chase, Monte Zweben, Richard L. Piazza, John D. Burger, Paul P. Maglio, and Haym Hirsh. Approximating learned search control knowledge. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 218–220, Ithaca, New York, 1989.

[Chien, 1989] Steve A. Chien. Using and refining simplifications: Explanation-based learning of plans in intractable domains. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 590–595, 1989.

[Christensen, 1990] Jens Christensen. A hierarchical planner that creates its own hierarchies. In *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 1004–1009, Boston, MA, 1990.

[Church and Church, 1983] Russell M. Church and Kenneth W. Church. Plans, goals, and search strategies for the selection of a move in chess. In *Chess Skill in Man and Machine*, chapter 6, pages 131–156. Springer-Verlag, 2 edition, 1983.

[Clark and Matwin, 1993] Peter Clark and Stan Matwin. Using qualitative models to guide inductive learning. In *Proceedings of the 10th International Conference on Machine Learning*, Amherst, Massachusetts, 1993. Submitted.

[Clark and Niblett, 1987] Peter Clark and Tim Niblett. Induction in noisy domains. In Ivan Bratko and N. Lavrač, editors, *Progress in Machine Learning*, Wilmslow, UK, 1987. Sigma Press.

[Clark and Niblett, 1989] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[Cohen, 1990] William W. Cohen. Learning approximate control rules of high utility. In *Proceedings of the 7th International Conference on Machine Learning*, pages 268–276, Austin, Texas, 1990.

[Cohen, 1992] William W. Cohen. Abductive explanation-based learning: A solution to the multiple inconsistent explanation problem. *Machine Learning*, 8:167–219, 1992.

[Coiera, 1989a] Enrico Coiera. Generating qualitative models from example behaviours. DCS Report 8901, School of Electr. Eng. and Computer Sc., Univ. of New South Wales, Sydney, Australia, 1989.

[Coiera, 1989b] Enrico Coiera. *Reasoning with Qualitative Disease Histories for Diagnostic Patient Monitoring.* PhD thesis, Department of Computer Science, University of New South Wales, 1989.

[Coiera, 1992] Enrico Coiera. The qualitative representation of physical systems. *The Knowledge Engineering Review*, 7(1):55–77, 1992.

[Coiera, 1993] Enrico Coiera. Qualitative superposition of unmodelled systems. Submitted to IJCAI-93, 1993.

[Collins and Michalski, 1989] Allan Collins and Ryszard S. Michalski. The logic of plausible reasoning: A core theory. *Cognitive Science*, 13:1–49, 1989.

[Crawford *et al.*, 1990] J. Crawford, A. Farqhuar, and B. Kuipers. QPC: A compiler from physical models into qualitative differential equations. In *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 365–372, Boston, MA, 1990.

[Danyluk, 1987] Andrea Pohoreckyj Danyluk. The use of explanations for similarity-based learning. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, Milano, Italy, 1987.

[Danyluk, 1989] Andrea Pohoreckyj Danyluk. Finding new rules for incomplete theories: Explicit biases for induction with contextual information. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 34–36, Ithaca, New York, 1989.

[Davies and Russell, 1987] T. R. Davies and S. J. Russell. A logical approach to reasoning by analogy. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 264–270, Milano, Italy, 1987.

[deGroot, 1965] Adriaan D. deGroot. *Thought and Choice in Chess*. Mouton, The Hague, 1965.

[DeJong, 1989] G. DeJong. Explanation-based learning with plausible inferencing. In *Proceedings of the 4th European Working Session on Learning*, pages 1–10, Montpellier, France, 1989.

[deKleer and Brown, 1984] J. deKleer and J. S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

[Dontas and Zemankova, 1988] K. Dontas and M. Zemankova. APPLAUS: An implementation of the collins-michalski theory of plausible reasoning. In *Proceedings of the 3rd International Symposium on Methodologies fo Intelligent Systems*, Torino, Italy, 1988.

[Doyle, 1986] R. J. Doyle. Constructing and refining causal explanations from an inconsistent domain theory. In *Proceedings of the 5th National Conference on Artificial Intelligence*, Philadelphia, PA, 1986.

[Drastal *et al.*, 1989] G. Drastal, G. Czako, and S. Raatz. Induction in an abstraction space. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 708–712, Detroit, Michigan, 1989.

[Ellman, 1988] Tom Ellman. Approximate theory formation: An explanation-based approach. In *Proceedings of the 7th National Conference on Artificial Intelligence*, pages 570–574, Minneapolis, Minnesota, 1988.

[Falkenhainer *et al.*, 1989] Brian Falkenhainer, Kenneth D. Forbus, and Dedre Genter. The structure mapping engine: Algorithm and examples. *Artificial Intelligence*, 41(1), 1989.

[Falkenhainer, 1987] Brian Falkenhainer. Scientific theory formation through analogical inference. In *Proceedings of the 4th International Workshop on Machine Learning*, pages 218–229, Irvine, California, 1987.

[Falkenhainer, 1990] Brian Falkenhainer. A unified approach to explanation and theory formation. In J. Shrager and P. Langley, editors, *Computational Models of Discovery and Theory Formation*. Morgan Kaufmann, Los Altos, California, 1990.

[Fawcett, 1989] Tom E. Fawcett. Learning from plausible explanations. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 37–39, Ithaca, New York, 1989.

[Feng, 1991] Cao Feng. Inducing temporal fault diagnostic rules from a qualitative model. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 403–406, Evanston, Illinois, 1991.

[Fikes and Nilsson, 1971] R. E. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

[Flann and Dietterich, 1989] Nicholas S. Flann and Thomas G. Dietterich. A study of explanation-based methods for inductive learning. *Machine Learning*, 4:187–266, 1989.

[Flann, 1989] Nicholas S. Flann. Learning appropriate abstractions for planning in formation problems. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 235–239, Ithaca, New York, 1989.

[Flann, 1990] Nicholas S. Flann. Applying abstraction and simplification to learn in intractable domains. In *Proceedings of the 7th International Conference on Machine Learning*, pages 277–285, Austin, Texas, 1990.

[Forbus and Gentner, 1986] Kenneth D. Forbus and Dedre Gentner. Learning physical domains: Toward a theoretical framework. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach, Vol. II*, chapter 12, pages 311–348. Morgan Kaufmann, Los Altos, California, 1986.

[Forbus, 1984] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–169, 1984.

[Gentner and Landers, 1985] Derdre Gentner and R. Landers. Analogical reminding: A good match is hard to find. In *Proceedings of the International Conference on Systems, Man and Cybernetics*, pages 76–79, Tucson, Arizona, 1985.

[Gentner, 1983] Dedre Gentner. Structure-mapping: A theoretical framework fo analogy. *Cognitive Science*, 7(2):155–170, 1983.

[Giordana and Saitta, 1990] A. Giordana and L. Saitta. Abstraction: A general framework for learning. In *Working Notes of the AAAI Workshop on Automatic Generation of Approximations and Abstractions*, pages 245–256, Boston, Massachusettes, 1990.

[Giordana *et al.*, 1991] A. Giordana, L. Saitta, and D. Roverso. Abstracting concepts with inverse resolution. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 142–146, Evanston, Illinois, 1991.

[Giunchiglia and Walsh, 1989] Fausto Giunchiglia and Toby Walsh. Abstract theorem proving. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 372–377, 1989.

[Giunchiglia and Walsh, 1992a] Fausto Giunchiglia and Toby Walsh. Theories of abstraction: A historical perspective. In *AAAI-92 Workshop on Approximation and Abstraction of Computational Theories*, San José, California, 1992.

[Giunchiglia and Walsh, 1992b] Fausto Giunchiglia and Toby Walsh. A theory of abstraction. *Artificial Intelligence*, pages 323–389, 1992.

[Hammond *et al.*, 1991] Kristian J. Hammond, Colleen M. Seifert, and Kenneth C. Gray. Functionality in analogical transfer: A hard match is good to find. *The Journal of the Learning Sciences*, 1(2):111–152, 1991.

[Hammond, 1990] Kristian J. Hammond. Explaining and repairing plans that fail. *Artificial Intelligence*, 45(1–2), 1990.

[Hsu *et al.*, 1990] F.-h. Hsu, T. S. Anantharaman, M. S. Campbell, and A. Nowatzyk. Deep thought. In T. Anthony Marsland and Jonathan Schaeffer, editors, *Computers, Chess, and Cognition*, pages 55–78. Springer-Verlag, New York, 1990.

[Hsu, 1987] F.-h. Hsu. A two-million moves/s CMOS single-chip chess move generator. *IEEE Journal of Solid-State Circuits*, 22(5):841–846, 1987.

[Keller, 1987] Richard M. Keller. Concept learning in context. In *Proceedings of the 4th International Workshop on Machine Learning*, Irvine, California, 1987.

[Keller, 1988] Richard M. Keller. Learning approximate concept descriptions. Technical Report KSL-88-57, Stanford University, Knowledge Systems Laboratory, Stanford, California, 1988. Reprinted in [AGA, 1990].

[Klein and Finin, 1987] David Klein and Tim Finin. What's in a deep model? A characterization of knowledge depth in intelligent safety systems. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, Milano, Italy, 1987.

[Knoblock, 1989] Craig A. Knoblock. Learning hierarchies of abstraction spaces. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 241–245, Ithaca, New York, 1989.

[Knoblock, 1990a] Craig A. Knoblock. Abstracting the tower of hanoi. In *Working Notes of the AAAI Workshop on Automatic Generation of Approximations and Abstractions*, pages 13–23, Boston, Massachusettes, 1990.

[Knoblock, 1990b] Craig A. Knoblock. Learning abstraction hierarchies for problem solving. In *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 923–928, Boston, MA, 1990.

[Knuth and Moore, 1975] Donald E. Knuth and R. W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, 1975.

[Kodratoff and Tecuci, 1987] Yves Kodratoff and Gheorghe Tecuci. DISCIPLE-1: Interactive apprentice system in weak theory fields. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 271–273, Milano, Italy, 1987.

[Kodratoff and Tecuci, 1989] Yves Kodratoff and Gheorghe D. Tecuci. The central role of explanations in DISCIPLE. In Katharina Morik, editor, *Knowledge Representation and Organization in Machine Learning*, pages 135–147. Springer-Verlag, Berlin, 1989.

[Korf, 1980] Richard E. Korf. Toward a model of representation changes. *Artificial Intelligence*, 14:41–78, 1980.

[Kuipers, 1986] B. J. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.

[Laird *et al.*, 1987] J. E. Laird, A. Newell, and Paul S. Rosenbloom. SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[Lavrač *et al.*, 1991] N. Lavrač, S. Džeroski, and M. Grobelnik. Learning nonrecursive definitions of relations with linus. In *Proceedings of the European Working Session on Learning*, Porto, Portugal, 1991.

[Lebowitz, 1986] M. Lebowitz. Integrated learning: Controlling explanation. *Cognitive Science*, 10(2):219–240, 1986.

[Lenat and Feigenbaum, 1991] D. B. Lenat and E. A. Feigenbaum. On the thresholds of knowledge. *Artificial Intelligence*, 47:185–250, 1991.

[Levy, 1986] David N. Levy, editor. *Computer Chess Compendium*. Batsford Ltd., London, 1986.

[Marsland and Schaeffer, 1990] T. Anthony Marsland and Jonathan Schaeffer, editors. *Computers, Chess, and Cognition.* Springer-Verlag, New York, 1990.

[McCarthy, 1990] John McCarthy. Chess as the drosophila of AI. In T. Anthony Marsland and Jonathan Schaeffer, editors, *Computers, Chess, and Cognition*, pages 227–237. Springer-Verlag, New York, 1990.

[Michalski and Tecuci, 1992] Ryszard S. Michalski and Gheorghe D. Tecuci, editors. *Machine Learning: A Multistrategy Approach, Vol. IV.* Morgan Kaufmann, San Mateo, CA, 1992.

[Michalski *et al.*, 1986] R. S. Michalski, I. Mozetič, J. Hong, and N. Lavrač. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of the 5th National Conference on Artificial Intelligence*, pages 1041–1045, Philadelphia, PA, 1986.

[Michalski, 1980] Ryszard S. Michalski. Pattern recognition and rule-guided inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:349–361, 1980.

[Michalski, 1983] Ryszard S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20, 1983.

[Michalski, 1992] Ryszard S. Michalski. Inferential theory of learning: Developing foundations for multistrategy learning. In Ryszard S. Michalski and Gheorghe D. Tecuci, editors, *Machine Learning: A Multistrategy Approach, Vol. IV.* Morgan Kaufmann, San Mateo, CA, 1992.

[Minton, 1984] Steve Minton. Constraint-based generalization: Learning game-playing plans from single examples. In *Proceedings AAAI-84*, pages 251–254, Austin, Texas, 1984.

[Minton, 1990] Steve Minton. Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42:363–392, 1990.

[Mitchell *et al.*, 1986] T. M. Mitchell, R. M. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

[Mostow *et al.*, 1990] Jack Mostow, Thomas Ellman, and Armand Prieditis. A unified transformational model for discovering heuristics by idealizing intractable problems. In *Working Notes of the AAAI Workshop on Automatic*

Generation of Approximations and Abstractions, pages 290–301, Boston, Massachusettes, 1990.

[Mozetič and Holzbaur, 1991] Igor Mozetič and C. Holzbaur. Extending explanation-based generalization by abstraction operators. In *Proceedings of the 5th European Working Session on Learning*, pages 282–297, Porto, Portugal, 1991.

[Mozetič, 1986] Igor Mozetič. Knowledge extraction through learning from examples. In Tom M. Mitchell, Jaime G. Carbonell, and Ryszard S. Michalski, editors, *Machine Learning: A Guide to Current Research*, pages 227–231. Kluwer Academic Publishers, 1986.

[Mozetič, 1987a] Igor Mozetič. Learning of qualitative models. In *Progress in Machine Learning*. Sigma Press, Wilmslow, England, 1987.

[Mozetič, 1987b] Igor Mozetič. The role of abstractions in learning qualitative models. In *Proceedings of the 4th International Workshop on Machine Learning*, Irvine, California, 1987.

[Muggleton and Buntine, 1988] Stephen H. Muggleton and Wray L. Buntine. Machine invention of first-order predicates by inverting resolution. In *Proceedings of the 5th International Conference on Machine Learning*, pages 339–352, 1988.

[Muggleton and Feng, 1990] Stephen H. Muggleton and Cao Feng. Efficient induction of logic programs. In *Proceedings of the 1st Conference on Algorithmic Learning Theory*, pages 1–14, Tokyo, Japan, 1990.

[Muggleton, 1991] Stephen H. Muggleton. Inductive logic programming. *New Generation Computing*, 8:295–318, 1991.

[Newell *et al.*, 1960] A. Newell, J. Shaw, and H. Simon. Report on a general problem-solving program for a computer. In *Proceedings of the International Conference on Information Processing*, UNESCO, Paris, 1960.

[Pazzani and Kibler, 1992] Micheal Pazzani and Dennis Kibler. The utility of knowledge in inductive learning. *Machine Learning*, 9:57–94, 1992.

[Pazzani, 1988] M. Pazzani. Selecting the best explanation for explanation-based learning. In *Proceedings of the 1988 Spring Symposium on Explanation-based Learning*, pages 165–169, Stanford University, California, 1988.

[Pazzani, 1989] Michael J. Pazzani. Explanation-based learning with weak domain theories. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 72–74, Ithaca, New York, 1989.

[Pazzani, 1990] M. J. Pazzani. Integrating explanation-based and empirical learning methods in OCCAM. In *Proceedings of the 3rd European Working Session on Learning*, pages 147–166, Glasgow, Scotland, 1990.

[Pearce, 1988] D. A. Pearce. The induction of fault diagnosis systems from qualitative models. In *Proceedings of the 7th National Conference on Artificial Intelligence*, pages 353–357, Minneapolis, Minnesota, 1988.

[Plaisted, 1981] D. Plaisted. Theorem proving with abstraction. *Artificial Intelligence*, 16:47–108, 1981.

[Plotkin, 1970] G. D. Plotkin. A note on inductive generalisation. In B. Meltzer and Donald Michie, editors, *Machine Intelligence 5*, pages 153–163. Elsevier North-Holland, New York, 1970.

[Plotkin, 1971] G. D. Plotkin. A further note on inductive generalisation. In B. Meltzer and Donald Michie, editors, *Machine Intelligence 6*, pages 101–124. Elsevier North-Holland, New York, 1971.

[Quinlan, 1990] John Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[Rajamoney and DeJong, 1987] Shankar Rajamoney and Gerald F. DeJong. The classification, detection and handling of imperfect theory problems. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 205–207, Milano, Italy, 1987.

[Roos, 1992] Nico Roos. A logic for reasoning with inconsistent knowledge. *Artificial Intelligence*, 57:69–103, 1992.

[Rosenbloom and Laird, 1986] Paul S. Rosenbloom and J. E. Laird. Mapping explanation-based generalization onto SOAR. In *Proceedings of the 5th National Conference on Artificial Intelligence*, Proceedings of the 5th National Conference on Artificial Intelligence, 1986.

[Rosenbloom and Newell, 1986] Paul S. Rosenbloom and Allen Newell. The chunking of goal hierarchies: A generalized model of practice. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach, Vol. II*, pages 247–288. Morgan Kaufmann, Los Altos, California, 1986.

[Russell and Grosof, 1987] S. J. Russell and B. N. Grosof. A declarative approach to bias in concept learning. In *Proceedings of the 6th National Conference on Artificial Intelligence*, 1987.

[Russell, 1986a] S. J. Russell. *Analogical and Inductive Reasoning.* PhD thesis, Stanford University, 1986.

[Russell, 1986b] S. J. Russell. A quantitative analysis of analogy by similarity. In *Proceedings of the 5th National Conference on Artificial Intelligence*, Philadelphia, PA, 1986.

[Russell, 1987] S. J. Russell. Analogy and single-instance generalization. In *Proceedings of the 4th International Workshop on Machine Learning*, Irvine, California, 1987.

[Sacerdoti, 1974a] Earl D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2):115–135, 1974.

[Sacerdoti, 1974b] Earl D. Sacerdoti. *A Structure for Plans and Behavior.* American Elsevier, New York, 1974.

[Saitta *et al.*, 1991] L. Saitta, M. Botta, S. Ravotto, and S. Sperotto. Improving learning by using deep models. In *Proceedings of the First International Workshop on Multistrategy Learning*, pages 131–143, Harper's Ferry, West Virginia, 1991.

[Scherzer *et al.*, 1990] T. Scherzer, L. Scherzer, and D. Tjaden. Learning in bebe. In T. Anthony Marsland and Jonathan Schaeffer, editors, *Computers, Chess, and Cognition*, pages 197–216. Springer-Verlag, New York, 1990.

[Sebag and Schoenauer, 1990] Michèle Sebag and Marc Schoenauer. Incremental learning of rules and meta-rules. In *Proceedings of the 7th International Conference on Machine Learning*, pages 49–57, Austin, Texas, 1990.

[Sebag and Schoenauer, 1992] Michèle Sebag and Marc Schoenauer. Learning to control inconsistent knowledge. In *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 479–483, Vienna, Austria, 1992.

[Shannon, 1950] Claude E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41(7):256–275, 1950. Reprinted in [Levy, 1986].

[Shapiro, 1981] Ehud Y. Shapiro. An algorithm that infers theories from facts. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 446–451, 1981.

[Shapiro, 1982] Ehud Y. Shapiro. *Algorithmic Program debugging*. The MIT Press, Cambridge, 1982.

[Shavlik and Towell, 1989] Jude W. Shavlik and Geoffrey G. Towell. An approach to combining explanation-based and neural learning algorithms. *Connection Science*, 1(3), 1989.

[Srinivasan *et al.*, 1992] A. Srinivasan, S. H. Muggleton, and M. E. Bain. Distinguishing noise from exceptions in non-monotonic learning. In *Proceedings of the 2nd International Inductive Logic Programming Workshop*, Tokyo, 1992.

[Tadepalli, 1986] Prasad Tadepalli. Learning in intractable domains. In *Machine Learning: A Guide to Current Research*. Morgan Kaufmann, Los Altos, California, 1986.

[Tadepalli, 1989] Prasad Tadepalli. Lazy explanation-based learning: A solution to the intractable theory problem. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, 1989.

[Tadepalli, 1990] Prasad Tadepalli. On quantifying approximation. In *Working Notes of the AAAI Workshop on Automatic Generation of Approximations and Abstractions*, pages 257–266, Boston, Massachusettes, 1990.

[Tecuci and Kodratoff, 1990] Gheorghe Tecuci and Yves Kodratoff. Apprenticeship learning in imperfect domain theories. In Yves Kodratoff and Ryszard S. Michalski, editors, *Machine Learning: An Artificial Intelligence Approach, Vol. III*, pages 514–551. Morgan Kaufmann, San Mateo, California, 1990.

[Tecuci and Michalski, 1991] Gheorghe D. Tecuci and Ryszard S. Michalski. A method for multistrategy task-adaptive learning based on plausible justifications. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 549–553, Evanston, Illinois, 1991.

[Tenenberg, 1987] J. Tenenberg. Preserving consistency across abstraction mappings. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 1011–1014, Milano, Italy, 1987.

[Unruh and Rosenbloom, 1989] Amy Unruh and Paul S. Rosenbloom. Abstraction in problem solving and learning. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 681–687, 1989.

[Valiant, 1984] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[Varšek, 1992] Alen Varšek. Qualitative model evolution. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 1311–1316, 1992.

[Weld and deKleer, 1990] D. S. Weld and J. deKleer. *Readings in Qualitative Reasoning about Physical Systems*. Morgan Kaufmann, San Mateo, California, 1990.

[Widmer, 1989a] Gerhard Widmer. An incremental version of Bergadano & Giordana's integrated learning strategy. In *Proceedings of the Fourth European Working Session on Learning*, Montpellier, France, 1989.

[Widmer, 1989b] Gerhard Widmer. A tight integration of deductive and inductive learning. In *Proceedings of the 6th International Workshop on Machine Learning*, Ithaca, New York, 1989.

[Widmer, 1991] Gerhard Widmer. Using plausible explanations to bias empirical generalization in weak theory domains. In *Proceedings of the 5th European Working Session on Learning*, Porto, Portugal, 1991.

[Widmer, 1992a] Gerhard Widmer. Learning with a qualitative domain theory by means of plausible explanations. In Ryszard S. Michalski and Gheorghe D. Tecuci, editors, *Machine Learning: A Multistrategy Approach, Vol. IV*. Morgan Kaufmann, San Mateo, CA, 1992.

[Widmer, 1992b] Gerhard Widmer. Qualitative perception modeling and intelligent musical learning. *Computer Music Journal*, 16(2):51–68, 1992.

[Wilkins, 1982] David E. Wilkins. Using patterns and plans in chess. *Artificial Intelligence*, 18:1–51, 1982. Reprinted in [Levy, 1986].

[Wilkins, 1990] David C. Wilkins. Knowledge base refinement as improving an incorrect and incomplete domain theory. In Yves Kodratoff and Ryszard S. Michalski, editors, *Machine Learning: An Artificial Intelligence Approach, Vol. III*, pages 493–513. Morgan Kaufmann, San Mateo, California, 1990.

[Williams, 1988] B. Williams. MINIMA — A symbolic approach to qualitative algebraic reasoning. In *Proceedings of the 7th National Conference on Artificial Intelligence*, pages 264–269, Minneapolis, Minnesota, 1988.

[Zweben and Chase, 1988] Monte Zweben and Melissa P. Chase. Improving operationality with approximate heuristics. In *Proceedings of the AAAI Spring Symposium on Explanation-Based Learning*, pages 100–106, Palo Alto, California, 1988.