Creative Problem Solving and Automated Discovery

— An Analysis of Psychological and AI Research —

Klaus Schmid

Technical Memo

TM-95-04

**Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH**

**Technical
Memo**
TM-95-04

Creative Problem Solving and Automated Discovery

— An Analysis of Psychological and AI Research —

Klaus Schmid

September 1995

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, Sema Group, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry of Education, Science, Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct systems with technical knowledge and common sense which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Computer Linguistics
- Programming Systems
- Deduction and Multiagent Systems
- Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Dr. Dr. D. Ruland
Director

Creative Problem Solving and Automated Discovery

— An Analysis of Psychological and AI Research —

Klaus Schmid

DFKI-TM-95-04

This work has been supported by a grant from The Federal Ministry of Education, Science, Research and Technology (FKZ ITWM-9405).

ISSN 0946-0071

Creative Problem Solving and Automated Discovery

— An Analysis of Psychological and AI Research —

Klaus Schmid
German Research Center for Artificial Intelligence (DFKI)
P. O. Box 2080, D-67608 Kaiserslautern, Germany
Email: schmid@dfki.uni-kl.de

Abstract

Since creativity is the ability to produce something novel and unexpected, it has always fascinated people. Consequently, efforts have been made in AI to invent creative computer programs. At the same time much effort was spent in psychology to analyze the foundations of human creative behaviour. However, until now efforts in AI to produce creative programs have been largely independent from psychological research.

In this study, we try to combine both fields of research. First, we give a short summary of the main results of psychological research on creativity. Based on these results we propose a model of the creative process that emphasizes its information processing aspects. Then we describe AI approaches to the implementation of the various components of this model and contrast them with the results of psychological research. As a result we will not only reveal weaknesses of current AI systems hindering them in achieving creativity, but we will also make plausible suggestions — based on psychological research — for overcoming these weaknesses.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the 16th century, Copernicus introduced the heliocentric model of the solar system. In 1859, Charles Darwin published the book "The Origin of Species". In 1905, Albert Einstein published a theory which later became renowned as the special theory of relativity.

Each of these events changed the way we interpret our world. They can truly be regarded as major creative achievements. Creative feats like those have always fascinated people. However, creativity comes in many forms: as a major scientific achievement like those cited above, as innovations in art like *cubism*, as a new technical innovation like a car, or — more often — as everyday innovations like a witty remark.

Due to the fascination creativity has on people, it is not astonishing that several efforts exist to develop software that enables computers to achieve a certain level of creativity. Some of these efforts were made in the field of art (e.g., TALE-SPIN is a story-writing program, AARON generates drawings, etc.), other efforts were made in the field of science (e.g., AM is a program for making discoveries in the field of mathematics and the BACON-series of programs was designed for replicating innovations in the field of chemistry). Only recently a new sub-discipline of design emerged in AI: *creative design*. The aim of this discipline is to develop software that is able to design artifacts which can truly be regarded as creative or that is able to assist people in designing such artifacts.[1]

As this study was conducted at the IIS-department[2] of the DFKI GmbH, we will restrict ourselves to creativity in science and engineering. However, the principles underlying creativity in art are often regarded as being basically identical to those underlying creativity in technical disciplines (cf. [Bod91, Wei89]). Consequently, our study may have implications for creativity in other areas, too.

1.1 Goal and Solution Approach of This Study

The central goal of this study is:

> The identification of hindrances to the creativity of existing AI systems and of ways to provide AI systems with a higher degree of creativity.

In order to achieve this goal we decomposed it into four subgoals:

1. The meaning of the term *creativity* should be clarified.
2. Abilities and qualities that are useful for generating creative products should be identified.

[1] Sometimes this is also called *innovative design*. However, some authors distinguish between innovative design and creative design by the degree the design differs from common practice in a field. Here, we will follow this terminology. In section 5.3 we will elaborate on this distinction in more detail.

[2] Forschungsbereich Intelligente Ingenieursysteme

3. Existing AI systems should be analyzed with respect to the criteria identified in the preceding step.

4. Based on the contrast between existing AI implementations and the criteria identified as being important for creative systems, proposals for improving AI approaches should be made.

Our approach for achieving the first two subgoals is based on a review of psychological literature. Two reasons were decisive for choosing this approach:

- Humans are the only generally accepted source for creative products. Consequently, they are the only existing model of creative behaviour. The study of this model is the topic of psychological research.
- Creativity has been studied most intensively in psychology.

As we will recognize in the course of this study, using psychological models as a reference has additional advantages:

- In the near future autonomous creative systems are rather unlikely. Instead, systems for supporting human creativity are more realistic. Based on this assumption it may seem inadequate to restrict this study to abilities necessary for autonomous creative behaviour. However, we believe that the same abilities are necessary for support systems, although at a lesser degree of perfection. We will not discuss the additional abilities necessary for supporting creativity in order to keep this study at a reasonable size. Relevant material can be found elsewhere (e.g., [Fis92]).

  For such support systems our decision to base this study on psychological models is advantageous, because these systems should behave in a way intelligible to the user. This is more likely for systems modelled after human behaviour.

- As we will see, creative solutions often require that the problem itself gets redefined. If such a change of the problem specification by a computer shall be accepted by humans, the reasons for doing so need to be intelligible to humans. Here again, it is helpful if the behaviour of the computer is based on human behaviour.

Now, we will briefly discuss the approaches taken for achieving the various subgoals:

Define the term creativity: Defining the term *creativity* first, before discussing it, is a generally accepted approach in psychology, but usually not used in AI. Instead many authors appeal to an intuitive understanding of creativity (e.g., [RP93, SL89, Yaz89]). As no general agreement exists about what creativity is, a commitment to a certain definition is always disputable. However, we believe that a debatable definition is better than none at all, as it allows to clarify differences in understanding.

We will discuss different approaches for defining creativity used in psychology in chapter 2. Three basically different frames of reference for the definition of creativity can be distinguished: the creative product, the creative process, and the creative person.

Abilities and qualities: Some basic abilities and qualities that are necessary (or at least useful) for creativity are discussed in chapter 2. A more detailed discussion of the abilities is given in chapter 3. Using the commonly used Wallas model of the creative process as a starting point, we give a new model of the creative process, which refines the stage model given by Wallas and relates the stages to the various operators that are especially important in them. Due to the operator-level description that this model provides, it is more apt for a discussion of the creative process from a information processing point of view than the original model.

In chapter 4, we will shortly discuss the results of our review of psychological literature.

Analysis of existing AI systems: In order to provide the reader with an overview, we will review two areas of AI, that are especially important for the discussion of creativity: *automated discovery* and *creative design*. We will do this in section 5.2 and 5.3 respectively.

For each of these two types of systems we will give a general model which subsumes the existing models we know. Additionally, we will shortly discuss the range of variation that exists for the components of the models. Naturally, both models are instances of our general creativity model.

In chapter 6, we will discuss the AI approaches to the implementation of the various abilities we identified as being important for the creative process in chapter 3.

Proposals for improvements: In chapter 6, we will also contrast the AI implementations of the various abilities with the results we achieved in chapter 3 about the human form of these abilities.

This will lead us to several possibilities for the improvement of AI implementations of these abilities and to opportunities for future AI research. Further suggestions for the improvement of AI systems in a more fundamental way (i.e., not restricted to single abilities) can be found in chapter 7.

Naturally, this study must not be misunderstood as a guideline for the implementation of creative systems, as this is well beyond the scope of this work. Instead it should be regarded as a compendium of ideas, gathered in the hope that they will further the creative process of building creative programs.

1.2 Structure of the Study

In the following chapter we will begin with an overview of psychological approaches to the study of creativity and an outline of the results thus achieved. In chapter 3 we will examine models of the creative process and will introduce a process model which we will use as a standard model throughout this study.[3] In the next chapter we will summarize and discuss the results of the psychological part of this study.

Chapter 5 is devoted to an introduction of two disciplines in AI that have especially tight relations to the topic of creativity: *automated discovery* and *creative design*. In chapter 6 we will examine in detail the various abilities (operators) we identified as being important in chapter 3. For each ability we will discuss the contributions AI made to its implementation and we will contrast the result of these efforts with psychological results.

In chapter 7 we will present the conclusion of our study and point out open problems that deserve additional work in AI. Finally, appendix A relates the results of this study with the VEGA project that provided the environment for conducting this study.

1.3 A Reader's Guide

As mentioned above, quite different subgoals are addressed by different parts of this study. Consequently, the various parts may be important for readers with different interests. Besides the goals we identified as our four subgoals in 1.1, this study also contains material for readers with other interests.[4]

A reader who wants to understand what abilities are necessary for being creative, what abilities are successfully simulated by AI-systems, and where possibilities for improvements exist, may find

[3] Throughout this study we will concentrate on the creative process and not on the creative product, as neither computer science nor psychology are usually competent for such a study (cf. [Ulm68, p. 29]).

[4] One might regard this as *emergent value* of this study.

the study interesting as a whole. However, most readers will probably have more restricted intentions:

- A reader trying to get a basic understanding of the meaning of the term *creativity* will find chapter 2 most interesting.
- If the reader is interested in the results of psychological research with respect to creativity (especially the creative process), chapters 2 and 3 will be most useful.
- A reader who only wants to get a general overview of creativity and some ideas for improving AI systems in order to help them be more creative should read chapters 2 and 7.
- In case the reader wants to get a compact description of an ability we identified as being important for creativity, he should read the *contributions from AI* part of the corresponding section in chapter 6.
- If the reader is also interested in the relation of this ability with its human form, the study of the corresponding section in chapter 3 and the *comparison with AI* part will be most interesting.
- A reader who is mainly interested in the AI aspects of creativity should read chapter 2 for a basic understanding of creativity, the beginning of chapter 3 for a introduction of our model, and should then read chapter 6. If the reader is not acquainted with the "information filtering model", he should also read appendix B.
- A reader who wants to get an overview of *automated discovery* or *creative design* systems should read section 5.2 or 5.3 respectively. These sections are also of interest to readers who are not acquainted with these types of systems and want to read chapter 6.

Further, a reader interested in the relation of this study to the VEGA approach should also read appendix A.

Chapter 2

Creativity in Psychological Literature

2.1 Aspects of Creativity

Three different approaches to the analysis of creativity can be distinguished:

1. analysis of the creative product
2. analysis of the creative person
3. analysis of the creative process

Psychological research mostly uses either approach two or three, because the first one is more apt for the philosophy of art or the philosophy of science (cf. [Ulm68, p. 16]).

In the course of this thesis we will concentrate on the study of the creative process. This can be expected to be the most important area for the purposes of this study, because AI deals with the description and invention of methods for knowledge processing. The three approaches are not equivalent, however: neither every product of thinking nor every reasoning process of a creative person needs to be creative. Moreover, the result of a creative process is not necessarily a creative product.

The last point is by no means obvious, therefore we want to illustrate it by an example. Consider the following question:

> 120 competitors participate in a tournament held according to the single elimination system. How many matches have to be made until the winner is established?

We gave this question to different people and could observe different strategies of answering it:[1]

1. Some people, who frequently organize or participate in competitions, did know the number of matches needed for 128 participants by heart or could easily deduce it. They simply subtracted the 8 matches which are unnecessary due to byes.
2. Another simple approach (which could not be observed) would be to model the tournament as a binary tree with the leaves (l) as players and the inner nodes (n) as matches. With this model one could apply the formula $n = l - 1$ from graph theory.

[1] This can by no means be seen as a controlled psychological experiment. We cite it here, nevertheless, because it illustrates the difference between creative process and creative product very nicely.

3. A slightly different strategy (which could be observed) was to use the same model and count the number of inner nodes: 1 node on the top level, ..., 32 nodes on the sixth level and 56 ($= 64 - 8$) on the last level.

4. One person gave a more interesting solution: He transformed the problem into a chain of matches (that is the winner of a match always plays against a competitor, who did not play yet) and gave the number of matches in this kind of tournament. However, he could not answer why this gives the correct solution.

5. No one could give at first try what is probably the most elegant (and creative) solution: In a tournament held according to the single elimination system there is exactly one player who wins every match. All the other players (119) lose in exactly one match. In each match only one player loses. Therefore, there are 119 matches.

The solution to the problem is not novel because one can conclude directly from the question that the solution is a number probably somewhat bigger than 100. Therefore, it is no creative product (see section 2.2 for complete criteria). However, solution 4 and 5 differ from the other solutions. In solution 4 the way in which such a tournament is typically held is transformed and in solution 5 the losers of the matches, who are not mentioned in the problem statement, are used for computing the total number of matches. Such transformations of the problem or of the informations used to solve it are called *restructurings* in the context of creativity. The amount of restructuring involved is usually regarded as a measure of the creativity involved. Consequently, solution 4 and 5 must be considered more creative than the other three. Hence, *creative processes need not result in creative products*. (If one does not regard the solution path itself as a product.) If no restructuring is necessary, e.g., the person already knows about solution 5 and simply reuses it, then this act is no longer regarded as being creative. The criterion of restructuring shows that we can only speak of a creative process (in the sense of psychology) if the reasoner has standard ways of reasoning *and* the possibility to break away from these. Therefore, a program simply doing breadth-first-search can hardly be regarded as being able of creative processes.

In principle, creative products can result from non-creative processes, e.g., by combining the basic constructs of a language in every possible way. This leads to a complete exploration of the solution space, thereby producing all possible products and therefore all creative products, too. However, this leaves open the important problem of selecting the useful products. This is often illustrated like this: Imagine a herd of apes sitting in front of typewriters pressing the keys totally at random. In principle, it is possible that one of the apes produces the play *Hamlet*, however he would not be able to recognize its value.

Generally, in research on creativity the belief is held that the criteria for a good product are usually not completely specified at the outset, but evolve during the generation of the product. Therefore, the process of creating a creative product can hardly be partitioned into a phase of blind variation and a phase of recognition. Accordingly, it is generally accepted that people produce creative products in creative processes. This believe is expressed by Boden (cf. [Bod91, p. 227]): "brute force search must be monitored by intelligence if anything creative is to result."

Above we saw that criteria for selecting a product are of crucial importance for creativity (Hamlet). This hints to a view of creativity as a goal-oriented process. Indeed, in psychological literature creativity is usually treated as a special case of problem solving: creative problem solving (cf. [Wei89, p. 16]). Russel also calls it "problem solving plus" (cf. [SK74, p. 14]). The problem to be solved may be given or self-proposed. In this view the creative product corresponds to the problem solution, the creative person to the problem solver, and the creative process is a problem solving process. This problem solving approach is especially apt for this study, because our main interest is in creativity in technical and scientific domains. However, many authors treat creativity in art as problem solving, too.

2.2 The Creative Product

A major part of the interest in creativity is due to the interest in creative products. Nevertheless, just as there is no generally accepted definition for creativity, there is none for the creative product either. There are, however, numerous attempts. Their most important features are summarized in the definition given in [Wei89]: A creative product has to be novel and solve the problem.[2]

Though this summary is very concise, the different definitions used in literature vary considerably. Instead of 'novel' Guilford uses the term 'unusual' in the sense of statistical rareness within a test population (cf. [Ulm68, p. 30]). Wallach and Kogan even demand that the idea (the product) needs to be unique within the test population (cf. [Ulm68, p. 30]). On the other hand Bruner demands that the product must be 'unexpected' (cf. [Ulm68, p. 31]). The definition proposed by Newell (cf. [ROD80, p. 116]) has the most resemblance to Weisberg's definition: He demands that the product shall be 'novel' and 'unconventional'.

Various authors use different frames of reference for judging the novelty of a solution. The possible frames range from 'it is unique within history' to 'the problem solver did not know it before' (cf. [Ulm68, p. 31]). Using the problem solver as the frame of reference is probably the most practical of these proposals (cf. [Bod93, p. 7]). In particular, choosing history or society as the frame of reference leads to the problem that the creativity of a product depends not only on the problem solver and his achievement but also on external conditions, which are not available for immediate examination.

The meaning of the requirement that the product must solve the problem is also not obvious. For example, Newell (cf. [ROD80, p. 116]) demands that the problem gets more detailed defined during the problem solving process. Consequently, he does not require the product to solve the problem, but only to be 'valuable'. Other authors demand that the product shall be 'useful' (cf. [Ulm68, p. 31]). The requirement, that the problem specification becomes more detailed during the course of problem solving, is especially important in the field of *creative design* (cf. [McL92], [Kol94], [Nav91]). Due to pragmatic reasons this aspect is often neglected in psychological experiments.

An interesting example for a creative problem solution is given by Poincaré (cf. [Wei89, p. 33]): Poincaré's original problem was to prove that there are no Fuchs' functions. However, his result was that there are Fuchs' functions and they are identical to the non-euclidean transformations.[3] Thus Poincaré was not able to solve his original problem. Instead he solved a slightly different problem with the property that from its solution one can conclude that the initial problem was unsolvable.

Another interesting example for the *restructuring of a problem specification* is the Sydney Opera House (cf. [McL92, p. 66]). This building was designed in response to a public competition. The entry form to the competition included six disqualification clauses. One entry was liable for disqualification under four. Nevertheless, this proposal was finally realized, because it did comply in an excellent way with several criteria which were not mentioned in the entry form.

The possibility that problem criteria may change during the problem solving process makes it very difficult to judge whether a product truly solves the corresponding problem.

The purpose of this study is to examine some methods which could make it possible for computers to behave creatively. Consequently it is necessary to have some criteria for deciding about the creativity of a product, even if this product is produced by a computer. Such a definition can be given similar to the one suggested by Weisberg:

- The product shall be novel in the sense that it is either a non-standard instantiation of a pattern known to the problem solver or it involves a hitherto unknown pattern.[4]

[2] This implies that there needs to be (at least implicitly) a problem that calls for a solution. However, if this is not the case then the product can not be regarded as being useful.

[3] Additionally, this example is important because the process model proposed by Wallas (cf. section 2.4.1) is based on Poincaré's detailed description of this experience.

[4] This criterion is very much alike to the requirement that creativity involves the exploration or transformation of a conceptual space given in [Bod94, p. 119].

- The product solves the problem, i.e., it complies with every problem criteria which is regarded as being essential with respect to the domain knowledge of the problem solver (including the knowledge that has been elaborated during the problem solving process).

2.3 The Creative Person

2.3.1 Measuring Creativity

In order to be able to compare creative and non-creative people it is necessary to classify people with respect to this criterion. There exist two different approaches to this problem:

1. The creativity is measured with tests.
2. The creativity is judged by another person acquainted with the subject or at least with his work.

Both approaches have their disadvantages. The results of creativity tests have only a small correlation with the behaviour shown outside the test situation (cf. [ROD80, p. 130]). Therefore Ausubel (cf. [Ulm68, p. 59]) concludes that only judgement can give a valid measurement of creativity. However, judgements often highly correlate with the criteria of productivity and effectiveness (cf. [Ulm68, p. 69]).

2.3.2 Qualities of Creative People

Independently of the method of measuring used, it can be shown that intelligence is a very important precondition to creativity (cf. [SK74, p. 50]). Accordingly, the *boundary value hypothesis* implies that below an IQ of 120 intelligence is decisive for creativity, but above that value both qualities are independent (cf. [ROD80, p. 126]). However, some experiments failed to confirm the independence of these two qualities even above an IQ of 120 (cf. [ROD80, p. 126]).

Despite these problems of identification, a certain consensus exists with respect to the qualities of creative people. Weisberg ([Wei89]) mentions that in general a *high level of self-confidence* and *certainty of ones own creativity* is ascribed to a genius. Also, creative people distinguish themselves by a *good ability to concentrate*, by *high commitment* and a *bigger readiness for risk* (cf. [Wei89]). Landau points out in [Lan69] that for the incubation phase (cf. section 2.4.1) a *high level of frustration tolerance*[5] is necessary.

The cognitive styles of creative people seem to be different from those of less creative people. These differences manifest (at least) along the following dimensions (cf. [SK74], [Ulm68]):

1. More *open-mindedness with respect to experiences*, especially such of ambiguous nature.
2. *Preference of complex stimuli*.
3. *Tolerance with respect to diverging details*.
4. *Field-independence*, that is they are less fooled by surrounding stimuli (e.g., figures within a complex picture are more easily identified).
5. *Reflexiveness*, that is they tend to reflect upon their opinion critically.

[5] In psychological literature frustration has a different meaning than in common language. Here it signifies any situation in which a person can not reach his goal. This definition is independent of the feelings the person associates with the failure.

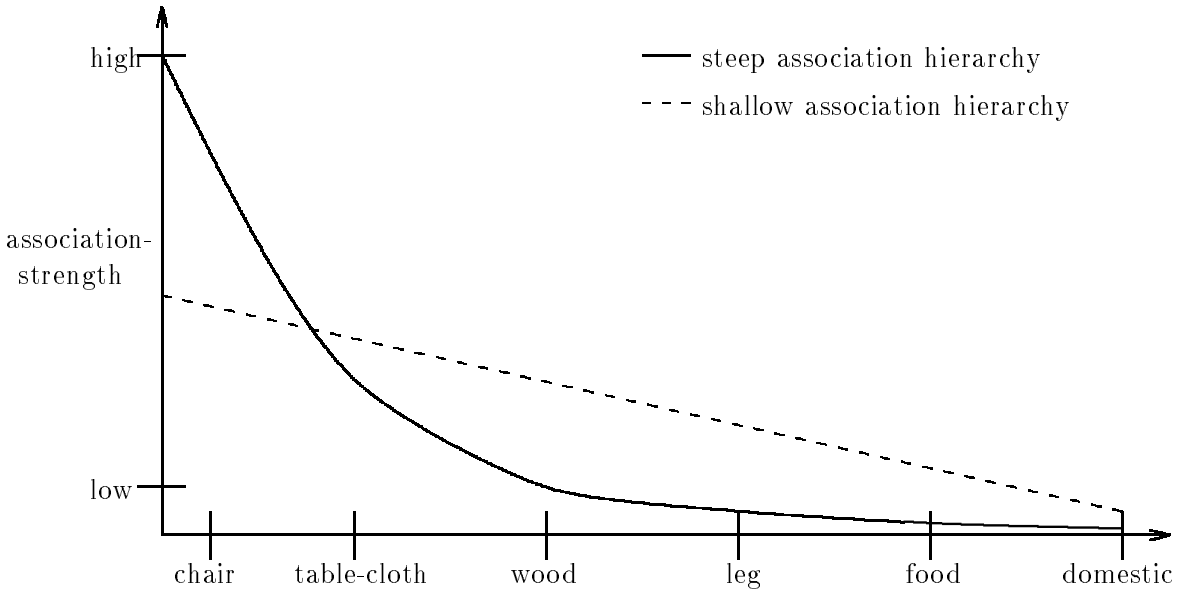

Figure 2.1: Association strength for the word 'table' according to Mednick

Some abilities seem to be especially prominent among creative people. For example, Ulman mentions the *ability to form categories* (cf. [Ulm68]). Mednick (cf. [SK74, p. 29]) proposes an *ability of broad association*: "Creative people [produce] only few but extraordinary associations during a long time interval"[6]. He proposes the model of the association hierarchy for explaining this ability. This model correlates the remoteness of a concept (w.r.t. a reference concept) with the probability that it is associated. According to this model non-creative people have a steeper association hierarchy than creative people. Figure 2.1 illustrates this model. Guilford also emphasizes the importance of *divergent thinking* in creative reasoning, i.e., the consideration of a large number of alternatives. Seiffge-Krenke ascribes *flexible systems of control* to creative people in [SK74]. Newell, Shaw and Simon (cf. [SK74, p. 73]) mention "a process of learning which generates new heuristical methods"[7].

We believe that these qualities are at least partially interconnected: forming categories in the presence of complex stimuli is alleviated by a certain tolerance with respect to diverging details and a certain readiness for risks (in this case of wrong classification).

Although there are some common characteristics of creative people these are not sufficient for creativity alone. Especially, for undertaking creative acts which are novel with respect to history more is required. The abilities and attitudes of a person must fit the state the area of research is in. This point can be very clearly illustrated by the example of Einstein: In the beginning of the twentieth century he could contribute considerably to the advance of physics. However, when Quantum Physics became the dominant theory of physics his attitude to it made it virtually impossible for him to contribute to physics in the way he had done before (cf. [Wei89, p. 113–115]).

2.3.3 Guilford's Model of Creativity

Guilford identifies several factors in his intelligence model which he holds to be important with respect to creativity (cf. [ROD80, p. 120]):

Fluency: Ability of recalling information from memory, e.g., recalling associations, word sequences, etc.

[6] Kreative [produzieren] relativ wenige, aber ungewöhnliche Assoziationen über einen längeren Zeitraum hinweg.
[7] einen Lernprozeß, der neue heuristische Methoden hervorbringt

Sample task: mention as many words as possible ending in '-tion' within four minutes.

Flexibility: Degree of being bound to certain categories in thinking.

spontaneous: Without an exact goal.
Sample task: give as many uses as possible for a brick (only different categories are counted): for building, as paper-weight, etc.

flexible: With an exact goal.
Sample task: transform one figure made of matches into another one with only as few moves as necessary.

Originality: Ability to produce original (that is statistically rare) answers.
Sample task: which city like actors the most? — publicity!

Elaboration: Ability to make plans.
Sample task: a draft of a plan needs to be refined.

Sensitivity: Sensitivity for problems.
Sample task: give ways of improvement for some household utensils.

Redefinition: Ability to see an object or part of an object from a new point of view.
Sample task: which of the following objects is suited for lighting a fire: a string, a cabbage, a pocket watch, a fish or a needle? — pocket watch!

These factors are especially important, because many tests of creativity use this (or a similar) characterization (cf. [HC91]). Sometimes these abilities are termed 'divergent thinking', because the solutions they give are not uniquely determined. These abilities are also emphasized in most popular books on creativity.

2.3.4 Knowledge and Creativity

Creative people possess the ability to generate new heuristic methods at need, but heuristic mechanisms are no replacement for missing knowledge (cf. [SK74, p. 41]). Especially, Weisberg ([Wei89]) emphasizes the necessity of domain knowledge for creative achievements.[8] Thus, he points out that for the mastery of chess-playing or composing more than ten years of experience are necessary (this even holds for Mozart).

Having the right kind of knowledge available is as important to the creative process as it is to any problem solving process. This is pointed out by Kulkarni and Simon in an analysis of the discovery of the urea cycle by Hans Krebs (cf. [KS88]). They mention that his familiarity with the tissue slice method can be regarded as "his source of comparative advantage" (p. 173). On the other hand "his lack of expert knowledge of organic reactions ... conferred on him some real benefits", too. This shows that *knowledge can have adverse effects*, too. This is due to the fact that people tend to use knowledge, which they regard as being helpful w.r.t. their goal and which they have available in an usable form. Note the similarity of this principle with the *principle of rationality* proposed by Newell (cf. [Die86, p. 288]):

> If an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action [as one of the possible actions to perform next].

However, there are several differences to the model of the *rational agent*:

- The classification of knowledge as being helpful may be wrong.

[8] According to Simon (cf. [Fis92]) an expert — and creative achievements are usually made by experts — possesses at least 50 000 chunks of knowledge.

- The subject may possess helpful knowledge, but it is not in a usable form, i.e., he may not regard it as being related to the goal, he may be unable to apply it to the problem, etc . . .
- The axiom *knowledge closure* (cf. [Die86, p. 290]) certainly does not hold for human problem solvers:

  > *Knowledge closure.* If an agent knows a body of facts, F, then the agent also knows any facts that are deductive consequences of F. In other words, an agent knows the deductive closure of his knowledge.

However, *the above restrictions certainly hold for any problem solver working with limited resources (time, space).* That is, he or it (man or machine) can be mislead by knowledge, which is not really helpful for achieving the goal.[9]

What has been pointed out about the knowledge of facts is also true w.r.t. to strategic knowledge. By permanent training certain habits of problem solving become automatic. This ability to automate is a necessary part of intelligence according to Sternberg's definition (cf. [KP88]). This automation of problem solving behaviour is also termed a *problem set* or, especially if it has adverse effects, *fixation on past experiences*. It can become a hindrance to problem solving, if the generated mechanisms respond to situations in which they are not helpful. This is illustrated in the following example: Given a switch and a relay and in need of a weight people typically used the object they had not used before in its usual function. This effect is the more prominent the less time elapsed between the original use and the search for a weight. Finding the correct reaction despite these effects is also counted as creativity by some authors (e.g., Weisberg).

In this section we illustrated that no single ability of creativity exists. Instead the necessary abilities and knowledge depend on the situation and can probably not be inferred from the problem description alone.

2.4 The Creative Process According to the Wallas-Model

This section gives an overview of the Wallas-model of the creative process, because this model is used most often in literature. However, the restriction to the Wallas-model is no real limitation, as the different models of the creative process differ not very much (cf. [SK74, p. 15]). In the next chapter a variant of this model will serve as a basis for a more detailed analysis of the creative process.

2.4.1 Outline of the Wallas-Model

The Wallas-model distinguishes four stages:

Preparation: During this stage knowledge and ideas relevant to the problem solution are gathered and the first problem solving attempts are made. Finding the problem (if not specified at the outset) belongs to this stage, too.

Incubation: The first problem solving attempts failed (the problem solver is frustrated). This is a plausible reason why creative people need a high level of frustration tolerance (cf. page 8). Without it they would simply give up in this stage.

Some psychologists describe this stage as if no conscious work on the problem is done (cf. [Lan69, p. 66]). However, most authors advocate the view that moments of rest and work alternate during this stage. Olton calls this *creative worrying* (cf. [Wei89, p. 44]).

Illumination: Suddenly and unexpectedly the solution is found. This solution is sometimes called illumination, too.

[9] Using the principle mentioned above, that a problem solver tends to use knowledge which it regards as being helpful w.r.t. its goal and which he has available in an usable form, *we can identify the possession of knowledge and its usage* under the aforementioned restrictions.

Verification: The problem solver verifies whether the solution is correct and whether it solves the problem. This may require further elaboration (cf. 'ability to elaborate' as proposed by Guilford, page 10).

There have been different experiments to observe these stages in creative problem solving (cf. [SK74, p. 19]). From these experiments it can be concluded that *they can be observed but do not necessarily occur in this order*.

In particular, the hypothesis which is regarded as illumination must not necessarily be correct and can be rejected during the verification stage. In such a case a sequence of illuminations can occur (cf. [SK74, p. 34]). Another interesting point about these stages is that they occur independently from the creativity of the subjects.

2.4.2 Illumination in the Creative Process

The following questions are of special interest to psychology:

- What constitutes an illumination?
- What happens during the incubation stage that leads to an illumination?

The illumination is often described as a restructuring of the knowledge (or the problem): A new interpretation is superimposed on the information at hand. This step is regarded as being crucial for the creativity of the process (cf. [Ulm68, p. 28]). The new features discovered in this step are called *emergent value*. McLaughlin (cf. [McL92, p. 44]) defines emergent value as "the value that presents itself only during the course of the development of a creative product". Some authors stress the role which the interpretation of communication plays in the recognition of new aspects of ideas (e.g., [ECJS94, p. 46], [KW93, p. 54]).

Several explanations exist for the processes which occur during the incubation stage and prepare the illumination:

- The Gestalt theory proposes the view of insightful problem solving. In this view problem statement and knowledge complement each other, resulting in a solution to the problem.
- Behaviourism (cf. [Wei89, p. 59]) suggests that previous problem solving experiences are associated with the current situation due to similarities with the original problem situations. One of them is then transferred onto the current situation along with the corresponding solution. If such a transfer is not possible, trial-and-error is the only possibility left for solving the problem.

  In this context the ability to associate concepts which are very far apart is especially important. This ability is postulated for creative persons by Mednick. Consequently, the transferred experience need not to be the most similar situation. Indeed, this is rather unlikely for creative solutions.
- A combination of both proposals is suggested by Harlow based on experiments he conducted with apes. He could show that they are able to learn a strategy by trial and error which then enables them to insightful problem solving (cf. [Wei89, p. 49]).
- Koestler's bisociation theory (cf. [Wei89, p. 40]) proposes that *new* combinations of concepts are spontaneously formed.
- Learning theoretical approaches emphasize the role of forgetting (e.g., wrong assumptions), which happens during the breaks. This can help in overcoming adverse habits or states of fixation (cf. page 11).

Chapter 3

An Analysis of the Creative Process

While in the last section we introduced the Wallas-model, this chapter will give a more detailed analysis of the creative process based on results from psychological literature on creativity. In this analysis we will use results from *cognitive science*, too.[1] Usually studies in *cognitive science* do not claim to be concerned with creativity. However, there are several reasons why they are of interest with respect to the subject of this study:

- *Cognitive science* deals with the examination of human problem solving behaviour.
- Many studies deal with *scientific discovery*. In this context finding the correct hypothesis for explaining an observation is of central importance. This can be often regarded as an act of creativity.
- In this area examinations of the process are usually very detailed and instructive.
- The models which are developed for explaining the results are often given in a way to allow for computational simulation.

In the following section we will describe a model of the creative process which was developed based on the Wallas-model. Then we will relate the various components of this model with results taken from psychological literature. In chapter 6 we will contrast these results with ideas from AI.

3.1 A Model of the Creative Process

In this section we will present an important result of our study: *a new model of the creative process*. Our model is loosely based on the Wallas-model, but also compatible with other models proposed in psychological literature. Moreover, we took into account work in *creative design* (e.g., [KW93], [Nav91], [RG92], ...) and *scientific discovery* (e.g., [KD88], [KS88], ...). While other models only define a number of stages, our model also sketches an operator/process level and tries to describe control and information flow. Naturally, this model is only hypothetical. Although a validation of this model is beyond the scope of this study, it is plausible as it is consistent with experimental evidence presented in literature.

However, the central focus of this model is not to give a new concept of the creative process or to be directly implementable, but to help in organizing the material taken from psychological literature

[1] But only as far as the analysis of human behaviour is concerned. Computer simulations will be described in chapter 6.

in a systematical way, such that the different actions performed are emphasized. Like the Wallas-model, this model consists of four stages. These stages are: *problem recognition*, *preparation*, *incubation*, and *verification*. However, these stages are introduced only for relating this model to the models commonly used in psychological literature. For the purposes of this study the description levels 'process' and 'operator' are more interesting. In the rest of this section we will extract five subprocesses of the creative process (cf. table 3.1). The relation between processes and stages is not one-to-one (cf. figure 3.1). The various operators that are grouped together to form a process can be regarded as alternatives the problem solver possesses for performing an activity. The control component must decide which of the abilities (operators) should be used in a specific situation. Further, each of these operators should be regarded as representing a whole group of knowledge processing activities, as a human possesses several alternatives for forming concepts.

3.1.1 Stages of the Creative Process

Problem recognition. Most psychological examinations of the creative process use the Wallas-model as the fundamental approach. However, other models have been developed, too (cf. [SK74, p. 15]). A comparison of these models reveals that many authors regard 'problem recognition' as a separate stage and not as a part of the preparation (e.g., Torrance, Harris). This is ingenious, because the abilities needed in this stage are different from those needed in the rest of the preparation stage (cf. 'sensitivity for problems' (Guilford), page 10). Additionally, many abilities are unique to this stage. Therefore, the model given here contains problem recognition both as a stage and as a process.

Preparation. The problem recognition stage is followed by the preparation stage, which serves the gathering of information with relation to the problem. This can either be done by extracting information from the environment, or by activating already known knowledge. This activation of knowledge is important, because it makes those information items available, which will be considered for solving the problem. From a technical point of view, the fact that most operators are only applied to the active knowledge and not to all the knowledge leads to a considerable narrowing of the search space. In the context of the preparation stage, mental simulation (as a substitute for real experiments) and association are the most important activation operators.

Questions asked or experiments carried out during the preparation stage are done without any hypothesis about a possible problem solution. They serve only for information gathering and are not used for verification purposes. Experiments of this kind have been reported in a study conducted by Klahr and Dunbar (cf. [KD88, p. 26]). According to the SDDS-model, which was proposed by the authors for explaining the observed behaviour, these experiments are only made for inducing new hypotheses which are capable of explaining the results of the experiments. Such a behaviour corresponds to the 'trial-and-error'-behaviour, which was postulated by *Behaviourism* (cf. page 12). The first hypotheses are generated in this stage, too.

Incubation. During the incubation stage the activation of knowledge continues. Additionally, the restructuring of knowledge begins.

According to the Wallas-model, the illumination stage would be the next stage. However, in most models of creativity this is not regarded as a separate stage, but as part of the previous stage. This seems sensible, if one takes into account what an illumination is:

- Suddenly and unexpectedly the solution is found (cf. page 11).
- A connection between the current idea and the goal is discovered (cf. [SK74, p. 18]).

Therefore, the term 'illumination' can be either used to denote the moment of finding the solution or to denote the finding itself. While the Wallas-model uses the former interpretation, we will use the later in the model proposed here, and will regard the moment of finding the solution as part of the incubation stage. According to this interpretation, *the illumination is the result of a successful*[2]

[2] This does not mean that the idea is correct, it is only necessary that it *seems* to solve the problem for being an illumination.

Name of the process	Task the process solves
Problem recognition	Identify a problem
Search for knowledge	Search in the experiment space; acquire external information for inducing new ideas or for completion of existing ideas
Activation of knowledge	Activation of knowledge which can aid in the problem solving process
Restructuring of knowledge	Search in the hypothesis space; generation of new representations of knowledge, which are more apt for solving the problem.
Evaluation	Evaluation of newly activated knowledge with respect to the goal; generation and modification of tasks

Table 3.1: The processes in the creativity model

incubation. Depending on the kind of problem, the illumination can be a sketch, an abstract plan, a hypothesis for explaining an observation, or anything similar.

Verification. As can be seen from the enumeration above the illumination is not the final result of the creative process. Instead, some details still need to be added. For that purpose, some additional restructuring steps may be necessary (especially planning and adaptation). These are carried out during the verification stage. In this stage again some experiments can be made for verifying that the completed product does indeed solve the problem. However, they are different from the experiments in the preparation stage insofar as they are used for testing the hypotheses and not for inducing them.

Figure 3.1 gives a survey of this model of creativity and Table 3.1 gives an overview over the processes in this model. Additionally, table 3.2 gives a survey of the I/O-behaviour of the various operators. Those processes which are operators by themselves are also included.

3.1.2 Evaluation and Control in the Creative Process

The evaluation process in this model is applied for every addition to the active knowledge, because each addition must be examined with respect to the consequences it has on the problem solving process (promising hypothesis, change in the problem specification, etc.). Therefore, it can not be ascribed to any stage in particular. This evaluation process usually is not mentioned in psychological literature or it is only mentioned as a part of the verification stage. However, Seiffge-Krenke [SK74, p. 31] mentions that associations are selected critically and realistically. The change of design criteria (cf. [Kol94], [McL92]) points to the necessity of evaluation, too.

There is one aspect which is only superficially described in figure 3.1: This is the aspect of control. This is especially true at the process and operator level. Control is strongly influenced by the results of the evaluation process, because this process judges the quality of the various results, chooses the best hypothesis, recognizes opportunities for changes in the problem description, etc.

Control has not been a central concern in this model because psychological literature does not describe it very detailed. Especially, little is said with respect to the sequence of the various stages. In principle, it seems possible to go back from any stage to any earlier one. It is even possible that the verification is unsuccessful. This may result in returning to the preparation stage (cf. [SK74, p. 17]). However, it seems impossible to skip a stage, because the information generated in one stage is usually needed in the following.

Usually the creative process is described as if the different stages would be executed only once. Especially, there is only one illumination (the creative leap), which suddenly brings the problem solver much closer to his goal. More recently, some authors did abandon this model in favor of a *model of small steps*, which suggests, that several illumination steps bring the problem solver successively closer to his goal (e.g., Boden, Weisberg, ...). The changes of hypothesis during

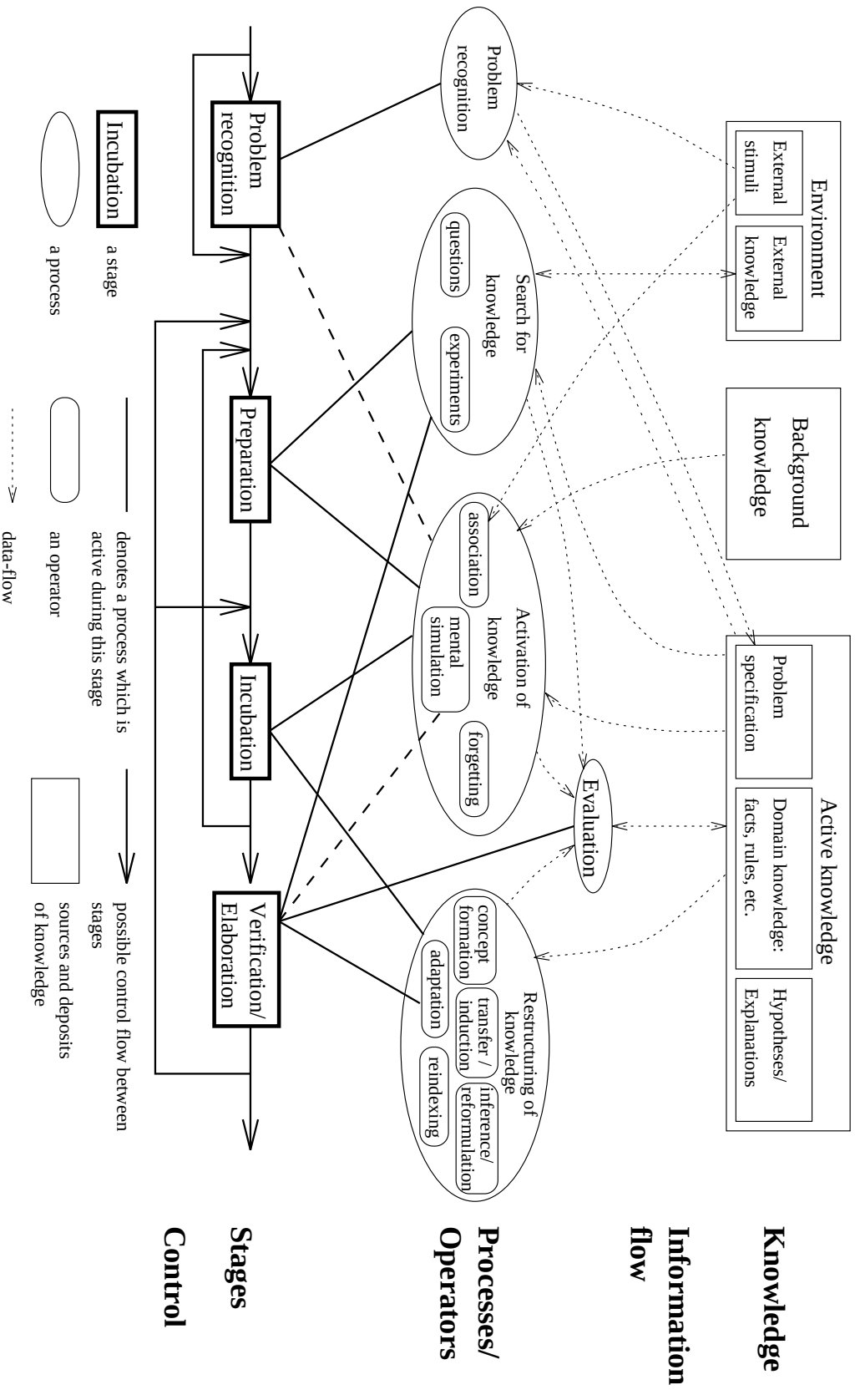

Figure 3.1: Stage model of creativity

Name	Input	Output	Side-effect
problem recognition	external stimuli or deficiencies in the knowledge base	problem specification	(new problem solving context is created)
questions and experiments	knowledge about the environment, internal question, (hypothesis about the answer)	new knowledge (external knowledge)	manipulation of the environment
association	external stimuli or active knowledge, inactive knowledge	active knowledge	—
mental simulation	problem knowledge (activity to simulate), background knowledge and problem knowledge	active knowledge, inferred knowledge	—
forgetting	active knowledge, (knowledge about usage)	—	knowledge becomes deactivated
concept formation	active knowledge*	inferred knowledge	—
transfer/induction	active knowledge*	inferred knowledge	—
adaptation/inference	active knowledge*	inferred knowledge	—
reindexing	inferred knowledge†, background knowledge	active knowledge (degree of activation)	creation of new retrieval paths in the knowledge base
evaluation	entries to the active knowledge, active knowledge		• other operators may be called: (e.g., mental simulation) • new tasks may be generated ⋮

*The starting point is the active knowledge, but as a result other knowledge may become activated.

†There exists some evidence that the reindexing is influenced by new concepts created, new relations inferred, etc.

Table 3.2: The operators in the creativity model

search of the hypothesis space as proposed by Klahr and Dunbar (cf. p, 25) can be regarded as such steps. However, this view is not contrary to the model of figure 3.1. Instead, this behaviour can be described by iterating through the model. Then, an illumination is only one step towards the goal. The important fact is that the basic processes remain untouched by this transformation.

In addition to iteration, it seems also reasonable to expect recursions in the creative process. For example, the design of an experiment can be a creative process on its own. Or the problem can be divided into subproblems (cf. strategy (HG3) 'divide-and-conquer' in [KS88]).

3.2 Knowledge

In this section we will relate the memory model used in our model to results from cognitive science (especially the PUPS-theory, cf. [And89b], [Gos92]).

3.2.1 Outline of the Memory Model

In the model shown in figure 3.1 several categories of knowledge can be distinguished:

External knowledge: Information which is not known to the problem solver at the outset. This includes information encoded in the environment, which is accessible either directly (by listening or looking) or indirectly (e.g., laws of nature or information another person possesses). Especially, some features of the environment can act as stimuli and evoke associations.

Internal knowledge:

Background knowledge: All the knowledge contained in the long term memory of the problem solver.[3] In case the problem solver is human, knowledge about many different domains is available, but only a small fragment of this knowledge is useful for solving the problem at hand.

Active knowledge: Knowledge which has been recalled from long term memory or which has been extracted from the environment. Information about the immediate environment also belongs to this category. Only this knowledge is readily available during the course of problem solving. Therefore, in order to solve a problem, relevant knowledge must first become activated.

Because — especially during the incubation stage — work is in progress on several different problems, the active knowledge can be regarded as being partitioned into multiple problem solving contexts, one for each problem on which problem solving currently is in progress. However, after changing from one problem to another, knowledge which was used in the prior context can easily be remembered in the current one, too, because it is still highly active (see 'strengthening' below).

For the sake of clearness, we distinguish three different categories (per problem solving context) of active knowledge according to the role it plays:

Problem specification: The problem on which work is currently in progress. This includes all criteria (required or desired) which are used for judging a product.

Hypotheses: Different possible (partial) solutions or explanations, which have not yet been rejected.

Background knowledge: Other knowledge which is perceived as being related to the current problem: facts, rules, etc. In particular, knowledge about applicable strategies belong into this category.

[3] Knowledge, which becomes activated is not deleted from the background knowledge. Consequently, (with respect to informational content) background knowledge and active knowledge may overlap. See the glossary (appendix C) for a more detailed definition of this term.

3.2.2 Relation to Other Memory Models

Due to the aim of this study, there are several differences between this memory model and the memory model used in the PUPS-model:

- The PUPS-model differentiates between procedural and declarative knowledge, while this model does not.
- In the model used here, the active knowledge is further subdivided, while in the PUPS-model the working memory is simply an aggregation of knowledge items.

In addition to these direct opposites, there are some differences in emphasis, too. One such difference is the difference between strengthening (in the PUPS-model) and problem solving contexts (in the model employed in this study).

Anderson ([And89b, p. 322]) proposes that knowledge is strengthened by repeated use. That is, every use of a knowledge item increases the probability that it is selected in a future recall. The strength of such an item is given by the following formula:

$$S = \sum_{i=1}^{n} t_i^{-d}. \tag{3.1}$$

In this formula S is the total strength of the item, t_i is the elapsed time since the i-th use of an item and d is a decay rate[4]. Here use denotes for a rule, that its preconditions are satisfied, and for a fact, that it is used to satisfy the precondition of a rule.

Anderson claims, that function 3.1 "can be derived as an optimal solution to the information retrieval demands facing the human" (cf. [And89b, p. 322]). He supports this claim by the following three arguments (among others):

1. Due to the strong recency component information is readily available if two uses are in short succession.
2. Knowledge, which has proven successful multiple times in the past, can be easily retrieved.
3. In the long-term, long-term tendencies dominate, because strengthening due to recent use of the knowledge decays more rapidly.

The context switch, which was mentioned above, can be described by strengthening alone. This is due to the fact, that knowledge, which is no longer used for solving the new problem, will simply loose strength. However, immediately after changing to a new problem, information, which is related to the old one, can easily be recalled. Whenever we will use the term *context-switch* in the future, this can be regarded as switching to a new problem combined with strengthening, too.

However, it should be noted that the idea of switching between different groups of knowledge items during the problem solving process, has already been introduced by the SOAR-model (cf. [SNK91]). This substantiates our claim that the model employed here can be regarded as a valid computational model of human problem solving behaviour.

3.2.3 Experience in Creative Problem Solving

Some authors emphasize the importance of experience in (creative) problem solving. As creative problem solving typically involves the solution of a new kind of problem or the development of a novel problem solving approach, no experience can exist with respect to this problem solving approach. However, experience with related approaches/problems is usually be necessary. Further,

[4]Given the theory of Mednick (cf. page 9) it seems a good hypothesis to assume that d is smaller for creative people than for non-creative people. However, we do not know of any experimental validation of this hypothesis.

the preparation and incubation stages are important for the acquisition of experience and the corresponding reorganization of the knowledge base.

The development of experience extends into two areas: changes in the knowledge base (organization principles, available indices, ...) and the acquisition of control knowledge (strategies, heuristics, ...).

The knowledge base of an expert is structured according to the following criteria (as opposed to that of a novice):

- It is organized according to some abstract properties (e.g., energy) and not to surface properties (e.g. inclined plane) (cf. [ES90, p. 581], [Sch93, p. 285]).
- It is more organized for use (cf. [Sch93, p. 285]).
- It contains much knowledge about appropriate cases (cf. [Sch93, p. 303]).[5]

These differences in knowledge are accompanied by differences in problem solving behaviour: Novices mostly use weak methods (e.g., progressive deepening, mental simulation, etc.), but they usually have problems recalling the necessary information from memory (cf. [Sch93, p. 303]). The weak methods are regarded as innate knowledge by Anderson (cf. [And89b, p. 348]). With the acquisition of experience new heuristics and strategies are developed. These can be helpful even when they are used in a domain (slightly) different from that in which they were acquired (cf. [Sch93, p. 285]).

3.3 Problem recognition

In psychological experiments the problem is usually completely specified at the outset. Therefore only little experimental evidence exists for certain strategies of problem discovery. Nevertheless, this point is often emphasized as being important for creativity.

A hint to a problem can be that facts do no longer correspond to the relevant theory (cf. [Ulm68, p. 22]), e.g., in natural science. The absence of observations can pose a problem, too. For example, in particle physics the necessity of introducing additional quantum numbers was indicated by the absence of some reactions (cf. [Kok91]). Another source for problems is the combination of different theories (cf. [Ulm68, p. 23]). McLaughlin regards these problem sources as basic motivations of problem solvers: "Finally, a tendency to seek new information coupled with a need to maintain a consistent world view would explain the motivation behind developing this world view in such a way that new information that conflicted with the old world view was consistent with the updated world view." (cf. [McL92, p. 60]).

When testing for 'sensitivity for problems' using the Guilford test of creativity a different class of problems is used: inadequacies of household utensils. Although this is a standard question for tests of creativity no description of the basic strategies used for answering it could be found. However, it seems that the association of negative experiences combined with mental simulation is the most probable technique.

However, some points are still open: About which object will be asked "how can it be improved?", which theories are combined? Curiosity is generally accepted as a driving force in the selection of problems. We will return to this point in section 3.8 'control'.

3.4 Search for Knowledge

The task of this process is to extract knowledge from the environment for the induction of new ideas and the completion of sketches. While the first goal is especially important in the preparation

[5] Cases are usually less important for creative problem solving, as creativity is only necessary, if no directly applicable case is available. However, cases may be useful as a basis for cross-domain analogy or CBR, involving non-trivial adaptations.

stage, the second is important in the verification stage.

Two different techniques can be distinguished:

1. Questions
2. Experiments

However, this difference is only externally visible, internally both can be regarded as questions (i.e., desire for information). For experiments the transformation process is more complex than for questions. But for questions the mapping between internal and external representation is also not necessarily one-to-one. The search for knowledge seems to be very important for the preparation stage and the creative process in general, because Seiffge-Krenke mentions in [SK74, p. 68], that a major difference between creative people and other people is, that the former have a larger desire for information and actively search for it.

The ability to form questions (internally and externally) is widely regarded as an essential part of creativity. It is possible to learn a strategy of questioning by practicing. Moreover, practice is crucial for learning such strategies, although they can be further improved with a special training (cf. [SK74, p. 78]).

Questions and experiments are also used in the verification stage, because sometimes feedback from the environment is the only possibility for verifying whether an illumination solves the problem.

In the area of *cognitive science* much research deals with scientific discovery. In this field, the selection of experiments is important. However, only few studies in this area examine experiments which are conducted without a hypothesis (cf. page 12), although these experiments are especially important to the preparation stage.

An exception from this rule is the work done by Klahr and Dunbar ([KD88]). They describe scientific discovery as a combination of search in the experiment space and in the hypothesis space. In particular, it is possible that experiments are conducted without an explicit hypothesis, but only for inducing new hypotheses. The authors could observe such behaviour with subjects who could not think of an explanation that would be in accordance with the experimental results observed so far.

> The task their subjects had to perform was to find out what function a key labeled `RPT` on the steering of a programmable robot (BigTrak) had. The steering had multiple buttons. Besides the `RPT`-key there were also keys for making the robot turn right ($\rightarrow$), move forward ($\uparrow$), etc. Each function took a numerical argument with the consequence that the corresponding function is repeated n-times.
>
> The functions of all the keys except the `RPT`-key were known to the subjects. The function, which was attached to the `RPT`-key took one argument n, too. It lead to one repetition of the n previous instructions in the program of the robot.

In this study, experiments could be characterized by the program-length (λ) and the argument to the `RPT`-function (N). This two-dimensional space was called *experiment space* by Klahr and Dunbar. It could be observed that eight different hypotheses made up for most of the experimentation. The experiment space could be partitioned depending on the hypotheses which were compatible with the experiments in this segment. Experiments in segment III ($\lambda > 3,\ 1 < N < \lambda$) were only consistent with the correct hypothesis, while experiments in the other segments were consistent with multiple hypotheses. This illustrates, that experiments can produce different amounts of information depending on their position in the experiment space. Some subjects could find the correct hypothesis only after making an experiment in segment III, while others could find the correct hypothesis by conducting a search of the hypothesis space.

In a second study the authors asked their subjects to think of as many functions the `RPT`-button could trigger as possible. This provided their subjects with several hypotheses and enabled them to use experiments which discriminated between hypotheses. Thereby, subjects in study 2 needed much

less experimentation to discover the correct function than subjects in study 1 (cf. [KD88, p.30]). This shows that the knowledge about possible answers can be used advantageously in this process.

The initial hypothesis was generated by analogy with the other functions of the robot and the knowledge that `RPT` probably stands for *repeat*.

Klahr, Dunbar, and Fay (cf. [KDF90]) conducted a study in which the subjects were given an initial (wrong) hypothesis about the `RPT-key` in the BigTrak-environment. The aim of this study was to examine the heuristics used by the subjects for constructing experiments in more detail. They recognized several heuristics (cf. [KDF90, p.388]):[6]

- *Maintain observability.*
  Construct experiments, such that the results are easy to see and to distinguish from other possible results.
- *Design experiments giving 'characteristic' results.*
  The experiments should generate results, such that each part of the result can easily be attributed to some part of the experiment setup. For example, in the BigTrak setup all the actions should be different, such that the results can be easily attributed to the instructions.
- *Focus on one dimension of an hypothesis.*
  Only one aspect of a hypothesis is examined at one time and only the corresponding changes to the experiment setup are made.

3.5 Activation of Knowledge

The task of the "activation of knowledge" is to make that knowledge active, which can help to solve the problem. In particular, it deactivates obstructive knowledge. This process is mainly used in the preparation and incubation stages. However, mental simulation is used in the verification stage, too. But there it is not used for activating knowledge but for evaluating ideas. The following enumeration of operators is not complete. It is only a collection of techniques deemed important in psychological literature.

3.5.1 Mental Simulation

Mental simulation can be regarded as some kind of internalized experimentation. Due to this internalization no principally new knowledge can be generated. It is, however, a method for recalling pieces of knowledge in an orderly manner. The knowledge recalled includes function principles, possible problems, or interesting properties of an idea. Additionally, some kind of inferencing is done leading to a certain amount of restructuring. Therefore, it performs two functions: it activates knowledge and it is a problem solving technique. With respect to the creative process the first function seems to be more important.

Usually mental simulation together with hill-climbing, progressive deepening, means-end analysis, etc. are considered as weak methods. Schraagen ([Sch93]) conducted a study in which subjects had to devise experiments in the area of gustatory research. He observed that subjects who had experience in experiment design in general but not in experiment design for gustatory research usually used progressive deepening and mental simulation. Experts with experience in the problem domain simply reinstantiated already known experiments. On the other hand, novices were unable to use these techniques successfully, because they had to many problems recalling the necessary information. Kolodner et al. ([KW93, p.55]) point out that mental simulation is used both for the elaboration of the problem specification and for evaluation. This is possible, because the subjects

[6] Some of these heuristics are not mentioned here, but instead in section 3.7, as this is a more appropriate place for them.

get a very detailed description of the effects of their ideas, making them pay attention to details they otherwise would not notice.

Mental simulation can be used for setting up expectations about the outcome of an experiment (cf. [KD88]), too. Moreover, this technique is probably necessary for constructing experiments, but none of the psychological studies we evaluated examined the details of experiment design.

3.5.2 Recalling by Association

An association between two items exists, if the recall of one item leads to the remembrance of the second one, or at least promotes it (cf. [DHS87]). Recalling some information by association can be triggered by an external stimulus. However, experiments have shown that subjects are sometimes not able to remember the triggering stimuli (cf. [Wei89, p. 49]).

Associations are generally regarded as being important in the incubation stage. But there is no accordance about the nature of these associations. While Mednick holds the belief that associations are determined by the problem specification, Maltzmann assumes free association and Campbell even argues in favor of blind variation (cf. [Ulm68, p. 28]).

The concept of blind variation has many similarities with the bisociation theory proposed by Koestler. It is based on the idea that seemingly unrelated facts are suddenly perceived as being related (cf. [Wei89, p. 40–43]). Weisberg illustrates this theory with Gutenberg's invention of the printing-press. He argues that it originated as a combination of the concepts of a seal and a winepress: When Gutenberg visited a wine festival he examined a winepress and realized that a similar device could be used to press many letter seals onto paper.[7]
This can also be described in the model used in this section:

> Gutenberg had discovered the (sub-)problem to press the seals with homogeneous pressure onto paper. When he saw the winepress, it was associated with the concept of pressure and added to the active knowledge. Later the knowledge was restructured, resulting in a combination of both concepts (cf. section 3.6.5).

Several criteria are mentioned for the generation of associations (cf. [DHS87]):

Contiguity: The association is due to spatial or temporal closeness.

Part/Whole: A part of a concept or something it is part of is associated with the concept.

Similarity/Contrast: The association is due to a similarity or a contrast of two concepts.

Cause-effect: Facts which are related by a cause-effect-relation are associated.

Besides those primary association laws there are also secondary association laws. These are conditions which are not sufficient for creating an association but facilitate this considerably:

1. vivacity
2. recentness
3. frequent repetitions

Criteria 2 and 3 are already subsumed by formula 3.1.

Besides the direct association of two items it is also possible that the association is mediated by other items (cf. [Ulm68, p. 28], [McL92, p. 61]). The PUPS-theory proposes a spreading-activation scheme for selecting the base of an analogical transfer, too.

[7] Obviously, Weisberg is wrong in this description of the invention of the printing press, as the printing press was already known before Gutenberg invented movable letters. However, this does not invalidate the argumentation that bisociations as proposed by Koestler can be simulated in the model used in this section, too.

However, not every association, which is possible according to the rules above, is used: Only the useful ones are tried out (Bruner in [Ulm68, p. 28]). According to [SK74, p. 31] the *associations are selected critically and realistically*. Campbell (cf. [Ulm68, p. 29]) assumes a continuous selection process.

3.5.3 Forgetting

This operator eliminates knowledge from the active knowledge and thereby reduces the search space for many operators, which are restricted to the active knowledge (cf. table 3.2). If this knowledge is not encoded in the background knowledge, too, it is lost for the problem solver. Otherwise it can again become active. Many psychological experiments support the view, that knowledge does not get lost, even if it cannot be remembered (cf. [And89a]). However, the subtle distinction between inaccessible knowledge and knowledge that does not exist in the knowledge base will not effect our discussion of problem solving behaviour.

Formula 3.1 already gives a description of forgetting, because a decrease in strength corresponds to forgetting. Another approach is described in [Cox94]. Here the author describes the model of forgetting by inference, which is also used in psychology. This model describes the retrieval of information from memory by the association of a context A with a cue context. If a new context B is stored, which is also associated with the cue context, then in the future A and B will compete for retrieval. If B is stronger associated with the cue context than A is, then this may effectively lead to the forgetting of A. According to Anderson ([And89a]) such an interference effect also increases the probability that both A and B are forgotten. Additionally, the time needed for reproducing either A or B is increased due to this interference.

Seiffge-Krenke ([SK74]) points out the importance of forgetting for overcoming states of fixation on past experiences: "The more time elapsed between the usual and the unusual use of an object, the smaller was the fixation on the former use."[8]

In principal, forgetting can be understood as the ability to reduce sets[9]. That is, forgetting reduces the positive as well as the negative effects of sets. People differ in their ability to forget. However, forgetting must not be confused with the ability to overcome sets, because it is also possible to overcome a set, if the knowledge is still active. The negative effects sets have on a persons problem solving behaviour differ from person to person (cf. [SK74, p. 91]). However, forgetting the knowledge, which constitutes a set, always leads to overcoming it.

The difference between forgetting and exclusion of knowledge due to a negative evaluation (cf. section 3.7) are twofold:

1. The evaluation is made as soon as knowledge shall be added to the active knowledge, while forgetting eliminates knowledge from further consideration, which is already part of the active knowledge.

2. While forgetting is only based on *uses* of knowledge (independent of their outcome), during evaluation an attempt is made to forecast the *usefulness* of some knowledge.

3.6 Restructuring of Knowledge

In this section we will describe various techniques for restructuring. Among these are transfer, induction, adaptation, concept formation, etc. But before we begin with the detailed discussion of these operators, we will first discuss the central role that restructuring plays in the creative process.

[8] Je mehr Zeit zwischen der gewöhnlichen und der ungewöhnlichen Verwendung von Gegenständen verstrichen war, desto geringer erwies sich die funktionale Gebundenheit der Gegenstände.

[9] A set is a stimulus-reaction-scheme.

3.6.1 The Importance of Restructuring

Illumination plays a crucial role within the creative process. It is usually regarded as the result of a restructuring step, that is, the information at hand is combined in a new way. Therefore, restructuring is central to the novelty of the creative product and thereby to the generation of emergent value (cf. [ECJS94]).

We want to illustrate the process of restructuring with an example from [KD88]:

The subjects were given the task to find out what function a button labeled `RPT` had. Similar to the experiment space described on page 21 a *hypothesis space* can be identified.

The hypothesis space (comprising only the eight common hypotheses) can be partitioned into two main segments, depending on the role attributed to the argument N of the `RPT`-function[10]:

1. N can denote the number of repetitions (NROLE: Counter)
2. N can denote the range of repetition (NROLE: Selector)

The separation between these two segments is due to the facts that some variables become constants when going from one partition to another (e.g., the number of repetitions is set to 1 when choosing NROLE: Selector) and other variables change their range (e.g., the unit of repetition can either be a step or the whole program when NROLE: Counter and a step or a segment when NROLE: Selector).

Therefore, going from one segment to another in the hypothesis space can be regarded as a major restructuring. Similarly, changes of the hypothesis within one partition can be regarded as a minor restructuring. This view is shared by Margaret Boden (cf. [Bod91, p. 100]), who wrote: "to transform a frame by changing a low level slot-filler is less creative than to redefine the frame at a higher level".

Subjects change their hypothesis when enough evidence against it has been collected. However, that change is usually not made as soon as the first negative evidence is encountered. Instead, subjects have a certain tolerance to negative outcomes to account for any misbehaviour of the system or for any errors they make. Additionally, subjects are more willing to abandon a hypothesis they hold, if they are aware of another hypothesis which accounts for the result of the experiment.

In *creative design* often a different kind of restructuring is required: there is no change of the hypothesis due to the result of an experiment, but due to some interesting quality of a possible solution the problem specification is changed (cf. [Kol94], [Nav91]).

For the area of *creative design*, Rosenman and Gero ([RG92]) mention four basic principles for constructing creative products:

1. Mutation
2. Combination
3. Analogy
4. First principles

Analogy will be discussed in section 3.6.2.1. Mutation and combination can be regarded as adaptation techniques. First principles describes the construction of a product by searching the solution space with the problem specification as a goal.

However, no universal description of the processes which can lead to a restructuring exists. Therefore, we will now describe some methods, which are mentioned in literature or are obvious from examples.

[10] In principle, this hypothesis space can be embedded in a higher-dimensional space, for example, by making the number of arguments to the function a variable. This was not done here, because the subjects inferred in analogy to the other functions that `RPT` has only one argument.

3.6.2 Knowledge Transfer and Rule Induction

The transfer of knowledge from one situation to another and the induction of rules are tightly interconnected as a transfer can always be described as an instantiation of a rule generalized from an earlier situation.

However, knowledge transfer and rule induction are usually treated independently. This holds true for many studies in psychology as well as in AI. Therefore, we will start with a description of transfer first and will then show how this can be extended to a method of rule induction.

3.6.2.1 Knowledge Transfer

As Clement points out, analogical reasoning (knowledge transfer) is a technique, which is commonly used in scientific problem solving (cf. [Cle88]). While most approaches to knowledge transfer are based on the assumption that knowledge is transferred from well-known previous situations, his experiments showed, that the source situation is often specifically created for the purpose of transferring knowledge from it. In his experiments only 8 out of 31 analogies were based on previous situations, in the remaining situations a problem, which can be solved more easily, was generated first and then knowledge was transferred from its solution. A well-known example, where such an approach is helpful, is the multilated checkerboard problem[11], which becomes trivial when it is seen as the match-making problem[12] (cf. [Fis92]).

However, in most studies on knowledge transfer the transfer is based on previously solved situations. Under these conditions the success of the knowledge transfer depends crucially on the ability to find an appropriate source situation for the transfer. *In our model the retrieval mechanism is the association of knowledge*. The association of the source and the target situation need not be based on the transfer of words, objects, etc., but can be limited to the relations between elements (cf. [SK74, p. 83–86]).

With respect to the elements of the situation which are transferred, some variations are possible, too. It is possible, that the same reaction as in the source situation is made, certain elements or relations can be transferred, or the proceeding in the source situation is transferred (so-called non-specific transfer).

With the PUPS-theory ([And89b]), Anderson tried to give a model, which is able to explain how the various kinds of transfer are made. The ability to react to a stimulus in the same way as has proven successful on an earlier occasion is modeled by the formation of a production rule. A chain of rule-applications can also be made into a new rule (knowledge compilation). This models the ability to automate, which is assumed to be a necessary part of intelligence by Sternberg (cf. page 11).

More complex kinds of transfer are described with analogy in the PUPS-model. The central idea of this theory of analogy is the *no-function-in-identity principle*:

> If there is a production rule `IF A THEN B` and it shall be applied to a situation `A'`, then this can be done if there exists a substitution (from constants to constants) such that:
>
> 1. every constant, which is replaced occurs in `A` and `B` *and*
> 2. after applying the substitution the resulting rule matches the current situation `A'`.

In situations in which in `A` occurs a function (e.g., add) and in `B` an operator (e.g., +) the formation of an analogy is not possible due to restriction 1. This is the reason why the PUPS-theory includes the *principle of functional elaboration*. This allows to substitute the operator with

[11] *The multilated checker board problem.* A checkerboard with 64 squares and a set of 32 rectangular dominoes are given. Obviously, the 32 dominoes can be arranged to cover the board completely. Now suppose that two (black) squares were cut from opposite corners of the board. Can the remaining 62 squares of the board be covered using exactly 31 dominoes?

[12] *The match-making problem.* In a village there are 32 bachelors and 32 unmarried women. Through tireless efforts, the village matchmaker succeeded in arranging 32 marriages. Then one night, two drunken bachelors fatally stabbed each other. Is it possible to arrange 31 marriages among the 62 survivors?

an explicit relation between operator and function. Thereby, the function occurs in `B`, too, and can therefore be replaced. In general, this principle allows to introduce any relation which is necessary to comply with restrictions 1 and 2 (cf. [And89b, p. 334]). That is, the principle allows to use domain knowledge to enrich the representation in such a way, that a transfer is feasible. The following example shows an application of this principle (cf. [And89b, p. 331]). Here the relation 'implements' is introduced and `=function` denotes a variable:

`IF` goal is to achieve the function `(add 2 3)`
  and the context is `COMMON LISP`
`THEN` use the form `(list =function 2 3)`
    where `=function` implements `add`.

In addition to the two principles mentioned above, Anderson introduced the *form-to-function-principle* and the *function-to-function-principle*. While the first principle relates a form to a function, just as the production rule above relates a function to a form, the second produces a production rule, which relates two interpretations of a function (e.g., `(second list)` and `(first (rest list))`).

Obviously, analogical transfer need not produce correct results. This can again be illustrated using the BIGTRAK-example. Its steering has buttons labeled $\uparrow$, $\leftarrow$, $\downarrow$, and $\rightarrow$. If it is known that the $\uparrow$-button makes the tank move forward it can be inferred with the principles mentioned above that the $\downarrow$-button makes the tank move backward and the $\leftarrow$-button makes the tank move to the left. However, only the first conclusion correct, because the $\leftarrow$-button makes the tank turn left without moving. Interesting about this example is that subjects made exactly this misanalogy in a psychological experiment (cf. [And89b, p. 334]).

3.6.2.2 Rule Induction

Instead of assuming a transfer between source and target situation, one can assume the induction of a generalized rule. Indeed there are some reasons for this view. For example, humans are able to form a generalization based on only one example (cf. [And89b, p. 328]).

The PUPS-model employs exactly this view. Instead of using the analogy-substitution as described above, a substitution is formed in which every constant which shall be substituted is replaced by a variable. This substitution is then used for forming a generalized production rule, which in turn is applied to the problem. However, no more constants than necessary for forming the transfer are replaced by variables (cf. [And89b, p. 330]).

Besides the generalization of a transfer it is also possible to induce a rule purely based on some examples. A very well known example for this is the recognition of a regularity underlying a sequence of numbers, e.g., given the sequence $3-5-7$ possible regularities are: only odd numbers, only prime numbers, etc. As this example shows the regularity is usually not uniquely determined by the examples. Often rule induction is regarded as concept formation, i.e., find the concept which covers all the examples.

However, in this study we will distinguish between concept formation and rule induction (concept characterization) as follows: the term *concept formation* will be used for the unsupervised aggregation of items into a concept, while *rule induction* describes the generation of rules that allow to classify a new item as belonging to one of these concepts.

Both abilities together enable people to recognize redundancy in the information they possess. In the information theoretical interpretation of intelligence this ability is regarded as being very important. Hofstätter (cf. [ROD80, p. 91]) even defines intelligence as the ability to recognize redundancy.

Anderson (cf. [And89a]) distinguishes three different types of concept descriptions (rules):

- conjunctive concepts — e.g., a cross and a circle[13]

[13] This type of concepts is most easily recognized by humans.

Classification	positive example	negative example
correct	keep the current hypothesis	keep the current hypothesis
wrong	the new hypothesis is given by the common features of the old hypothesis and the current example	impossible, if no mistake has been made

Table 3.3: The holistic strategy

Classification	positive example	negative example
correct	keep the current hypothesis	keep the current hypothesis
wrong	change the hypothesis such that it is compatible with the preceding examples, i.e., choose a hypothesis which has not been refuted yet	change the hypothesis such that it is compatible with the preceding examples, i.e. choose a hypothesis which has not been refuted yet

Table 3.4: The selective scanning strategy

- disjunctive concepts — e.g., a cross *or* a circle
- relational concepts — e.g., there is the *same* number of circles and of squares

Finding a regularity is composed of different subtasks:

1. Finding the relevant features
2. Finding the rule which connects these features

Because it is rather difficult to solve both tasks at once, usually either the relevant features or the kind of regularity searched for is usually given to the subjects.

When the task is given to find a conjunctive concept description from a sequence of examples, which are either classified as being positive or negative examples then most subjects use (or try to use) one of the following two strategies (cf. [And89a, p. 264]):

- In the so-called *holistic strategy* people use the first (positive) example as their initial hypothesis. When the next example is presented they perform one of the actions given in table 3.3. This strategy was used by approximately 2/3 of the subjects.
- A subject employing the *selective-scanning-strategy* uses as his initial hypothesis only the subset of the features of the initial example, which he considers as being important. When additional examples are presented he performs the actions described in table 3.4. The main problem of this strategy is that the subject must verify a new hypothesis against all previous examples if the initial set of features is chosen wrong. At this point most subjects failed.

Different problems arise if the regularities searched for are not strict, but involve probabilistic reasoning. One can regard the Bayes-theorem as a normative model for reasoning in such situations (cf. [And89a]). However, humans show systematic derivations from this model. Some of these deviations are due to the fact that humans typically use an assessment of the share of examples which belongs to a certain group for estimating a probability. For example, when confronted with the task to estimate the ratio of words beginning with 'k' and words with 'k' at the third position,

people systematically overestimate the number of words beginning with 'k', as these can be more easily remembered.

Other errors include:

- Sometimes a-priori-probabilities are completely ignored.
- People tend to employ a rule of averaging: if a large number of events of one kind has occurred they believe that a different kind of event becomes more likely.
- Often people are confused by the similarity of events. Consequently, many people believe that the sequence $1-1-1-1$ is less likely than the sequence $1-3-6-2$ when throwing a dice. They confuse the probability for the sequence $1-3-6-2$ with the probability that an irregular sequence like $1-3-6-2$ occurs.

Up to now we examined the induction of arbitrary regularities. However, there is some kind of regularity which is of especial importance to human reasoning abilities: causal relations. A causal relation can be induced from observations of the environment and is believed to describe a fixed behaviour of the environment. The discovery of causal relations is also called *causal induction*. The PUPS-theory proposes three principles for performing causal induction:

Identity heuristic: If the same token appears twice in a sequence then the first appearance somehow caused the second one.

Previous action: If an event has no other appearant cause then ascribe its cause to the immediately preceding action.

Principle of minimal contrast: This principle can be used to compare two sequences of events and thereby induce causal relations: the first sequence shall contain the events A and B (in this order) and the second shall contain the events A' and B'. Furthermore, A and A' shall only differ in a characteristic c and B and B' shall only differ in characteristic d.[14] In this case we can induce, that the expression of characteristic c somehow causes the expression of d.

These causal relations can be used with the principle of functional elaboration to enable a transfer of knowledge, too.

According to Wisniewski and Medin (cf. [WM91]), the view that rule induction is purely based upon the presented examples is overly simplistic. Based on their experiments they propose that when subjects produce classification rules for a category, then the examples for the category tightly interact with the theories the subjects have. Additionally, these theories influence the recognition of features: whether a subject pays attention to a certain feature, which abstractions of features are formed, and how the example is interpreted (e.g., a part of a drawing may either be interpreted as a harpoon or as a long stick).

3.6.3 Adaptation

The term *adaptation* is often used to denote any adaptation of existing knowledge to a new situation. In this sense transfer can also be regarded as some kind of adaptation. However, here we will use the term adaptation only for describing two activities:

1. The adaptation of a preliminary problem solution, which is not completely adequate for solving the problem. For example, if the result of a transfer does not directly solve the problem, but needs further modifications.

[14] Suppose, we have a frame representation of the events, then the expression of a characteristic can be interpreted as the value of a slot or as the values of a group of slots. Note, that the applicability of this principle depends on the representation of the events.

2. If a general rule is contradicted by an example and must therefore be revised.

The adaptation of a problem solution that is remembered from a previous problem solving experience or that is generated by transfer is often necessary, even if elaboration techniques as those described in section 3.6.2.2 are employed. However, only little could be found in psychological literature about this activity, as it is often indistinguishable from the surrounding activities (e.g., transfer). Consequently, often the term *restructuring* is used for denoting adaptation *together* with one of the other activities.

An important technique for performing adaptation steps seems to be hill-climbing (cf. [And89a, p. 194]). However, other general problem solving techniques like means-end-analysis can also be used for adaptation.

Various authors note that it is often important for arriving at a creative answer to redraw assumptions made, as they may hinder problem solving (cf. [And89a, SK74, Wei89]). This may even include the commitment for a certain problem solving method, as, for example, the assumption that the problem can be solved by adapting a previous problem solution may be wrong. This is also called *overcoming states of fixation*. Rephrasing steps can often be used for adaptation, too (e.g., change in the perceived function of parts, cf. section 3.6.5).

An adaptation of a general rule is necessary if new information that is in conflict with it becomes available. In the PUPS-model a transfer is described by the instantiation of a general production rule. Consequently, an unsuccessful transfer can point to the necessity of adapting a rule. An example for this is the misanalogy described on page 27. In this case the production rule is too general and must be made more specific. This is also called *discrimination* of a rule.

The rules in the PUPS-model are represented as frames with a precondition-slot. This slot describes general conditions of applicability of the rule (e.g., the context). If an erroneous application of a rule is encountered then some constraints are added to the precondition-slot.[15] The additional constraints can be selected from current experiences or by comparison with earlier applications (cf. [And89b, p. 337]). From these possibilities a discrimination condition is chosen randomly, where the probability for selection is proportional to the strength of the knowledge items (cf. [Gos92, p. 17]).

A similar kind of adaptation was already illustrated in tables 3.3 and 3.4. Here, if a misclassification is made the concept description is generalized if it is too specific and specialized (discriminated) if it is too general. The latter can only occur with the selective-scanning-strategy, because the holistic strategy always uses the most specialized hypothesis which is compatible with the examples.

When adapting probabilities in the light of a new event this should be done according to the Bayes-theorem. However, people tend to deviate systematically from this norm. Usually, they adjust their probability estimate *conservatively*, i.e., they change it in the correct direction, but not to the full extent, which is required by the Bayes-theorem.

When the subjects hold prior theories about how a classification rule should look and the examples are complex enough, then the subjects have a wider range of possibilities than simply adapting their rules. Wisniewski and Medin (cf. [WM91]) observed three different strategies their subjects employed when told that their classification was incorrect:[16]

1. *The subjects used theory-based features in context-specific ways.*
   For example, a subject should classify a drawing as being drawn by a creative or a non-creative child. He used the rule: a creative child makes more detailed drawings. Upon misclassifying a drawing as being made by a creative child he concluded that a creative child would make *more detailed* drawings, i.e., the subject adapted the interpretation of the feature *detailed*, not the rule itself.

[15] On page 25 we pointed out already, that humans do not always change their hypothesis immediately, if negative evidence is found.

[16] The subjects should give classification rules and should classify children drawings using these rules. Two different categories existed. These categories had meaningful labels, like *drawn by creative/non-creative children*. This leads the subjects to using their background theories in generating the rules (this was shown by an additional experiment).

A second subject demonstrated rule-refinement. He should classify a drawing as being made either by a farm or by a city child. He concluded that the drawing was made by a city child, because it showed much detail. When told, that it was drawn by a farm child, he refined his rules: detail in clothing points to city children, while detail in body movement points to farm children. That is, although he refined his rules, he stuck to his theory-induced belief that detail is an adequate feature for distinguishing between these two types of drawings.

2. *Reinterpreting features to preserve theory-based belief.*
   In principle, this is no adaptation method, but a method for avoiding adaptation. The subject interpreted the clothing of the person in the drawing as a city uniform. Therefore, he concluded that the drawing was made by a city child. When told that it was made by a farm child, the subject reinterpreted the clothing as a farm uniform.

3. *Shifting to evidence that supports alternative theory-based beliefs.*
   Upon negative feedback the subject constructs a new explanation based on different features of the drawing that supports an alternate view. Similarly to strategy 2, this can be regarded as a strategy for avoiding adaptation.

Additionally, subjects reused the explanations that have been successful previously, while they avoided previously unsuccessful explanations.

Wisniewski and Medin hold the view that the above strategies are important, because people need not only adapt their theories, but also their interpretation and recognition of features (e.g., what is a detailed drawing).

Besides the adaptations described above, adaptations of the problem specification play an important role in creativity, too. Often it is regarded as characteristic of creative problem solving that a refinement of the problem description is made (cf. [Wei89]). This view is especially prominent in the field of *creative design* (cf. [McL92]). McLaughlin regards the recognition of *emergent value* and subsequent adaptations of the problem specification for taking advantage of these opportunities as very important for the creative process. Other authors (e.g., Fischer, [Fis92]; Logan and Smithers, [LS92]) stress the importance of identifying problems and performing corresponding changes to the current problem specification, too. This refinement of the problem specification seems to be most prominent in the preparation stage. However, especially creative people show a high readiness for changing their current design even late in the problem solving process, if additional aspects come to their attention.

However, most psychological experiments are performed with detailed problem specifications which must not be altered. Consequently, no detailed description of the processes which may be responsible for changes in a problem specification exist in psychological literature.

3.6.4 Concept Formation

Rule induction and concept formation are closely related. Indeed, they are often identified in psychological literature. However, for clarity we will employ a different view here. We will use the term *concept formation* only for the selection of the examples which shall be grouped in one concept and regard the generation of a compact description of such a concept as *concept characterization*. Because concept characterization can be done using rule induction techniques, we need not enter into this topic here.

The aspect of finding the relevant examples for forming a new concept has received only limited attention in (cognitive) psychology. Probably, because it is rather difficult to examine experimentally.

However, it is a very important activity for human problem solving behaviour. This is emphasized by the fact that humans readily introduce new concepts during problem solving (e.g., *things that will sell at a garage sale*) (cf. [FP91]). Some authors even regard *concept formation* as *the* basic activity in *scientific discovery* (cf. [KD88, p. 5]).

The importance of concept formation is emphasized in the information theoretical interpretation of intelligence. There it is also called *macro formation*. Three forms of *macro formation* can be distinguished according to Roth et al. (cf. [ROD80, p. 98, 99]):

- Encoding of complex internal models (e.g., car instead of: motor, four wheels, ...).
- Formation of a class from elements with common properties, thereby neglecting unimportant details.
- Formation of relations and the recognition of regularities. (According to the view employed here this is concept characterization.)

Wallach and Kogan (cf. [Ulm68, p. 92]) could show that people who are very creative have a high tolerance with respect to diverging details. This surely facilitates concept formation for them.

Rhine (cf. [SK74, p. 107]) distinguishes between first order and second order concepts. A first order concept is defined as above; *a second order concept essentially is an attitude* and results from combining several first order concepts, one of which includes a judgement (e.g., bright and unpleasant).

3.6.5 Reformulation / Inference

3.6.5.1 Reformulation

Sometimes a person possesses the necessary knowledge for solving a problem, but is unable to put it to good use, because the form of the knowledge is inadequate for reasoning about the problem.

We presented an example of such a situation with our match problem on page 5. Here, everybody had the necessary knowledge for finding solution no. 5. However, nobody found this solution, because it is not obvious to rephrase the problem such that the winners of the matches are emphasized.

Several techniques for the reformulation of knowledge are mentioned in literature (cf. [SK74, p. 34], [Wei89, chap. 3]):

1. Modification of the accentuation (figure-ground-differentiation): recognition of structure in the available information.
2. Combination of several parts into a new whole.
3. Disintegration of a whole into parts.
4. Change in the perceived function of parts.

The task of redefinition (cf. page 10) can be regarded as an application of principles 3 and 4. An application of principles 2 and 3 is described by Wertheimer (cf. [Wei89, p. 61, 62]):

> Some children were taught how to calculate the size of a rectangle with the help of small squares. Then they got the task to calculate the size of a parallelogram. Most children gave up. However, one girl recognized that the corners were problematic. With the help of a scissor she transformed the task as shown in figure 3.2.

The step from A to B can be regarded as an application of principle 3 and the step from B to C can be accounted for by principle 2.

3.6.5.2 Inference

While reformulation only changes the form of the knowledge, but not its contents, inference processes also add knowledge to the knowledge base.

Two basic approaches to inference can be distinguished: deduction and induction. We treated the second topic already in section 3.6.2.2. Deduction is usually regarded as being identical to

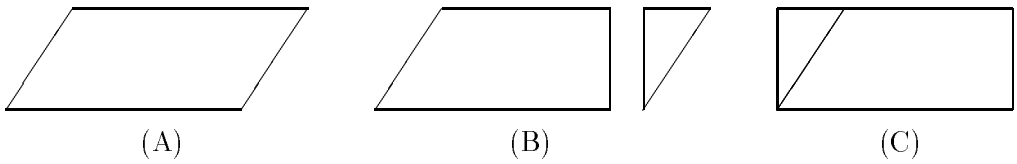

Figure 3.2: Transformation of a parallelogram into a rectangular

logic. However, when humans deduce facts they make mistakes in a systematic way. According to Anderson (cf. [And89a]), the human inability to use *modus tollens*[17] is especially prominent. However, people are often able to use *modus tollens* in special contexts, e.g., if this is necessary for recognizing fraud.

Some other systematic deviations of human behaviour from pure logic include:

- People tend to use implications as if they would hold both ways, i.e., given the implication $A \rightarrow B$ they also assume $B \rightarrow A$ would hold.
- Given formulas with quantors like 'all', 'some', 'no', etc. they tend to believe that a conclusion which contains the same quantors as the premises is correct (and one which does not is wrong). For example, given "no A is a B" and "no B is a C" many people regard "no A is a C" as a correct conclusion.
- In the interpretation of the quantors given above people often make additional assumptions, e.g., they often use "some As are Bs" as if besides $A \cap B \neq \emptyset$ also $A \cap \bar{B} \neq \emptyset$ and $\bar{A} \cap B \neq \emptyset$ would hold.

Most attempts for explaining this behaviour are based on the assumption that humans use heuristics for making such deductions. Some suggested heuristics are:

- People only use the quantors for judging the correctness of a rule (this heuristic gives the correct answer in more than 80% of all cases).
- The *conversion hypothesis* assumes that people replace the given formulas with stronger formulas which are easier to handle, e.g., they replace "all As are Bs" by "A and B is the same".
- People construct a micro world which is in accordance with the premises and verify the conclusion there.

Anderson assumes that the heuristics employed vary from person to person and from problem to problem.

Although heuristics can be regarded as a source of errors, we assume that they are very important for the ability to handle complex problems that humans show. The reason for this assumption is that is is usually less difficult to repair a faulty reasoning than to reason correctly at first try. Consequently, we think that it would also be advantageous for artificial creative problem solvers to possess such heuristics in order to be able to deal with complex problems.

3.6.6 Reindexing

The restructuring techniques mentioned up to now where concerned with the *form* of the knowledge. Now we will discuss a phenomenon which is important for the *accessibility* of the knowledge[18]: This

[17] Let be known that $A \rightarrow B$ and $\neg B$ hold, then it is possible to conclude that $\neg A$ holds, too.

[18] A more detailed discussion would have to take into account the fact that newly created active knowledge is randomly incorporated into the background knowledge.

is the *reminiscence phenomenon*. This term describes the effect that, after having a short break after a learning period more can be remembered than directly after the learning period (cf. [SK74, p.83]). In order for this phenomenon to occur it is necessary to have a rest, that is no conscious work on the problem is made. These phases of rest are typical for the incubation stage. Therefore, this phenomenon is regarded as an important explanation for the processes, which occur in this stage, especially in the learning theoretical approach.

We think that this phenomenon can easily be explained by an indexing process, which works subconsciously. This indexing can either happen with respect to existing indices or it can include the generation of new indices. The most probable explanation is a combined approach, in which new knowledge is indexed with respect to existing indices and new indices are generated either due to new regularities, which show up in the knowledge, or because a need for them arose in the past problem solving experiences.

3.7 Evaluation and Verification

The Wallas-model emphasizes the evaluation of the proposed solution (illumination) during the verification stage. However, every change in the active knowledge needs to be evaluated with respect to its implications for the solution of the problem. Due to this reason we introduced the evaluation process in figure 3.1. Every addition to the active knowledge is judged by this process. Despite the fact, that evaluation is a central part of the verification stage, evaluation and verification must not be identified, as is often the case in psychological literature. The illumination is no complete idea but only a sketch (cf. page 14). Therefore, the verification stage must add the necessary details to the already preliminary evaluated sketch and re-evaluate the completed product (cf. 'ability of elaboration' in the Guilford-model, page 10). In this context a complete product corresponds to a product, which complies with the criteria of a field for communicating a product. In architecture this may be a model together with accompanying plans, in mathematics this may be a proof, which is adequately formalized. The verification (including completion) of an idea often takes more time than the rest of the creative process.

Although, the verification stage is different in contents and aim from the previous stages, this difference can not necessarily be observed from the outside. For example, experiments are still performed; however, not for inducing new hypotheses, but for filling out the details and for testing the current hypothesis (cf. [KD88, p.32]).

The goal-oriented process of elaboration of the illumination is typical for the verification stage. Sometimes the evaluation process itself (even when called in other stages) can trigger some elaborations: Klahr and Dunbar ([KD88]) observed that sometimes their subjects made up their expectations for the outcome of an experiment *after* conducting it.

The task of the evaluation process is to estimate whether an intended modification of the active knowledge is useful for reaching the goal. Because this judgement is only based on sketches, it can only be of heuristical nature. In this process emergent value, which was generated during restructuring, must be recognized, too, because this may be important for changing the problem specification.

The judgement and selection of ideas in the evaluation process is an important part of the creative process. Seiffge-Krenke (cf. [SK74, p.31]) mentions especially the critical and realistic selection of associations, which creative people show. Weisberg also mentions an experiment, which emphasizes the fact that creative people possess a rich base of criteria for analysis (cf. [Wei89, p.161]):

> Several students were asked to draw a still life based on several objects given to them. Especially those students who had produced a highly rated picture held the opinion that their picture could still be improved and was not yet finished.

On page 8 it was already mentioned that creative people are usually more field-independent than non-creative people. This implies that their criteria are only weakly influenced by the environment.

In addition to these hints for a strict application of criteria there are also indications for a tolerant selection: e.g., the open-mindedness with respect to experiences, the tolerance to diverging details, and the readiness for risks as typical qualities of creative people (cf. section 2.3.2). Moreover, in [Wei89, chap. 4] precipitate judgement and the dismissal of ideas, which are false but capable of development, are mentioned as hindrances to creativity.

We do think that the key to this apparent contradiction is the abundance of criteria a creative problem solver possesses. Thereby, he can recognize that the idea does not comply with some criteria, but does accomplish other goals very well. This insight can then be used to extract certain features of the current idea for adding them to the future product. In particular, this may include the discovery of new criteria, which were not found earlier, but are sensible in the case at hand. These new criteria may then lead to a modification of the problem specification as was illustrated by the examples of Poincaré and of the Sydney Opera House (cf. page 7). The reformulation of the problem specification is also influenced by the attitudes the problem solver has (e.g., if the view over a valley is nice, then the living-room window should be in this direction).

In the BigTrak-experiment Klahr and Dunbar ([KD88, p. 36]) could observe two criteria for selecting hypotheses of a rather general nature:

Functionality: Does the hypothesis assign a function to every observed element?
e.g., hypotheses were often dismissed when they did not assign a meaning to the argument N of the `RPT`-function.

Plausibility: Hypotheses which perform some arbitrary arithmetic operation on N were often rejected.

Besides this, Klahr and Dunbar could observe some interesting behaviour of their subjects:

1. Usually the subjects did only try to find positive instances of their hypothesis.
2. A hypothesis was not always rejected if the outcome of an experiment was not in accordance with it.
3. Sometimes a hypothesis was rejected although the result of the last experiment was in accordance with it.

Klahr and Dunbar tried to explain this behaviour within the frame of their SDDS-model, which they gave in [KD88]:

1. This behaviour is useful if the number of positive instances is small in comparison to the total number of instances. In this case the probability is very small, that although the hypothesis is wrong the experiment complies with it. However, people use this strategy even when this precondition is not satisfied.
2. With their tolerance to negative outcomes people tried to allow for failures of their memory and of the experiment. This can be deduced from the fact that this tolerance is increased if the subjects are explicitly told that the possibility of failures exists (cf. [KD88, p. 43]).
3. The rejection of a confirmed hypothesis is due to the fact, that a second hypothesis is also confirmed, which is either more plausible or more interesting.

In more detailed experiments Klahr, Dunbar, and Fay (cf. [KDF90, p. 388]) came to the conclusion that the a priori strength of a hypothesis influences the experimentation strategy used: If the hypothesis is regarded as having a high strength, experiments are constructed in such a way that they shall confirm the hypothesis. Otherwise, experiments are constructed for disconfirming the hypothesis or for discriminating between multiple hypotheses.

In addition to the rating of the active knowledge, the evaluation process may have a second outcome: It can trigger new tasks. This is illustrated by the following examples:

- A product does not comply with the problem specification, but it satisfies one of the constraints in an excellent way. In this case an analysis of the product features which contribute to the excellent compliance would be helpful. Again, this can be a major task.

- If an experiment does not show the expected result, the cause of this phenomenon should be analyzed, as this promises new insights. Kulkarni and Simon identified this as an important heuristic employed by Hans Krebs in his research on the urea cycle.

  However, Klahr, Dunbar, and Fay (cf. [KDF90, p.388]) point out, that the surprising phenomenon must have a minimal magnitude or must affect an aspect the subject focuses upon at that time, if it shall be noticed.

3.8 Control

We mentioned already, that in the Wallas-model (cf. page 12) the different stages of the creative process do not necessarily occur in the given order. Usually, no stage is skipped, but it may come to repetitions in the sequence of stages. Especially, it is possible that the illumination is incorrect (although it is evaluated as being correct in the incubation stage), and is rejected in the verification stage. If this happens, usually, major restructurings are necessary (cf. [SK74, p. 17]).

With respect to the Wallas-model, some authors speak of an "interacting dynamic"[19] of the various stages. That is, nearly any transition between two stages is possible and these transitions occur in rapid succession. Neisser (cf. [SK74, p.33]) interprets these transitions as the result of multiple processes, where always the most dominant process becomes conscious. Besides repetition of stages recursion of the creative process is possible, too. For example, the construction of an experiment in the course of the development of a new scientific theory can be a creative act by itself.

The problems with defining a control-flow on the level of stages was one of the reasons for adding the process level and the operator level to figure 3.1. In this picture it can be seen, that the relation between stages and processes is not one-to-one.

Two aspects of control can be distinguished (cf. [Sch93, p. 287]):

- Which subproblem shall be solved? (Focus)

- How shall it be solved? (Selection of an operator)

Naturally, the aspect of focus is intimately related with the process of evaluation.

Creative problem solving often requires the subject to try out new paths. In order to do this, without getting lost in an abundance of possible paths, it is of decisive importance to distinguish between the ways to the goal and the dead-ends. This is were strategies and heuristic methods come in (cf. Newell, Shaw and Simon in [SK74, p.71]). An important problem of this approach is the possibility that the solution-space can become over-constrained. Creative people differ from other people, because they possess a wealth of heuristical methods and a learning process which produces new heuristical methods (cf. [SK74, p.73]). Campbell proposes the idea of "blind variation and selective memorizing"[20] as a possible implementation of such a learning process. Johnson observed that above all his subjects used strategies which had been successful earlier, and favored strategies which combined information from different steps. The results achieved by Harlow (cf. page 12) are another indication for such a process.

The heuristics and strategies employed in creative problem solving can be divided into two categories: domain-specific and general. Ulman ([Ulm68, p.27]) gives three examples of general strategies:

- conflict analysis (why does it not work?)

[19] interagierende Dynamik
[20] blinde Variation und selektives Behalten

- material analysis (what can be used?)
- goal analysis (what is needed and what can be dispensed with?)

A very general and also very important criterion for selecting the focus is curiosity. Characteristics of stimuli which arouse curiosity are:

- complexity
- novelty
- surprise
- incongruence

However, it is also possible that these characteristics lead to an avoidance behaviour, if they are to strong, because they also signify conflicts. As pointed out in section 2.3.2 creative people have a stronger sense of curiosity and a larger tolerance to conflicts than non-creative people, therefore the avoidance behaviour should not be that prominent with them.

In an information theoretical approach, curiosity can be interpreted as search for information (cf. [SK74, p. 119, 120]), because:

- complex patterns do contain more information than simple patterns *and*
- conflicts can be interpreted as uncertainty, which implies the possibility of finding additional information.

Especially in the scientific area, curiosity, as it arises when the outcome of an experiment differs from the expectations, can be an important mechanism of control as we saw above (cf. [KS88, KDF90]).

Chapter 4

Discussion of the Psychological Analysis

In this chapter we will summarize the preceding two chapters and discuss some consequences the presented facts have on the construction of artificial creative systems. In particular, we will answer the following questions from a psychological point of view:

1. What are the characteristics of a creative achievement?
2. Which abilities participate in the generation of a creative achievement?
3. How do these abilities interact?

4.1 Characteristics of a Creative Achievement

In section 2.2 we identified the following two criteria for a creative product:

- The product shall be novel in the sense that it is either a non-standard instantiation of a pattern known to the problem solver or it involves a hitherto unknown pattern.
- The product solves the problem, i.e., it complies with every problem criteria which is regarded as being essential with respect to the domain knowledge of the problem solver (including the knowledge that has been elaborated during the problem solving process).

The amount of creativity ascribed to an achievement usually varies depending on the amount of restructuring involved (e.g., was a new pattern constructed or was an existing pattern only instantiated in an unusual way). The amount of restructuring involved in a problem solution can only be judged from the viewpoint of the problem solver. Consequently, a creative achievement need not be novel with respect to history. However, if a solution is to be accepted by a human as being creative, the corresponding task must have a minimal complexity.

It is very difficult to judge the novelty of the result of the problem solving attempt from an external viewpoint, as the pattern of the solution may be familiar to the problem solver but not to the observer and vice versa. Similar problems are encountered when evaluating the criterion 'problem solving', because transformations of the problem specification are possible. The example of the Sydney Opera House (cf. page 7) illustrates this point very well. Consequently, it is very important that such restructurings arise in a controlled and reproducible way, as this is necessary for substantiating the claim that the product is problem solving.[1]

[1] As Seiffge-Krenke points out the realistic and controlled selection of ideas is a major difference between creativity and schizophrenic production (cf. [SK74, p. 31]).

Due to these problems a detailed assessment of the creativity involved in an achievement needs to take into account the creative process.

4.2 Abilities Needed for Creativity

Two basic capabilities which can contribute to a creative achievement can be deduced from the criteria for creative products mentioned above:

1. *Restructuring.* The problem solver should be able to use (or at least to recognize) standard solution patterns, but should also be able to overcome them and create new patterns. One possibility to comply with this requirement would be a Case-Based reasoner (the cases can be regarded as patterns) coupled with a searching facility, which would be used if the adaptation of cases would be insufficient for generating an acceptable answer.[2] However, this would only be a crude approximation.

2. *Recognition of emergent value.* The problem solver should be able to make useful modifications to the problem specification as a reaction to opportunities (emergent value) and problems found during the problem solving process.

The ability to overcome standard problem solving patterns is usually termed *restructuring* in psychological research. However, there are several different possibilities to bring forth a restructuring: defaults can be redrawn, the same knowledge can be stated in different ways ("the glass is half-full" instead of "the glass is half-empty"), new concepts and relations can be introduced, etc. Sometimes the term *restructuring* is not only used for modifications of the representation of knowledge, but also for modifications of the contents of the knowledge base. This includes adaptations of existing solutions, changes in the background knowledge (e.g., by inferences), and changes in the problem specification. The importance of this form of restructuring is emphasized by requirement 2. But this requirement is especially difficult to comply with for an artificial problem solver, as a large amount of background knowledge is required for judging newly introduced criteria.

In chapter 3 we identified several abilities as being important for creative problem solving. Although, in a specific creative process only a subset of these abilities is necessary, it seems that each of these abilities (operators) may be necessary in a creative process. The abilities we identified are:

1. Problem recognition
2. Search for knowledge (questions and experiments)
3. Mental simulation
4. Association
5. Forgetting
6. Knowledge transfer
7. Rule induction
8. Adaptation
9. Concept formation
10. Inference / Reformulation

[2] In principle, a program only based on search could also be able to find creative solutions (if creative solutions are contained in the search space), but it would be impossible for the program to recognize that a creative solution has been achieved; it would have to rely on the user to do this.

11. Reindexing

12. Evaluation and Verification

Probably, this list is not complete as neither our analysis nor the understanding of creativity in psychology are complete. It should also be noted that each of these operators should be regarded as representing a whole group of information processing activities. For example, any human possesses several different possibilities for concept formation. Such a large set of different approaches to solving a problem is surely an important prerequisite for creativity problem solving and should, therefore, also be available to an artificial creative problem solver.

Further, only little is known about the exact functioning of these operators in humans. However, it was possible to outline some basic ideas. Especially, with respect to the transfer of knowledge and rule induction it was possible to make detailed suggestions. Additionally, some systematic deviations from a completely rational behaviour could be identified. Some of these deviations can be explained by assuming that people often use heuristics which allow them to come to a conclusion with minimal effort, albeit at the risk of mistakes (cf. page 33). Other mistakes can be explained by the inability to recall the necessary information (e.g., previously seen examples). We believe that this kind of thinking is very important for human problem solving behaviour, as it allows to guess a solution and to repair it later. Often this requires much less effort than to find immediately a correct solution, especially for complex problems. As creative problem solving is usually only necessary for rather complex problems, we believe that heuristic reasoning is a very important prerequisite for creative problem solving. Consequently, heuristic reasoning is rather likely to be also very important for artificial creative problem solvers.

Heuristic reasoning can be regarded as a meta-ability, as it can be combined with any of the abilities mentioned above. A further meta-ability, which is tightly related to heuristic reasoning, is learning, as all aforementioned abilities may be improved by learning and even completely new methods may be learned.

Many of the abilities mentioned above are not typical of creativity, but are generally used in any form of problem solving. However, some are regarded as being typical of creativity like those termed restructuring above. This view was also expressed by Russel who termed creative problem solving "problem solving plus". The fact that many of the capabilities that are important in creativity are also essential to common problem solving activities can be regarded as a substantiation for the boundary value hypothesis (cf. page 8), which proposes that a certain minimum of intelligence is needed in order to be creative.

However, possession of these abilities is not sufficient, the problem solver must also be ready to apply them. Especially important for creativity is the readiness for applying restructurings. This can be regarded as open-mindedness in evaluation (cf. section 2.3.2). Again, this necessity can be illustrated using the example of the Sydney Opera House:

> The structural engineers retained for the project were asked to design the shells for the roof construction. These shells should rise fairly sharply from the base and that rise should decrease dramatically to form an almost horizontal roofing structure (cf. [McL92, p. 68]). The engineers were not able to devise such a construction, which would also comply with the other criteria given. Consequently, the architect of the opera designed shells of his own. However, contrary to the initial problem description the spines of these shells were formed by smooth curves.

It is rather unlikely that none of the engineers had this idea. Instead, it seems more probable that they did not feel justified to make such a departure from the original specification.[3] Perhaps in experiments, in which the independence of creativity and intelligence could not be demonstrated, such effects were important.

[3] This seems rather plausible, if one looks at the different education architects and engineers receive. Especially in the training of engineers the compliance with and not the variation of the specification is emphasized.

Open-mindedness and other qualities of creative people (e.g., preference of complex stimuli) tell us something about the control of the creative process, while other qualities (cf. section 2.3.2) like field-independence can be regarded as meta-abilities, as they tell us something about the way the operators are implemented in creative people.

4.3 Interaction of the Operators in the Creative Process

The operators mentioned in the preceding chapter are coupled by two components: they exchange their results through the knowledge base and the order of their application is determined by a control component.[4]

The organization of the knowledge base in the proposed model with its distinction between active and inactive knowledge is similar to the PUPS-model. This distinction seems ingenious for any problem solver which possesses a large amount of knowledge, of which only a small portion is relevant for any particular problem. It helps to cut down the search space far enough, that efficient problem solving is still possible. Moreover, it makes the operators 'association' and 'forgetting' necessary.

Additionally, experts differ in the organization of their knowledge base from beginners. In order to allow for efficient problem solving behaviour an expert-like organization of the knowledge base seems helpful. However, if a problem was solved creatively, either this kind of problem or at least the solution approach was unknown up to that time. Therefore, the problem solver will usually need to become an expert on this kind of problem first (although he may already be an expert in this domain). Probably the preparation stage is important for the reorganization of the knowledge base in this manner. Therefore, one can conjecture as an additional ability of a creative problem solver:

> The ability of improving the *organization of the knowledge base* based on problem solving experiences.

With respect to the control of the creative process we could only extract some very general rules from psychological literature. This may be caused by the fact that the control of the process is opportunistic (cf. [KW93]) and that control itself is a process of learning, i.e., new strategies can be generated dynamically.

Some of the basic forces that determine human problem solving behaviour can be understood based on the information theoretical interpretation of intelligence:

- The next tasks are often chosen in accordance with curiosity, i.e., a task is chosen, which promises the largest gain of information.[5] This is true for the selection of experiments (i.e., experiments which can discriminate between different theories are preferred) as well as for the decision which stimulus shall be further examined.

- The acceptance and the dismissal of hypotheses is not based on an all-or-nothing scheme. Instead a hypothesis is only accepted or dismissed if enough evidence for or against it has been collected.

However, this information theoretical behaviour is only approximated by human problem solvers. This may have two reasons: on the one hand humans do not consciously make any information theoretical computations (at best they make estimations), on the other hand additional factors influence the behaviour (e.g., which experiment is simpler to conduct).

Besides these information theoretical factors human behaviour is influenced by:

- a certain persistence (focus), which makes people stick to an old problem, even if a new one would be more informative,

[4] Obviously, the structure of any problem solver can be described in this way.

[5] However, if the complexity and, consequently, the information content of a stimulus is too high, people usually show an avoidance behaviour.

- a limited ability to cope with complex (and therefore information-rich) stimuli, i.e., above a certain level of complexity the subjects exhibit an avoidance behaviour. The maximum complexity tolerated depends on the subject. Creative people are believed to have a higher tolerance to complex stimuli (cf. section 2.3.2).

The strength of the focus (i.e., the difference in information content tolerated) and the maximum complexity of the stimuli tolerated are two variables which, at least partially, explain the difference in problem solving behaviour of different persons.

4.4 Summary

One can summarize in saying, that it was not possible to explain in detail how creative solutions are produced, but this could not be expected, as the psychological analysis of the creative process is far from being completed. However, some criteria could be deduced, which can help in judging a problem solving process as either creative or non-creative. Additionally, it was possible to identify some abilities which a problem solver (artificial or not) may need in order to be creative. In the following chapters, we will use the criteria found for evaluating some studies from AI with respect to the ideas they provide for the realization of a creative problem solver.

Chapter 5

AI Approaches to Creative Problem Solving

5.1 Introduction

In the previous chapters we saw that the abilities ordinary problem solvers possess are also needed by creative problem solvers. However, these abilities are sometimes regarded being implemented differently in creative problem solvers. Further, additional abilities are often emphasized in psychological literature as being necessary for a creative problem solver, e.g., problem recognition, restructuring of knowledge, ... Moreover, a creative problem solver must possess such meta-abilities like learning or field-independence. Due to the large range of these abilities a complete coverage of the relevant literature would necessarily include the areas of problem solving (planning, theorem proving, etc.) and learning. Obviously, this exceeds the possibilities of this study by far. Consequently, the literature which was analyzed had to be carefully selected. The main criteria which were used in this selection were:

- the described systems should integrate several of the operators mentioned in the preceding chapter *and*
- they should be regarded as showing (as far as possible) creative behaviour.

With the help of these criteria two areas of research could be identified, as being especially important for this study:

Automated Discovery: Usually these systems integrate (simple) problem solving capabilities with the ability to search for new knowledge and integrate it into the knowledge base. Additionally, they possess a mechanism of control which enables them to generate new problems and solve them. Some of these systems also have the ability to modify the representation of their knowledge.

Usually, *scientific discovery* is regarded as a subfield of *automated discovery*. With respect to the goals of this study systems for *scientific discovery* have the additional advantage that they model the process of scientific discovery as it is performed by humans. This activity is usually regarded as creative.

Creative Design: Systems of this kind are based on a standard problem solving method (e.g., Case-Based Reasoning) and possess the ability to modify the problem specification. This is regarded as being a necessary capability for situations in which the initial problem description is ill-defined. Fischer regards architecture as a prototypical example for this situation (cf. [Fis92]). This area of research is relatively new (the first workshops were held in the late

1980s) and still in some kind of adolescence. Today, most contributions to this field are still of a more theoretical nature than in the form of working systems. We hope that this study may contribute to this line of research.

Additionally, we used some studies which are occupied with more specialized aspects of creative problem solving. This includes forgetting ([MS88]), the restructuring of the knowledge base by experience ([ES90, Kol93]), and concept formation ([Wro94, Fis87, Leb87]).

The following two sections will give a short overview of approaches to *automated discovery* and *creative design*.

5.2 Automated Discovery

The term *automated discovery* subsumes a large number of different approaches. Basically, systems for *automated discovery* try to recognize regularities in their environment and validate them empirically. The discoveries made are integrated into the knowledge base of the system in such a way that they can be used in future discovery cycles.[1] Usually, these systems possess *functional autonomy*, that is they are able to generate new tasks on their own and to solve them. The discovery of regularities and their empirical validation is often described as search in two spaces: rule and instance space (or hypothesis and experiment space) (cf. [KD88]).

Lenat points out three properties of domains which can be explored by heuristic search[2]:

1. *observability*, i.e., the system is able to gather data about the domain under study.
2. *continuity*, i.e., regularities are valid for a large range of values.
3. *stability*, i.e., the regularities hold for a considerable period of time.

Out of these considerations he identifies *observability* as the most problematic issue.

Consequently, the ability to construct experiments is of decisive importance for the success of a discovery program. This ability enables the program to gain additional information and construct new hypothesis. There are three possible sources for the results of experiments:

1. The results are entered by the user (e.g., KEKADA, [KS88]).
2. The experiments are conducted by an external simulator (e.g., [SHM91]; LIVE, [She93]; DIDO, [SM93]).
3. The experiments are conducted by the discovery program itself (e.g., AM, EURISKO [Len83b]). In extreme cases the program itself can be the object of experimentation (EURISKO).

From approach 1 to 3 the knowledge acquisition problem gets smaller and smaller as experiments become increasingly simple to conduct. However, at the same time the frame of possible extensions of the theory becomes smaller. User inputs can be based on any theory (even one, which is unknown to the user himself), which in turn can be discovered by the program. However, a simulator based on Newtonian mechanics can lead the discovery system only to the discovery of Newtonian mechanics (or an equivalent formulation). In case 3 the program possesses already some part of the knowledge (perhaps implicitly) which must still be discovered in case 2.

The interaction between experimentation, hypothesis formation, and new experiments made for validating the hypothesis is sometimes called *discovery-cycle*. Haase compares it to the *model of assimilation and accommodation* proposed by Piaget for the description of the psychological development of children (cf. [Haa89, p. 154]).

[1] According to Haase (cf. [Haa89]) the ability of closed-loop learning is an important prerequisite to the success of any *automated discovery*-system.

[2] Most approaches to *automated discovery* employ heuristic search, i.e., their search for regularities is driven by heuristics.

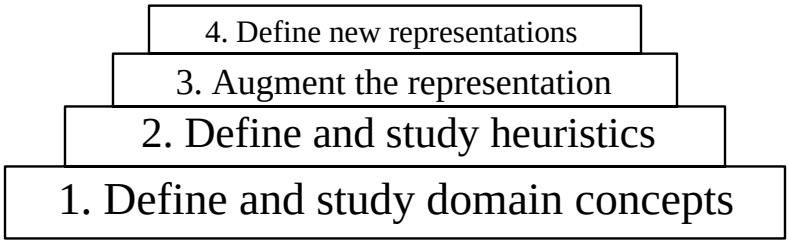

Figure 5.1: The levels of discovery

As Lenat points out (cf. [Len82b, p. 248]) four levels of discovery can be distinguished (cf. Figure 5.1). The basic idea of this model is that a decline in the efficiency of discovery on one level (defined by the a posteriori quality of the introduced discoveries) should lead to a continuation of the discovery process on a higher level. However, most discovery programs to date only mastered level 1.[3] The definition of concepts is equivalent to the clustering of data in categories for which a new term is introduced, while the study of domain concepts is equivalent to the search for rules, which hold among the different concepts.

Some systems also work on level 2 or 3, however it seems that EURISKO is still the only system which mastered level 1 through 3. Consequently, it is still regarded as the highest level of *automated discovery* ever achieved (cf. [SHM91]). There have also been attempts to reach level 4 (cf. [Haa89]), however their success is unclear.

In recent years the interest in *constructive induction* has intensified. This line of research tries to combine level 1 and 3 in order to get an optimal classification of examples. However, these systems can hardly be regarded as examples of automated discovery systems, because they do not put their results to further use and do not possess *functional autonomy*.

Functional autonomy is usually implemented using heuristical control (e.g., [Len82b], [KS88], etc.). That is heuristics are used to choose from a set of tasks the next task to be carried out based on experimental results and attempts of theory construction. The heuristics can either be predesigned or in the case of EURISKO self-constructed. The tasks can also be generated by heuristics.

Based on a survey of *automated discovery*-systems we give a general model of *automated discovery*-systems in figure 5.2. This model is a generalization of models given in [KS88] and [SHM91], but also takes into account the architecture of systems like AM, EURISKO, or LIVE. The range of variation of this types of systems is very large.

The discovery process can be partitioned into two phases: the generation phase and the evaluation phase. During the generation phase new experiences are generated, in the evaluation phase these experiences are examined for any improvements of the knowledge base or any new tasks they suggest. Of crucial importance for the speed with which the knowledge base can be improved is that the generation of experiences is *goal-oriented*. That is plausible generate and test is usually preferred to random generate and test, even if this may hinder some discoveries.

In the following comments we will try to illustrate some of the possibilities of variation, which exist with respect to the elements of the discovery model outlined in figure 5.2:

Agenda:

- multiple agendas may exist:
  - In [SHM91] a system is described which uses two intertwined discovery processes. One agenda is used for control problems and one is used for the development of new meta-rules (similar to heuristics).

[3] One can regard the introduction of new concepts as an augmentation of the representation, too.

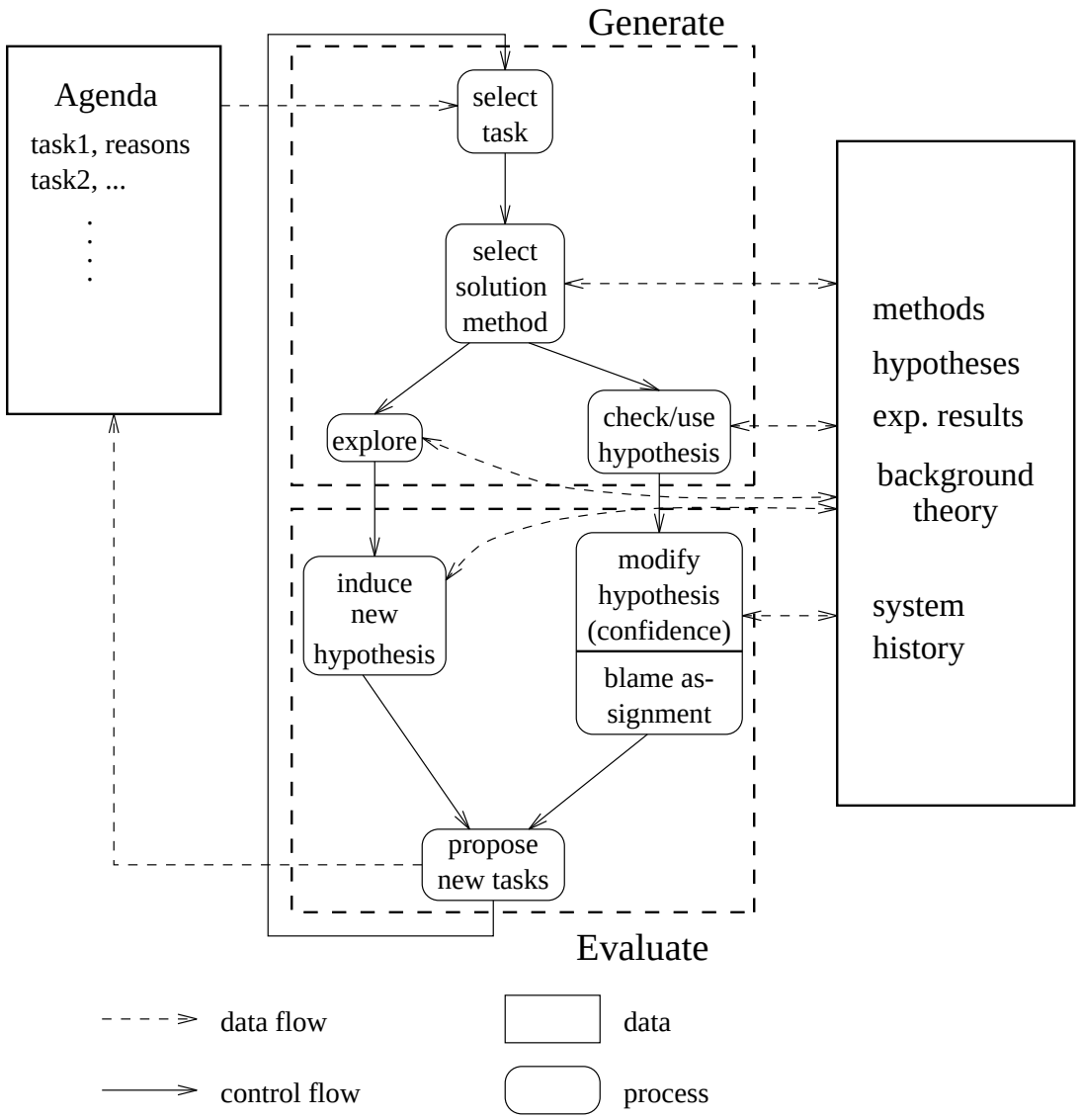

Figure 5.2: A general model of automated discovery

  - In EURISKO there is an agenda for each topic. The system is able to split and merge topics.

- no explicit agenda exists:
  In the LIVE-system ([She93]) a fixed goal is given at any time. If planning for achieving this goal fails, experiments (with hypothesis) and explorations (without hypothesis) are made for improving the theory far enough that planning becomes possible. These subtasks are executed as soon as they are proposed.

Overall control: Especially with respect to control this model is a rough approximation. At one extreme EURISKO uses a three level loop (topic, task, heuristic) for choosing appropriate actions, while in most systems control is implemented by a simple rule interpreter. The extent

of heuristical control also varies considerably among systems. While in EURISKO virtually everything is controlled by heuristics[4] the LIVE-system possesses a fixed control structure.

Focus: By giving the system a focus of attention it is possible to make it work on related problems over several cycles (e.g., generate hypothesis, carry out experiment for this hypothesis, modify values of it, etc.). Often this focus is not made explicit. AM is behaving in this way, because a heuristic which carries out a task can propose new tasks and give them a high initial worth.

Select solution method: The selected solution method may not only determine which kind of manipulation has to be made, but also which values are to be manipulated (e.g., if the temperature is too high, then open the *window*).

Hypothesis: If the system makes discoveries also at levels larger than 1, then a hypothesis can also be a heuristic, a new attribute, etc.

Modification of hypothesis confidence: Some systems change their hypothesis immediately upon a negative result (e.g., LIVE, EURISKO), while others change the confidence in a hypothesis first (e.g., KEKADA). The second kind of systems change a hypothesis only if enough negative evidence has been collected. This behaviour models more closely the behaviour of humans.

Blame assignment: The problem of deciding which hypothesis is responsible for a good or a bad result is not trivial. Especially, if more than one hypothesis contributed to the result of the current cycle or if results of a previous cycle may have influenced the current one. Some systems use the *bucket-brigade algorithm* or a modification of it (cf. [SHM91]) for blame assignment.

System history: The history of the systems actions and their results is important for the induction of new hypothesis and the modification of old ones.

5.3 Creative Design

Creative design is a rather new field within AI. Contrary to other systems for design support (e.g., CAD-systems), which aim at supporting routine design it is the aim of this field to support creative design activities or even automate them completely. According to Rosenman and Gero (cf. [RG92, p. 113]) three types of design activities can be distinguished:

Routine design: All design variables and their possible instantiations are well known.

Innovative design: The space of design variables is not changed, but their possible instantiations are. (The basic pattern of the design is already known.)

Creative design: The space of design variables is changed. (There is no obvious connection to prior designs.)

The distinction between the different types of design activities is not clearcut. Especially, the distinction between innovative and creative design is difficult. Some authors regard the ability to modify the problem specification as a sufficient criterion for *creative design* ([KW93]), while others regard it only as sufficient for innovative design ([Nav91]).

Navinchandra distinguishes three types of criteria for the evaluation of a design (cf. [Nav91]):

prior criteria: Criteria which are given as part of the design task.

emergent criteria: Criteria which become obvious during the design process.

[4] EURISKO even possesses heuristics for constructing new rule interpreters for the execution of a solution method ([Len83a, p. 68]).

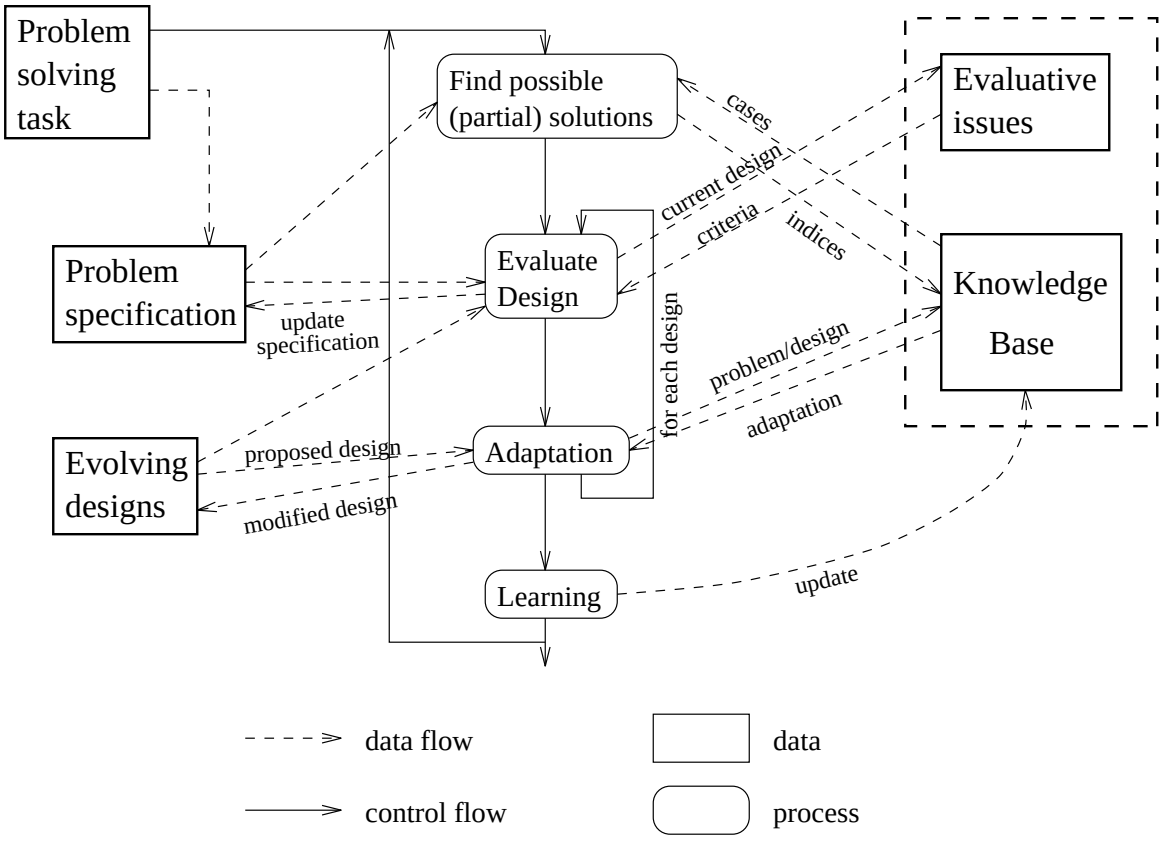

Figure 5.3: A general model of creative design

post-design criteria: Criteria which are recognized during the usage of the design.[5]

Modifications of the specification are triggered by the recognition of emergent criteria. These emergent criteria are either due to recognized possibilities or to recognized problems. The recognized possibilities are sometimes called *emergent value* ([McL92]).

The prototypical architecture for *creative design*-systems sketched in figure 5.3 is mostly based on work by Kolodner et al. ([KW93],[KP90]), but also takes into account [RG92], [Nav91], [BGP94], ... The following remarks will highlight some possible variations of this prototypical architecture:

Find possible solutions: Two basically different approaches can be distinguished:

1. Search in the search space defined by the problem specification (e.g., [Nav91]).
2. Recall of prior cases using the problem specification as an index (e.g., [KW93]).

Evaluate designs: An important characteristic of systems for *creative design* is that the problem specification can be modified. These modifications are made during evaluation. The current design can be used as an index to the evaluative issues. In the CYCLOPS-system ([Nav91]) it is also possible to relax constraints in order to search for additional designs (not acceptable according to the original specification).

[5] One can regard simulation (especially mental simulation) as an attempt to recognize these criteria during the design stage, i.e., to transform them into emergent criteria.

Kolodner and Wills subdivide the evaluation into critique and data collection, where data collection basically is experimentation (mental simulation) with the design.

Evaluative issues: Kolodner proposes ([Kol94]) a situation assessment step, in which additional general criteria are recalled based on the current problem (e.g., when designing a meal, it should be edible). In order to perform such a step it is necessary for the problem solver to possess general criteria. Naturally, most of these criteria will be domain-specific. Whether and how it is possible to learn such criteria is still an open question for research. Such a learning process could be based on previous problem solutions and their outcomes.

The more general questions of how new criteria can emerge and how they should be weighted are also still open. In the systems built to date these criteria as well as their weighting are precoded. These criteria can be coded in the form of rules, which check the design for opportunities ([Nav91]), or in the form of built-in cases. In CREATIVE JULIA ([KP90]) the criteria are partitioned into primary and secondary goals. The system is more ready to modify secondary goals.

Chapter 6

An Analysis of AI Approaches to the Creative Process

In the last chapter we gave an overview of two AI approaches to the automatization of the creative process. Here we will examine in more detail the different approaches used in AI for producing results similar to those of the operators postulated in chapter 3. Largely, the structure of this analysis is similar to this of chapter 3. However, we will use a more principled approach here.

6.1 Structure of the Analysis

In order to have a lucid organization of the various contributions AI made to the exploration of the creative process we refined the categorization which we gave in chapter 3.

Basically, the following three categories can be distinguished:

1. *Operators* — Which abilities does a creative problem solver need?
2. *Control* — How can the interplay between these abilities be organized?
3. *Knowledge* — How must the knowledge base be organized in order to allow for successful interaction of the various abilities?

The different operators can be distinguished into the following categories:

a. recognition of problems

b. search for information

c. activation of information

d. construction of new knowledge

e. restructuring of knowledge

The distinction between constructing new knowledge and restructuring knowledge was introduced due to pragmatic reasons. Thereby, we can distinguish between changing existing knowledge structures around and generating new knowledge. This distinction parallels the distinction between unusual instantiations of existing patterns and the generation of new patterns. However, in psychology usually both types of changes to knowledge structures are called restructuring.

The restructuring of knowledge can be further subdivided into:

I. changes to the accessibility of knowledge

II. restructuring on the syntactic level

III. restructuring on the semantic level

This distinction is helpful, because in each category the available knowledge is regarded from a different point of view.

Evaluation is tightly connected with the control of the creative process. Therefore, we will not discuss evaluation separately from control as we did in chapter 3. As a model for evaluation we will use the "information filtering model". Readers who are unfamiliar with this model should refer to appendix B for a brief introduction. As other operators (e.g., forgetting and association) are also related to the evaluation (and subsequent filtering) of information, we will refer to this model throughout this chapter.

Tables 6.1 through 6.3 give an overview of the structure of the creative process as we will use it in this chapter and a list of the questions we will try to answer in this context.

Table 6.4 gives an overview of the various systems we examined for answering these questions, along with the components of our model they contributed to. Although we regard 'evaluation' and 'learning to control' as parts of the control component, we mention them in table 6.4 explicitly, as they may be of interest independently of the other parts of the control component. Table 6.4 does not summarize all contributions the systems can make to the various components, but only the contributions we will discuss in this chapter. There may exist further relations to the components of our model we do not know of, as they were not clear from the literature we used.

In the creativity model based on psychological literature (cf. figure 3.1) we assumed that the knowledge base can be partitioned into active and inactive knowledge. In the following analysis we will also use this view, despite the fact that virtually no AI system possesses such an distinction. However, we believe that any really creative problem solver must possess such a wealth of knowledge that this distinction will be necessary for the problem solver in order to be efficient.[1]

The analysis of the contributions of AI systems to the different operators will be based on the following scheme:

Operator: Name of the operator and, if applicable, the names of various techniques, which can be used for implementing it.

Scope: Sometimes it is not clear from chapter 3 what kind of work is performed by this ability and what is done by others. Additionally, from an AI perspective (i.e., a more technical point of view) a distinction between the operators different from that used in chapter 3 may be more adequate. Consequently, we will clarify under this heading what falls into the scope of this operator and what does not.

Precondition: What are typical conditions for the use of this ability?
If several preconditions are mentioned, then, typically, each of these conditions describes a distinct situation, in which the operator may be used. The exact form of the operator may vary likewise. One may also regard this situation as if several different operators would exist, each slightly different, having one of the preconditions of the list as its precondition.

Purpose of call: What are typical goals which shall be reached by the application of this operator?

Input: To what kind of knowledge is this operator applied?

Knowledge used: What background knowledge is needed?

Knowledge generated: What modifications to the knowledge base of the problem solver are made?

[1] In fact, the CYC-system, probably the largest knowledge base to date, possesses the notion of problem solving contexts (cf. [GL92]).

Problem recognition:

- What categories of problems exist?
- How can a problem be identified?
- How can problems be represented?

Search for knowledge:

- What conditions may lead to the generation of experiments?
- Which restrictions with respect to internal questions can arise?
- How can experiments be constructed?
- How can the information an experiment may yield be estimated?

Activation of knowledge:

recall/association:

- Which indices are used?
- How can additional keys for recall be found?
- How can multiple keys be combined in a retrieval?
- Which constraints can be used for selecting answers?

mental simulation:

- How can simulations be performed?
- What information becomes activated?
- What information can be discovered?

forgetting:

- How may a problem solver profit from forgetting?
- How is the knowledge selected, which shall be deleted?

Construction of knowledge:

inference: —

transfer:

- How can a mapping for a transfer be built?
- How can a selection be made among possible mappings?

induction:

- What hints to regularities exist?
- What types of regularities can be recognized?
- What techniques for recognizing regularities exist?
- How can causal relations be recognized?
- How can knowledge aid in induction?

Table 6.1: Questions used for the analysis of AI literature (Part I)

Restructuring:

Changes to the accessibility of knowledge:

reindexing:

- What is an useful index?
- Which events may trigger a reorganization of the memory?
- How can a reorganization be performed?

Restructuring on the syntactic level:

reformulation:

- What techniques for the reformulation of knowledge exist?
- How are reformulation and modification of the representation language connected?

modification of the representation:

- Which modifications of the representation are possible?
- How can these modifications be described within the representation (if possible)?
- Which events can trigger a modification of the representation?
- Which elements are grouped together into a new concept?
- What additional information is generated (induced) during modification of the representation?
- How is the modification of the representation influenced by a performance task?

Restructuring on the semantic level:

adaptation of background knowledge:

- What techniques for adapting knowledge exist?
- How can parts of the background knowledge that need adaptation can be identified?

adaptation of problem solutions:

- What techniques for performing adaptations exist?
- What part of the solution shall be adapted?
- How can an adequate strategy for adaptation be selected?

modification of problem specifications:

- Which events may cause a modification?
- What types of modifications exist?

Table 6.2: Questions used for the analysis of AI literature (Part II)

Control:

Control model:
- What functions are provided by the control component?
- What informations must be managed?
- What relations may hold among the various tasks?
- How are the various functions of the control component interrelated?
- How are control and operator execution interrelated?

Choosing actions:
- What criteria are used for selecting tasks for further work?
- What techniques exist for task decomposition?
- What techniques can be used for finding applicable operators?
- How is a decision made between the various applicable operators?

Operator Application:
- What measures must be taken for creating an adequate environment for generating new tasks?

Evaluation:
- What methods exist for assessing the success of an action?
- What techniques for credit/blame-assignment exist?
- How can the information generated during credit/blame-assignment be used for generating new tasks?
- What approaches to learning can be distinguished?
- Which types of control knowledge can be learned?
- How does learning interact with problem recognition and blame-assignment?

Memory Organization:

Knowledge Types:
- What information is necessary/helpful to creative problem solving?
- In which categories can the information be partitioned?
- What roles can the knowledge items play during problem solving?

Knowledge base structure:
- Why should the knowledge base be structured?
- What methods for structuring exist?

Representation language:
- What kind of knowledge must be representable?

Table 6.3: Questions used for the analysis of AI literature (Part III)

Program	problem recognition	search for knowledge	association/recall	mental simulation	forgetting	transfer	induction	reindexing	reformulation	modify repr. lang.	adapt background know.	adapt problem solution	modify problem spec.	execute control	evaluation	learn to control	memory
ACME						×											
AM	×					×	×			×		×		×	×		
ARCS						×											
Arachne										×							
Aries																×	
BR-3	×									×							
Bacon							×			×							
Blip							×			×	×						
Chef				×													
Cluster										×							
Cobweb										×							
Cyc			×								×						×
Cyclops			×										×				
Dido	×	×			×		×			×							
Eureka			×		×			×									
Eurisko	×			×						×		×	×	×	×	×	
Exor										×							
Failsafe-2															×	×	
Funes					×												
Grapes		×				×											
Hypo											×						
ID3							×				×						
IDS														×			
IDeAL													×				
Inc										×							
JCM							×				×						
Judge												×					
Julia				×								×					
Kekada	×	×															
Kepler		×															
Live	×	×					×				×						
MacBeth			×				×				×						
MacLearn					×												
Meta-Aqua								×							×		
Mobal	×				×		×		×	×	×				×		
Occam							×			×				×			
Persuader				×													
Prodigy	×				×												
Rapter															×	×	
Runner								×									
Swale	×		×									×					
Unimem										×							

Table 6.4: AI systems examined in this study and their contributions to the components of the creativity model

We will not distinguish between the main output and side effects of the call, as this distinction may vary from call to call.[2]

Tasks generated: Besides the output of additional knowledge an operator may propose the generation of new tasks. For example, after the recognition of a regularity, its validation becomes an important task. These are actions which need not to be carried out immediately, however, they are made plausible by the current task.

Contributions from AI: Under this heading we will give an overview of some AI approaches to implementing this (or a similar) operator. Usually, in AI literature complete systems are described. For each operator we will only describe the relevant material.

Comparison with Psychology: Here the *contributions from AI* are compared with ideas from psychology. This comparison is mostly based on the material discussed in chapter 3.

The above scheme should only be regarded as a maximal program, as some rubric may be inadequate for an operator or may be impossible to fill based on the available literature.

Neither the memory organization necessary for creativity, nor the control of this process are adequately covered in psychological literature. Further there is virtually no evidence in AI literature for systems adequately integrating the diversity of operators we propose here. Consequently, we will not use this scheme for discussing memory and control (the elements coupling the previous operators).

6.2 Problem Recognition

Operator: problem recognition

Scope: In this section we will interpret *problem recognition* more generally than we did in section 3.3. There we used the term *problem recognition* only to describe the recognition of new problems, which are independent from previously known problems. The recognition of additional problems during the problem solving process was regarded as part of the evaluation process. This was in accordance with the approach taken in psychological literature. However, from the viewpoint of the techniques used to recognize a problem both situations are mostly identical. For example, whether a deficiency of a household utensil (cf. page 10) is recognized during its construction or afterwards is not important with respect to the techniques which can be employed.[3] Consequently, we will handle both cases here.

Precondition:

1. new, interesting, or unexpected information was found
2. there is a lack of interesting tasks

Purpose of call:

1. find deficiencies
2. generate new tasks

Input: 1. the information, which triggered the call of the operator, is used as an initial focus

2. —

Knowledge used:

[2]The fact that side effect and main output of a call are interchangeable may be one reason for the rather abrupt changes, which are sometimes observed in creative reasoning. While an operator is called for one purpose its side effects may make plausible a different idea and trigger a different line of reasoning.

[3]However, in the first case more knowledge is already activated, therefore the chances are better for recognizing a deficiency.

- criteria for problems
- a focus, i.e., a (partial) problem solution, if one is supplied

Knowledge generated: problem specification

Tasks generated: task to solve the problem

Contributions from AI:
Most studies in AI, which are concerned with the recognition of problems only examine problems which arise during problem solving. Systems, which try to find new problems usually belong to the field of *automated discovery*.

We want to distinguish the following three aspects of problem solving:

Problem symptom: The symptom, the problem is recognized by.

Problem cause: The cause of the symptom. Usually, the relation between cause and symptom is not unequivocal.

Problem solving task: The task of solving the problem. The task can either be described at the level of problem symptoms or at the level of problem causes.

In this section we will only deal with the recognition of problem symptoms. Finding causes for problem symptoms (blame assignment) and generating problem solving tasks will be discussed in section 6.7 ("control"). We will also discuss these problems from the point of view of adaptation in section 6.6.4.2.

There are three levels on which problems can arise:

1. in the environment
2. in the reasoning process
3. in the knowledge base

In the first case reasoning is successful, but events observed in the environment contradict the results of this reasoning. In case 2 problems arise in the reasoning process, which prevent its successful completion.[4] In case 3 problems arise during the recall of knowledge. Reasons for this may be, that no adequate knowledge is available, it cannot be recalled, or the uncertainty of this knowledge is too high.

6.2.1 Problems in the Environment

Problems of this kind arise if an unexpected event occurs, i.e., an event which can not be explained by the current theory, given the current information about the environment. In this case it is necessary to find additional information or even to change the current model of the environment.

The SWALE-system (cf. [SL89]) is a Case Based Explainer: short stories are presented to the system, the system then tries to understand the sequence of events in the story (i.e., to explain it by its theory). If this is not possible, then the system tries to retrieve an adequate explanation and to adapt it to the current case (cf. [SL89, p. 360]). This adaptation of explanations and the corresponding improvement of its theory is the central task of the system. In order to make the system capable of recognizing anomalies, 170 anomalies have been collected and partitioned into 14 categories. Then the system was equipped with criteria for recognizing these categories (cf. [SL89, p. 362]). Some of these categories are: role filler of wrong category (horse jogging), premature event, premature termination of event sequence, delayed event, planning problems (a plan which must obviously go wrong), novel causal connection.

[4] Most contributions to problem recognition in AI are restricted to this kind of problems.

The task of the BR-3-system (cf. [Kok91]) is to expand a basic theory of particle physics by the introduction of additional quantum numbers, such that the system can explain the reactions (or the absence of reactions) it is told of. In this system problems are recognized by checking for each observable reaction whether the conservation of quantum numbers is guaranteed and for each unobservable reaction that the conservation is not guaranteed. If this is not the case additional quantum numbers are introduced.

The KEKADA-system, which tries to model the discovery of the urea cycle by Hans Krebs (cf. page 36) employs a single heuristic for recognizing problems (cf. [KS88, p. 153]): If the result of an experiment is not in accordance with the expectations, then study this phenomenon in greater detail.

Another system which reacts to unexpected events (so-called prediction failures) is the LIVE-system (cf. [She93]). This system is based on the notion of prediction sequences. A prediction sequence takes the form $< S_0, a_1, P_1, a_2, P_2, \ldots, a_n, P_n >$, where S_0 is the observation of the current state, a_i is an action, and P_i is a prediction ([She93, p. 148]). Prediction sequences can either be used in forward reasoning for predicting the results of actions, or in backward reasoning for planning. They are learned from the environment by observation.[5] If in an action sequence $S_0, a_1, S_1, a_2, \ldots$ a situation S_i is not compatible with the prediction P_i, then a prediction failure occurs. In this case the system tries to revise its prediction sequences.

A similar approach is taken by the DIDO-system (cf. [SM93]). In this system the different possible situations are represented by a specialization hierarchy. For each class of situations the observed results of the various possible actions and their probabilities are stored. Here each observation can lead to a change in the theory, because even if the observation has been made before, its probability is increased, if it is not already equal to 1.

6.2.2 Problems in the Reasoning Process

In this case the goal of the reasoning process cannot be reached. Either the process needs to be aborted or it is terminated regularly, but its result is no solution to the problem.

This kind of problems can arise in the LIVE-system, because the system generates plans in order to produce certain states of the environment. This system can recognize two types of problems:

- regression deadlock
- regression loop

In the case of a regression deadlock multiple subgoals are in conflict independently of their ordering. In a regression loop the same subgoal is generated over and over again.

Veloso and Carbonell (cf. [VC91]) emphasize the potential benefits of problem recognition in the reasoning process, too. They studied a planner which was combined with a learning component (PRODIGY with Derivational Analogy). They could show that sometimes it is more efficient to solve a smaller problem first, which is only important for learning, and then solve the larger problem instead of solving the large problem immediately. Consequently, it would be ingenious if the problem solver would be able to generate a smaller problem if the consumption of resources is to large or if the problem is regarded as too complicated from the outset. However, the authors did not examine ways to do this.

The type of reasoning problems, which is examined most often in AI literature is that the result of the reasoning does not directly solve the problem. This happens in systems, which reuse a problem solution in a new situation (e.g., analogy, CBR, ...). In this case an additional adaptation step must be made (cf. section 6.6.4.2).

[5] In order to do this, it is necessary to analyze which observations are important and which are not. This can be regarded as an attention filter. We will return to this point in section 6.5.3.3.

6.2.3 Problems in the Knowledge Base

Necessary knowledge can not be retrieved from the knowledge base, or is regarded as being too uncertain.

This kind of problems is recognized by the systems LIVE and DIDO (see above). LIVE recognizes that it does not possess an adequate prediction sequence for planning. Additionally, the LIVE-system uses the following three heuristics for recognizing and repairing deficiencies in the knowledge base:

1. *Goal seeking:* If an action B is known to change the feature F of objects, then explore B to change the value of F of the learner to be equal (or some other relation) to the value of F of some goal objects.

2. *Anomalous behaviour resolution:* Explore actions that apparently have no effect in the environment.

3. *Curiosity:* Once in a while, randomly explore some not-yet-explored actions with random parameters.

In the DIDO-system, for each possibly result of an action a probability is estimated based on the experience of the system. A problem is always given by a result of an action that has a high uncertainty. For measuring the uncertainty of a result the Shannon-uncertainty function is used:

$$H = -\sum_i p_i \ln_2 p_i \tag{6.1}$$

In formula 6.1 the p_i denote the a-priori-probabilities that the result belongs to class i. With the help of this approach DIDO is capable of learning in environments in which noise exists. However, the system is not able to explicitly recognize the noise.

The MOBAL-system (cf. [Wro94]) can also recognize problems. This system is a tool for theory revision. New facts can contradict other facts or conclusions from the knowledge base. This is recognized as a problem and triggers a theory revision process.

The systems developed by Lenat (AM, [Len82a] and EURISKO, [Len82b, Len83c, Len83a]) can hardly be evaluated according to the categories above. These systems belong to the category of *knowledge-rich discovery systems*, i.e., they possess a large concept hierarchy and a large number of heuristics. AM (and similarly EURISKO) possess so-called *suggest-heuristics*. These heuristics propose additional tasks.[6] A general classification of these heuristics is impossible, as they propose tasks of various kinds (cf. [Len82a, p. 180]):

93 After creating a new generalization G of concept C,
Explicitly look for ties between G and other close generalizations of C

94 After creating a new specialization S of concept C,
Consider looking for examples of S

Some of the tasks generated by these heuristics are usually not regarded as tasks in their own right, but as subtasks in the search for interesting concepts and conjectures.

6.2.4 Representation of Problems

With respect to the representation of problems three approaches can be distinguished:

Task: This is illustrated by AM and EURISKO. If a problem is recognized a problem solving task is generated immediately. This problem solving task is explicitly represented in the system.

[6] According to [Len82b] AM possesses 63 suggest-heuristics, which are attached to concepts on different levels of the hierarchy. Additionally other heuristics can propose tasks.

Implicit: The recognition of problems immediately leads to action for solving the problem (e.g., in the LIVE-system the failure of a prediction leads to a modification of the theory).

Explicit: An explicit representation of the problem is generated, which can be used in future actions. For example, in the SWALE-system an anomaly is explicitly represented and used for retrieving possible explanations.

Most systems use an implicit representation of problems, only some discovery systems use a task representation. In contrast, the explicit representation of problems is rare. Additionally, we do not know of any system which is able to manipulate these problem specifications. Even in *creative design* systems only the problem specification given by the user can be manipulated.

Comparison of AI Contributions to "Problem Recognition" with Psychology: In section 3.3 we cited McLaughlin's opinion, that humans tend to form a consistent world view. In chapter 4 we made this more precise:

- Problems solvers (either human or artificial) try to keep their theory consistent with their observations of the world.
- They try to keep the different parts of their theory consistent with each other, especially they try to reduce the uncertainty of their predictions.

Based on the examples given above, one can add that the successful and efficient execution of a task is a basic goal (this certainly holds true for a human as well). Consequently, three principle sources for problems exist[7]:

1. Problems may arise during the execution of a task.
2. The plausibility of existing knowledge may be doubtful. (Compare heuristic 2 used in the LIVE-system.)
3. Generally, a desire for new knowledge exists. (Compare heuristic 3 used in the LIVE-system and heuristic 93 in AM.)

For all three mechanisms domain-specific as well as domain-independent criteria may be used. However, most of the criteria we mentioned above are domain-independent (e.g., a domain-independent heuristic for problem source 3 is the uncertainty measure given by formula 6.1). This is mostly due to the fact that in this general discussion we concentrated on this kind of criteria. For example, AM possesses a variety of domain-specific suggest-heuristics, which we did not examine in much detail. An example of a rather domain-specific heuristic is heuristic 218 from AM (cf. [Len82a, p. 202]):[8]

> Given an implication,
> Try to weaken the left side as much as possible without destroying the validity of the whole implication. Similarly, try to strengthen the right side of the implication.

Finding problems of household utensils as it is studied by Guilford can be regarded as mental simulation combined with problem recognition in the reasoning process. However, Guilford expects that principally new criteria can be introduced. This is more adequately regarded as restructuring of problem specifications than as problem recognition. Consequently, we will return to this point in section 6.6.4.3.

We can summarize by concluding that with respect to the problem types discussed in section 3.3 corresponding studies exist in AI. Problems, which arise during the reasoning process drew the most

[7] We tried to order them based on decreasing importance.

[8] Note, that the AM's domain was mathematics. Consequently, this heuristic must be regarded as being specific to domain of logic and there to the operation `implication`. However, as logics is a rather generally applicable science, this heuristic would surely prove useful in a multitude of domains.

attention. We could identify some general rules for problem recognition, however domain-specific criteria are necessary, too. This points to the need for a large body of domain knowledge. Whether and how such criteria can be constructed is a poorly explored area in AI to date. We will return to this point in section 6.7.6.

Out of different possibilities for problem representation the explicit representation of problems seems to be the most adequate for creative problem solving, because this is the only one which allows for a restructuring of the problem specification.

6.3 Search for Knowledge

Operator: Questions / Experiments

Scope: In this section we will deal with the transformation of internal questions into external questions or experiments. Neither the creation, nor the consequences of answers to these questions will be treated in this section. Most of the internal questions are triggered by the results of other operators[9], while evaluation and control decide about the consequences of an answer.

Precondition:

1. The recall of knowledge fails. Either not enough knowledge is retrieved or the certainty of this knowledge is not high enough.
2. The contents of the knowledge base is judged unsatisfactorily and no other important task exists.

Purpose of call: Retrieve knowledge from the environment.

Input: 1. the attribute to be determined, the context in which it shall be determined, hypothesis about possible values (not necessary)

2. — (hints about weaknesses in the knowledge base)

Knowledge used: (domain) knowledge about the construction of questions/experiments.

Knowledge generated: answers to the internal questions

Tasks generated: —

Contributions from AI:

Questions typically arise in the form of questions to the user and are used in virtually every AI system. On the other hand, experiments are used almost exclusively in *automated discovery* systems.

Basically, four different motivations for the search for knowledge can be distinguished:

1. The existing theory needs to be completed.
2. A recall of knowledge is unsuccessful, or the information searched for can not be inferred.
3. The information found is rather uncertain.
4. Several answers are found. However, some of these contradict each other.

Based on these motivations three different types of experiments can be distinguished (motivation 2 and 3 are rather similar).

Exploration: Search for new information, which is not contained in the knowledge base. The operator is neither supplied with a phenomenon to be examined, nor with a hypothesis.

[9]For example, each problem found can be regarded as an internal question.

Determination: Search for a hypothesis. The operator is supplied with a phenomenon, which shall be examined, but not with a hypothesis about the outcome of the experiment.

Verification: Verify a hypothesis or distinguish between competing hypotheses.

From a psychological point of view exploration can be regarded as a purely curiosity-driven behaviour. By supplying a weakness in the knowledge base, which shall be remedied, an exploration experiment can be changed into a determination experiment. In section 6.2.3 ("Problems in the Knowledge Base") we examined how such weaknesses can be recognized. Real exploration is seldom used in AI systems. However, the LIVE-system ([She93, p. 151]) uses purely random exploration. This is based on its third heuristic:

> Once in a while, randomly explore some not-yet-explored actions with random parameters.

The results of these explorations can then lead to the discovery of new problems. That is, the discovery of problems can lead to exploration experiments, the results of which can in turn lead to the discovery of new problems. This shows that these two operations strongly interact.

Determination experiments can be expected to be important in the preparation stage, as in this stage usually not enough problem knowledge is available in order to generate an adequate hypothesis. On the other hand, verification experiments will be most often used in the verification stage, because they offer the possibility to assess the quality of a hypothesis either absolutely or relatively to other hypotheses.

Therefore, the different types of experiments are typically used during different stages of the creative process.

In principle, the "search for knowledge"-operator can be understood as a translation operator. It translates an internal question into an external representation and, correspondingly, translates the answer into an internal representation. In general, the representations used in symbolic AI have a direct correspondence to entities in the external world. Therefore, questions to the user can be translated by a simple mapping. The same holds true for the answers. This translation is rather simple, because the set of meanings which can be encoded in the representation of the system is a subset of the meanings intelligible to the user.[10] With respect to the answer it must be ensured that it belongs to the subset of representations intelligible to the system. Because this is rather simple, this translation process did not receive much attention in AI.

However, if this subset relation does no longer hold, then problems can arise which are not well examined, yet. This can be illustrated with the Charlie-problem (cf. [Wei89]). Here, the problem is to enlighten a situation by only using yes/no-questions. In this case the external vocabulary is a subset of the internal vocabulary. Consequently, a question like "What did Charlie die from?" must be translated into a complete plan using several questions.[11]

This quality of translation which sticks to this operator gets even more obvious in the context of experiments. Here, an internal question must be translated into an experiment setup and a plan for conducting the experiment. This special case is rather well examined, because it is very important in the context of *scientific discovery*. Consequently, we will base the rest of this discussion on approaches to experimentation.

Many different approaches to experimentation exist. These vary considerably with respect to their complexity. A very simple approach is employed in the DIDO-system ([SM93]). This system simply questions the environment using its internal vocabulary; no experiment construction is necessary. A somewhat more complex approach is used in the LIVE-system (cf. page 61). Here, the

[10] This is not necessarily true for a discovery system, which is able to generate new representations.

[11] Although not identical this task problem bears some similarities with the task of diagnosis. Here, the task is to induce a diagnose based on symptoms incrementally queried from the user. The symptom the user is asked for at any moment must be carefully selected, as not every *possible* symptom shall be determined for a diagnose.

system uses its planning abilities for constructing and carrying out a plan, which brings the environment into the necessary state.[12]

The KEKADA-system uses a heuristics-based approach to experiment construction. This approach is based upon heuristics called 'hypothesis-generators' and 'experiment-generators'. In both classes domain-dependent and domain-independent heuristics exist (cf. [KS88]). The hypothesis-generators propose a strategy of experimentation and the experiment-generators instantiate an experiment. However, these experiment specifications are still of a rather abstract nature (cf. [KS90, p. 270]), e.g., "carry out an experiment with ornithine and ammonia with certain concentrations in liver using the tissue slice method". The experiments are carried out by asking the user for the results.

These approaches can be applied for constructing exploration and determination experiments. In the case of a verification experiment an expectation for the result of the experiment can be made up using mental simulation. However, if an alternative hypothesis exists, as is often the case in *scientific discovery*, then it is ingenious to construct a discrimination experiment, i.e., an experiment, for which the different hypotheses yield different results.

Rajamoney (cf. [Raj93]) suggests an approach for constructing such experiments. This approach is called DEED (design of discrimination experiments based upon explanation differences) and is based on the following assumptions:

- Only few differences between the competing theories exist. This is typically the case if a basic theory exists, and the choice is only between two competing hypotheses for augmenting the theory.
- An experiment is known for which both theories predict an identical outcome, but this prediction is based on *different* explanations. This requirement is not very restrictive, too, as competing hypotheses are often induced from one experiment.

The approach only deals with physical experiments. These experiments are modeled using *Qualitative Process Theory* (cf. page 72).

The DEED approach is based on a stepwise transformation of the original experiment into a discrimination experiment. This transformation is conducted using the following modifications (cf. [Raj93, p. 188]):

- addition or removal of objects
- changes in the configuration of the objects (e.g., removal of connections between objects)
- changes to the starting conditions

These modifications are not guaranteed to lead to a discrimination experiment. Consequently, the resulting experiment must be evaluated using qualitative simulation (cf. section 3.5.1 "mental simulation").

The DEED approach allows the construction of a discrimination experiment. However, not all discrimination experiments are equivalent. Instead, experiments having different information content can be distinguished. These differences become especially prominent, if more than two hypotheses compete. Ruff and Dietrich (cf. [MS93, p. 122]) examined different strategies for *experience selection* (cf. appendix B "information filtering model"). They came to the conclusion that the best strategy is to select an experiment which halves the set of hypotheses. However, the strategy to select an experiment which eliminates at least one hypothesis was nearly as effective.

The usefulness of an experiment also depends on the question posed. Answering some questions is more important with respect to the quality of the knowledge base than answering other questions. We discussed this point already in section 6.2 "problem recognition".

[12] Basically, all approaches to experiment construction can be regarded as planning (or as the adaptation of plans, see the DEED-approach below).

Comparison with Psychology:
In AI as well as in psychology the process of experiment construction has received little attention to date. Only recently the interest in this process increased. One reason for this may be, that this process is often based on rather domain-specific knowledge. For example, in KEKADA (which closely models the behaviour of Hans Krebs) 5 out of 8 experiment generators are considered domain-dependent.

While humans consider secondary aspects, like the observability of the results, such aspects have not received any attention in AI up to now. Also only few AI systems (if any) are able to evaluate different possible experiments with respect to their information content.

In section 3.4 we mentioned that humans are able to acquire a strategy of questioning by practice. To the contrary, as far as we know, in AI experiment construction has not been combined with learning.

6.4 Activation of Knowledge

6.4.1 Association and Recall

Two different ways of retrieving knowledge exist: recall and association. While recall happens in a problem solving situation with a precisely defined goal, associations are made without any predefined purpose. Associations are used for enriching the problem solving context. However, the same techniques can be used for recalling knowledge and for associating knowledge. Consequently, a clear distinction between these operations is not always possible. Therefore we present them here together.

Usually, even in the case of a recall of knowledge, multiple matches are possible. In this case additional constraints must be used in order to select a small set of answers. In case of a recall a possible constraint is: if the operation is successful within the active knowledge, do not search the inactive knowledge.

Operator: recall / association

Scope: We will show how indices can be selected and used to search memory. Additionally, we will point out some strategies for choosing new indices in case a recall fails.

Precondition:

1. *Recall* — in a problem solving situation some information (a fact, rule, or case) is missing, which can be rather exactly characterized (e.g., how long is 1 mile, how can two numbers be divided).
2. *Association* — earlier operations failed due to a lack of active knowledge.

Purpose of call:

1. Urgently needed information shall be retrieved either from the active knowledge or from the inactive knowledge. (In the second case it will be added to the active knowledge.)
2. Additional information shall be added to the current problem solving context.

Input: 1. a key for characterizing the searched information

2. virtually any information from the current problem solving context will do[13]

Knowledge used: —

Knowledge generated: The retrieved knowledge becomes activated.

Tasks generated:

[13]However, the better the information is selected, the more helpful the retrieved knowledge will be (cf. section 6.4.1.1).

1. Upon failure a task may be generated for retrieving the necessary knowledge from the environment.[14]
2. —[15]

Contributions from AI:
In combination with the evaluation of the retrieved knowledge the operations described here can be regarded as an acquisition filter (cf. appendix B "the information filtering model"). Association as it is proposed in our model is typically no issue in AI, as most systems do not have a two stage memory model like the one we use here. There is one important exception however: the CYC-system. It also uses problem solving contexts (PSC) (besides other kinds of contexts) as we do (cf. [GL92]), however Lenat and Guha do not describe how PSCs are filled with knowledge.

Among the operations examined in AI, the retrieval of analogous situations is closest to association. Therefore, we will concentrate in our description of association on the material presented in this context. However, there is a notable difference between association as we propose here and the retrieval of source situations for analogical transfer as it is typically viewed in AI systems: while we regard association as an independent operation which may trigger an analogical transfer, the retrieval of source situations is usually regarded as being subordinated to the analogy operation. We will also allow for the second view, however this will be regarded as a recall of knowledge. Again this emphasizes the tight interconnection between recall and association.

The retrieval mechanisms for source situations often can be formally described by a similarity measure as is shown in [RW91]. However, a more implementation-dependent viewpoint allows for a better comparison with the ideas proposed in section 3.5.2. Therefore, we will base most of this description on Halls survey article on analogical reasoning ([Hal89]).

6.4.1.1 Choosing Indices

Hall distinguishes three types of indices, which can also be used for classifying the indices used in recall:

Nonselective indices: Arbitrary features in the current situation are used to index memory.

Task-specific indices: Only a subset of the features are used, which are important with respect to the task the system has to solve (e.g., in a system for story-understanding: actor, action, object, etc.). The selection of these indices is usually made a priori by the developer of the system.

Task-independent indices: More abstract features are used for retrieval (e.g., aims, plans, causal relations). These are also called *semantic categories*.

In [KW93] and [RG92] the authors hold the view that in *creative design* the most important indices are based on the problem specification. This would point to *task-specific* and *task-independent* indices. The retrieval based on *semantic categories* is typically very goal directed and should therefore be regarded as *recall*. Typically, the different types of semantic categories recognized in one system is very small. Accordingly, the number of independent keys which can be generated from one situation is small, too. Therefore these systems are usually able to form alternative keys.

In the SWALE-system described earlier (cf. page 60) the type of anomaly occurring in the story is used for retrieving possible explanations. If this fails causal preconditions of the anomaly and unusual features of the case are used for recalling explanations.

[14] The operation has already been called from another operation. That operation will be continued upon successful completion of the retrieval.

[15] New tasks will probably be generated by the evaluation process if the retrieved knowledge is accepted for inclusion in the activated knowledge.

A similar strategy is used in the CYCLOPS-system (cf. page 107). In this system cases are recalled in order to repair bugs in the current design. The bug is used as an initial key, if this fails the causal preconditions of the bug are used in turn.

The strategy used in the EUREKA-system (cf. page 89) is a little bit different. The EUREKA-system is designed to solve problems in the domain of physics. Therefore, its central problem is to find applicable (and helpful) equations. The first key used for retrieval is the variable searched for. If this does not result in the retrieval of any applicable equation, then other keys are used in the following order: new concepts[16], relations between objects, features of objects, and the objects themselves.

We see that usually the systems begin with an attempt to recall goal-related knowledge. If this fails they rely on cues drawn from the context or in some other way related to the goal. This can be regarded as a smooth transition between recall and association. A recall based on context-features can hardly be distinguished from an association.

6.4.1.2 Combining Indices

While task-independent indices usually are scarcely available, non-selective indices are available in large numbers. Consequently, here the problem is not to generate additional indices, but to combine the existing indices in a useful way. This kind of retrieval can be often described by a similarity measure: based on the features identical[17] in the current situation and the potential source situation a value (a similarity) is assigned to each possible source situation. This number can be modified by a weight assigned to the different features.

Retrieval mechanisms which use task-specific indices are often built in a similar way. However, as we mentioned already, an a priori commitment is made about the representational elements, which may point to future reusability of the source situation.

Winston's MACBETH-system ([Hal89]) also uses task-specific indices. The method employed in this system for computing the similarity between situations is especially interesting, because it is based on the concept-hierarchy of the system.

Beginning with the concepts[18] in the current situation the concept-hierarchy is traversed along its AKO-links (a kind of) towards more general concepts. Each concept representation has a slot called APPEARS-IN, which records all situations in which an instantiation of this concept or one of its specializations occurs. For each concept traversed a vote is given for all situations appearing in the APPEARS-IN-slot. This vote counts in direct proportion to the salience of the concept in the current situation from which the traversal started and in inverse proportion to the number of the situations on the APPEARS-IN-slot. All these votes are added and the situation with the highest value is retrieved.

Indexing mechanisms which are based on similarity measures have two important disadvantages:

- The number of potential sources is very large.
- The recognition of the source situation is based on the details of the situation and not on abstract properties. Therefore, it is difficult to find more abstract or even cross-domain analogies with this approach.

One possibility to avoid these problems is the use of task-independent indices. A second possibility is to impose additional constraints on the analogical mapping. However, because the enforcement of these constraints is inseparably intertwined with the elaboration of the analogical mapping, we will defer the discussion of these constraints to section 6.5.2.

[16] We are referring here to concepts which have not been part of the partial problem solution before and not to concepts which have been newly introduced into the ontology of the system.

[17] This can be relaxed to similar features, if a similarity on features has been defined.

[18] Only those concepts are used which play the role of actors, actions, or objects.

Comparison of AI Contributions to "Association and Recall" with Psychology: Among the relations, we mentioned in section 3.5.2, which can lead to the generation of associations, *similarity* received the most attention. This is due to the limited interest the phenomenon of association found in AI up to now. We could not find any evidence for the use of the relations of *contrast*[19] and *contiguity*. Also *cause-effect* relations are not used. However, as we will see in section 6.5.2, many analogy-based systems try to preserve these relations in an analogical transfer. *Part-whole* relations are typically not used for association, either. However, the CYCLOPS-system (cf. [Nav91, p.98]) is able to use this relation in a limited way, because it can retrieve parts of previous cases for problem solving. But this is not the typical use of this relation in human association.

The secondary association laws (vivacity, recentness, and frequent repetitions) are also not used in AI. However, they should probably be used as hints to the importance of the corresponding situations.

With respect to the retrieval mechanism, Anderson proposes a spreading activation scheme for the selection of the analogical situation. A similar scheme is used in the MACBETH-system. However, this kind of scheme can only be used with non-selective or task-specific indices, because it implies that a large number of keys can be generated from the current situation. This is usually not possible with task-independent indices as AI systems are only able to recognize a small number of semantic categories. Additionally, the recognition of semantic categories requires a large expertise in a field. As a creative problem solver needs to be able to move to new fields, it cannot have such knowledge, at least not at the beginning.

Association in humans may be based on all three indexing schemes. Which type of indices is used at any moment may depend upon the current focus of attention. That is, those features of the situation which draw the most attention of the reasoner are used as indices. This view is proposed by Gentner (cf. [Hal89]).

6.4.2 Mental simulation

Basically, one can regard mental simulation as the mental execution of a plan. If this plan includes the usage of a device, that device has to be simulated, too. Some authors use the term *mental simulation* only for the simulation of device behaviour. Kolodner and Wills ([KW93]) emphasize the importance of mental simulation for testing the robustness of a design.

Operator: mental simulation (qualitative simulation)

Scope: Several techniques for estimating the results of the application of a plan are discussed. Especial emphasize is given to techniques for modelling device behaviour. The interpretation of the results thus achieved is not part of this section.

Precondition:

1. *Activation* — insufficient knowledge about a task has been activated.
2. *Evaluation* — a plan or a design needs to be tested.

Purpose of call:

1. retrieve knowledge, e.g., what subtasks need to be performed, what conditions influence plan execution
2. test robustness (correctness) of a plan (design), e.g., by variation of the boundary conditions

Input: a plan (a device description), (boundary conditions)

Knowledge used: background knowledge for interpreting plans or device descriptions

[19] For example, this could be used with analogy for finding out, what *not* to do.

Knowledge generated: result of the plan (including side-effects), new problem solving criteria, activation of knowledge

Tasks generated: modify the plan (design) depending on problems or new criteria found

Contributions from AI:

In the context of planning, mental simulation can be regarded as the symbolic execution of a plan. However, this technique is seldom used for planning, because usually these systems are based on the assumption that all effects of the operators are known beforehand and the plans are built accordingly. The refinement of an abstract plan can also be regarded as an instance of mental simulation, because some conflicts become only visible when the steps are expanded.

Because mental simulation is a means of predicting the outcome of a proposed solution it is also called *projection*. Kolodner ([Kol93]) distinguishes three different types of projection:

1. Case-based projection
2. Simulation-based projection
3. Model-based projection

Typically, case-based projection is used, if no detailed causal model of the domain exists, but there exist experiences with similar situations. The underlying assumption of this approach is, that the current case will yield results similar to previous analogous cases. This technique can also be used for recognizing potential problems. JULIA and PERSUADER (cf. [Kol93]) use this technique to this purpose.

Simulation-based projection is based on having a component which is able to simulate the generated hypothesis (e.g., a plan, a device, etc.). This simulator may be a separate component, like in CHEF, a system for generating recipes (cf. [Kol93]), or in the system proposed by Silverman et al. ([SHM91]). Alternatively, the simulator may be part of the system itself. This approach is taken in EURISKO (cf. [Len83a]). This system designed fleets for the Traveller Trillion Credit Squadron (TCS) war game and tested them by simulating a match in which these fleets participated. The goal of this approach was to test the utility of the heuristics EURISKO generated for fleet design. In simulation-based projection a detailed model of the domain is either not accessible for the program (cf. [SHM91]), or it is not in an appropriate form, like in EURISKO (knowing the rules of a game, does not tell you what the best move is).

Contrary to this, in model-based projection a sound causal model is needed. This is most often available in the context of device design. Moreover, the need for simulation is especially prominent in this context, because a device description is inherently static, while many properties of a device can only be recognized based on a dynamic description (behaviour) of the device. This is also true for experiments, as an experiment can often be described as a plan for using a device. When simulating the plan, the device has to be simulated, too.

Some authors point out, that three representational levels for a device can (and must) be distinguished (cf. [Kui93b, RG92]):[20]

structural: the structure of the device (static physical description)

behavioural: the potential behaviours of the device, this includes all effects the device has (not only the desired ones) under all conditions (e.g., a car as a weapon, a car for transporting people, etc.)

functional: the desired behaviour of a device (e.g., a car for transporting people), this is sometimes also called the requirements or the specification of the device

[20] A similar distinction can be seen in software design: here the source code of a program, the effects (function) of a program, and its specification must be distinguished.

Based on a structural description only a behavioural description can be deduced. Usually the requirements are given at the outset and it must be verified that the behavioural description is a subset of the functional description. Deriving a functional description from a behavioural description can only be done based on background knowledge about the goals of the problem solver.

How a device can be simulated depends on the type of the design (e.g., is it of chemical, architectural, or physical nature). The simulation of physical processes is probably one of the most important types of design, as designs in the areas of civil or electrical engineering fall into this category, too. Accordingly, the qualitative simulation of physical processes received much attention (cf. [dK93]). Several calculi for qualitative simulation have been developed: the **Q**ualitative **D**ifferential **E**quations (QDE) calculus is based on an abstraction of the calculus of ordinary differential equations ([Kui93a]). The **Q**ualitative **P**rocess **T**heory (QPT) introduced by Forbus (cf. [Raj93]). Other approaches to the simulation of designs include the behavioural representation language (BRL) (cf. [BGP94]).

We want to illustrate this kind of description with a short overview of QPT. QPT is based on two relations between quantities:

1. *Qualitative Proportionality.* q_1 is qualitative proportional to q_2 ($q_1 \propto_{Q\pm} q_2$), if q_1 is monotonic increasing or decreasing in its dependence on q_2.
   For example, the acceleration of an object is qualitative proportional to the force on it.

2. *Direct Influence.* q_1 is directly influenced by q_2 ($I \pm [q1, q2]$), if the derivative of q_1 equals q_2, assuming there are no other influences on q_1.
   For example, the velocity of an object is positively influenced by its acceleration.

For example, if we have a water tank with an inlet and an outlet, the behaviour of the water level is defined by the following two relations:

- `(I+ (amount-of` *tank-water*`)` *in-flow-rate*`)` and
- `(I- (amount-of` *tank-water*`)` *out-flow-rate*`)`

In this example we can deduce, that if the water level is only influenced by these two relations[21], then the water level is increasing, decreasing, or steady, if the total flow-rate (*in-flow-rate* − *out-flow-rate*) is positive, negative, or zero, respectively.

As we mentioned already in section 3.5.1, mental simulation combines the retrieval of knowledge with a limited amount of inferencing. In mental simulation different types of knowledge can be retrieved, like rules, their conditions of applicability, and general constraints (e.g., physical laws). Mental simulation is in a better position to retrieve this knowledge, because it can use the plan (design) for *recalling* the knowledge, instead of having to rely on association.

The inferences are limited to making explicit implied effects of the design and conflicts between different parts of the design (plan). However, this can be very important because it sets the stage for adaptation.

Comparison with Psychology:

In section 3.5.1 we saw that human problem solvers use mental simulation mainly for two purposes:

- The activation of knowledge.
- Limited inferencing of additional knowledge.

Especially case-based projection can be regarded as a method for activating knowledge by mental simulation in AI systems. Some systems use this method for recalling possible adaptations, others use it for retrieving possible problems of the current situation. However, only few systems use this technique today, although it is an important one, especially if no complete domain theory is available.

[21] Because the QPT is based on partial causal relations, closed-world assumptions are formed for each inference.

Mental simulation is used by evaluation for identifying flaws in a design. While mental simulation produces information about a design, evaluation uses this information for judging the adequacy of the design.

Even for finding problems the capabilities of AI systems are still rather rudimentary, compared to those of humans. There are different approaches to problem recognition using mental simulation:

1. Simulate the design (plan) in a typical situation. (Does the design work basically?)
2. Simulate it with varying boundary conditions. (Is it robust?)
3. Simulate it in the context of an application plan.[22] (Is it useful?)

The first of these approaches can be regarded as typical for evaluation, while the other two are more apt for verification. Of these three approaches most AI systems use only the first approach to date.

6.4.3 Forgetting

Operator: forgetting

Scope: Forgetting is sometimes nearly indistinguishable from reindexing (cf. section 6.6.2.1) as this may also involve the deletion of memory structures. In this section we will try to draw a borderline between these two operators as follows: If the deletion is irreversible then it is forgetting, otherwise it is reindexing.

Precondition:

- The knowledge item has not been used for a longer period of time.
- The knowledge item has not been cost effective.[23]
- The active knowledge is too large.

Purpose of call: Try to reduce the size of the current problem solving context.

Input: —

Knowledge used: active knowledge

Knowledge generated: —

Tasks generated: —

Contributions from AI:

Forgetting is tightly connected to the retention filters described in the information filtering model (cf. appendix B). However, in our model (contrary to the "information filtering model") forgetting only removes knowledge from the active knowledge and not necessarily from the knowledge base. Consequently, it can be added again to the active knowledge at a later time.

As Markovitch and Scott point out (cf. [MS93]) forgetting can be employed for two goals:

- for producing better solutions
- for being more efficient

[22] Kolodner and Wills ([KW93]) mention one application of this approach by their subjects. The subjects should design a device for transporting eggs. When considering a device which threw the eggs they recognized by repeatedly simulating the device that all the eggs would land in the same place and would break.

[23] Either it has not been useful in the problem solving task, or the problem solving costs caused by the knowledge item have been higher than the benefits caused by it.

Forgetting is most often employed for making a problem solver more efficient. In this case the deletion of a knowledge item is usually based on a metric like the following one, which is used in PRODIGY for assessing the utility of control rules:

$$(\text{Mean time saving} \times \text{Probability of application}) - \text{Mean Match Costs} \tag{6.2}$$

In the FUNES-system (cf. [MS88]), a system which learns macro-operators in a simple planning domain, a similar metric was employed. Additionally, a random deletion of operators has been tested. While this strategy could also improve the performance of the system, more could be forgotten using the metric before forgetting had a negative effect on the performance of the problem solver. Additional experiments with this system demonstrated that it is less effective to learn a smaller number of operators in the first place than learning a large number of operators and forgetting most of them.

Other systems simply forget knowledge, which has not been used for a longer period of time (e.g., MACLEARN).

While the systems described above employed forgetting for being more efficient, some systems employ forgetting for generating better solutions. For example, in the DIDO-system (cf. page 61) a result of an action, whose probability falls below 0.05 is simply forgotten, because it is probably due to noise in the environment. Additionally, if a distinction in the classification hierarchy of the system does not contribute to its classification accuracy, the subclass is again merged with its parent class. Similar transformations are made in the EUREKA-system. However, these transformations are made for improving the retrieval of information from memory. Consequently, we will regard them as reindexing of knowledge and describe them in section 6.6.2.1.

Etzioni ([Etz88]) describes the *Hypothesis Filtering Method* for turning an arbitrary learning algorithm into a PAC-learner. It works by eliminating all hypotheses produced by the learning algorithm, which do not comply with the quality measure given.

Wnek and Michalski ([WM94]) and Wrobel ([Wro94]) give algorithms, which eliminate (forget) part of the representation. In the MOBAL-system ([Wro94]) a concept can be formed as need arises (cf. page 97). If the need for this concept no longer exists, then the concept is eliminated. (A concept is needed, as long as there exist rules which use it.)

Wnek and Michalski (cf. [WM94]) present the classification algorithm AQ17-HCI, a successor of the AQ-15 algorithm which employs ideas from *constructive induction* (cf. section 6.6.3.2). In this algorithm new attributes and values (constructed from existing attributes and values) can be added to the representation, if this improves classification accuracy. Attributes can also be eliminated, if they do not contribute significantly to the classification accuracy.

The basic assumption underlying all these retention filters is a certain kind of monotony. A knowledge item, which has not been important up to now will not be important in the future. This is combined with a hill-climbing approach. One element after the other is evaluated and forgotten. None of these algorithms consider interactions between the various knowledge items. For example, an item A might be eliminated and, consequently, may render item B virtually useless. This may lead to getting stuck in local maxima.

A somewhat different approach would be to use the same criteria for an utilization filter and base the forgetting only on the use of the knowledge items. This would lead to somewhat higher costs, but at the benefit of having the knowledge still available (for some time) if it is needed. If no such emergency situation would come up, then the knowledge item could eventually be forgotten.

Comparison with Psychology:

In section 3.5.3 we described two different approaches to forgetting:

1. forgetting by inference between possible answers
2. forgetting by reduction of activation

In the above survey of AI approaches to forgetting, we could identify no implementation of the first psychological model. This is due to the fact that our survey only covers symbolic methods. Neural networks show a behaviour which is comparable to this model.

With respect to the second model we saw that some AI systems do indeed implement this approach, however, most systems combine it with additional criteria. But, as we discussed above, these AI approaches can be reduced to utilization filtering combined with deactivation over time. Therefore, the deactivation model is not only a good psychological model for describing forgetting in humans, but it can also be regarded as a suggestion for improving forgetting in AI systems.

6.5 Construction of New Knowledge

As we pointed out already, general problem solving abilities are a prerequisite for a creative problem solver. This will become especially obvious in this section, because here we will discuss the construction of new knowledge (typically in the form of solutions to a problem). We will distinguish four different methods for constructing new knowledge:

1. Planning (Searching)
2. Theorem proving
3. Transfer
4. Induction

Out of these operators, operator 1 and 2 have received a large share of attention since the very beginning of AI. Consequently, a large number of books and other literature about these topics exists. Therefore, it is impossible to cover this subject adequately here. Further, these two operators have no *special* relation to creativity. Hence we mention them here only for the sake of completeness, but will not discuss them in detail.

The operators 'transfer' and 'induction' have also received much attention in AI research. Consequently, it is impossible to describe these operators completely, too. However, they are more important to creativity in a problem solver, because they lead to the generation of new hypotheses; an action which is often regarded as restructuring in psychology. Consequently, we will outline the basic ideas underlying the implementation of these operators. While operators 1 and 2 generate knowledge which is correct with respect to the internal model of the problem solver[24] operators 3 and 4 may generate hypotheses which may contradict this knowledge. Therefore, a critical evaluation of this information is especially necessary.

6.5.1 Inference

In AI inference operators come in a large variety: planning, configuration, design (from first principles), theorem proving (deduction), etc. These operators replace the more general 'inference' operator used in the psychological part of this study.

Operator: planning / configuration / design / theorem proving

Scope: Theorem proving includes such different approaches as resolution and term rewriting. In term rewriting critical-pair-generation can be regarded as inference, while term rewriting itself can be understood as reformulation. Therefore we will return to this point in section 6.6.3.1 (reformulation).

Precondition: simpler approaches (like adaptation of existing solutions) are unsuccessful.

[24] That is the generated information is usually not contradictory to the knowledge of the problem solver, however it may be falsified by the environment.

Purpose of call: generation of a (partial) solution to a problem (e.g., a design, a plan, a proof of a theorem)

Input: problem specification (goal)

Knowledge used: rules (operators) valid in the problem domain

Knowledge generated: a new (partial) problem solution or failure (with reasons)

Tasks generated: remedy causes of failure

Abstraction based approaches (i.e., multi-level approaches) to the implementation of these abilities are especially important to an understanding of human behaviour, as they reflect more closely human behaviour than single-level approaches.

Despite the many efforts in these fields, there are only few efforts to date for integrating these operators with other operators described in this study (e.g., concept formation).

Usually, AI systems employ correct implementations of deductive operations like *modus tollens*. This is contrary to humans, who often make mistakes when trying to solve such problems. However, this restriction to necessarily correct solutions also has a drawback: humans use efficient heuristics for solving such problems, while AI systems usually do not use an heuristic approach as this may sometimes yield erroneous results. This in turn leads to the fact that an AI system is often unable to derive a solution in situations, in which a human guesses the correct answer.

6.5.2 Transfer

Sometimes the transfer of knowledge is also regarded as restructuring. However, because the transfer of knowledge is used for generating new knowledge from scratch (although it uses other knowledge structures as some kind of blue-print) and not for changing existing knowledge structures around, we will regard it as an operator for constructing new knowledge. In AI the transfer of knowledge is usually called *analogy*.[25]

Operator: Analogy

Scope: Elaboration of a mapping from an analogical situation onto the current situation and application of the mapping. Neither the evaluation nor the consolidation of the results of this transfer is part of this section.

Precondition: Simpler approaches (like adaptation of existing solutions) are unsuccessful and there exists a solution to a similar problem (cf. association).

Purpose of call: generation of a (partial) solution to a problem.

Input: problem specification, similar solution

Knowledge used: domain knowledge

Knowledge generated: a new (partial) problem solution or failure (with reasons)

Tasks generated: If successful check the induced result, otherwise remedy causes of failure

Contributions from AI:

Analogy has received very much attention in AI. However, while most authors try to build a working system using analogy, only few try to explain why analogy (often) works. One notable exception from this rule is made by Lenat (cf. [Len83b]). In his theory of heuristics he tries to explain the usability of analogy. Because his discussion mostly refers to heuristics, we will defer it until section 6.7.6.

The retrieval of an analogical situation was already discussed in section 6.4.1. Now, we will shortly discuss the elaboration of an analogical mapping. However, usually these two processes are intertwined, as during the retrieval of a source situation, first contacts between the features of the current situation and features of the analogical situation are established. Consequently, the elaboration process starts with the features which were used during retrieval (indices, cf. section 6.4.1). Dependent on the type of indices used, two types of construction processes can be distinguished:

[25] Case-based reasoning (CBR) is also often regarded as a form of analogy (cf. [RW91]).

Bottom-up: Beginning with a mapping of the basic elements of the representation higher and higher relations are mapped. Basic mappings, which do not participate in higher mappings, are deleted and those which do are enforced. Munyer proposes this approach for a system employing analogy in the area of automated theorem proving. This system starts with the mapping of symbols, continues with the mapping of terms, etc.

Top-Down: The elaboration of the mapping starts with the most abstract entities. Typical representatives of this approach are systems which use task-independent indexing, as these begin with the mapping of the semantic categories used for retrieval.

Both methods try to find the analogical mapping (and consequently the analogical situation), which supports the most (and the most important) abstract relations. Especially causal relations are regarded as being important and should consequently be transferred.

During the construction of the analogical mapping several constraints are usually enforced in order to prune the search space. According to Hall ([Hal89]) these constraints can be classified into three different categories:

1. *Structural consistency.*
   This includes constraints on the mappings of the basic items in the situation description (e.g., items must be type-compatible) as well as constraints on the relational structure (e.g., preserve as many relations as possible). Constraints of the second type are often implicitly enforced by the construction method. We saw this above in the description of the bottom-up approach.

   Kodratoff ([Kod90]) emphasizes the importance of transferring causal relations for achieving *good* analogies. An analogy in which causality and similarity are independent (i.e., the causal relations are not transferred) is called *trivial*.

2. *Preserve a priori determined semantic categories.*
   This is usually enforced by a retrieval mechanism based on task-independent indices combined with a top-down approach to mapping construction. Sometimes this constraint is relaxed to the criterion of *semantic similarity*: mapped elements must have *similar* meaning.

3. *Preserve knowledge which is important in the current reasoning context.*
   In a system which uses analogy for problem solving, the problem, the cause of the problem, and the solution would be important to map adequately during analogical transfer, while other features (e.g., the color of an object) would be regarded as unimportant. Due to this reason many systems try to transfer cause-relations. Constraints of this type are sometimes also called constraints of *pragmatic centrality*.

Most systems can only handle one or two types of constraints. However, the ACME-system (**A**nalogical **C**onstraint **M**apping **E**ngine) integrates all three types of constraints (cf. [TCH89]). The constraint of *structural consistency* is handled by adequately selecting the initial partial mappings. The constraints of *semantic similarity* and *pragmatic centrality* are satisfied by a constraint satisfaction process similar to the process of competitive learning employed in neural networks. This system is able to (re-)construct several analogies from the domain of chemistry previously given by humans. In the course of this task it is also able to discriminate between various potential analogies.

According to Boden ([Bod91, p. 176]) ACME — with the help of ARCS, a program for the retrieval of analogies, and a rich knowledge base — is able to understand (i.e., to interpret in terms of an analogy) even complex literary metaphors.

Often the elaboration of the analogical mapping is integrated with the evaluation of the analogy. This becomes especially obvious in the case of *derivational analogy*, where elaboration and evaluation of the mapping and adaptation of the resulting plan are tightly integrated.

Comparison with Psychology:

In section 3.6.2.1 we pointed out that humans often use knowledge transfer with self-generated problems. This approach can also be regarded as a reformulation technique: the problem is rephrased in

a form, which allows for the use of additional knowledge or highlights certain aspects of the problem (e.g., the combination of different sexes in the match-making problem, see page 26). However, AI approaches are based on the transfer of knowledge from *previously encountered* situations. This also holds true for most psychological studies.

Humans are able to transfer knowledge either based on similarities between elements of a situation or based on similarities between the relational structure of the situations. As we saw above AI systems are able to do so, too. However, if the commonalities between the source situation and the current situation are restricted to the relational structure, then retrieval of these situations becomes difficult. Here, task-independent indices may help.

In psychological studies usually a distinction is made between the transfer of results from a source situation to the target situation and the transfer of the proceeding in the source situation to the target situation (non-specific transfer). Both approaches have counter-parts in AI. While most implementations of analogy in AI realize the first proceeding, *derivational analogy* can be regarded as an example of non-specific transfer.

In section 3.6.2.1 we illustrated the human ability of enriching the source situation in order to make a transfer possible using an example from the PUPS-theory (introduction of the relation 'implement', p. 27). This ability is rather difficult to simulate, because in the beginning the similarity between the situations hardly exists. However, the GRAPES-system (Pirolli et al., [Hal89, p. 88]) has the capabilities to build analogical mappings even in such problematic situations. Other systems even use additional analogies for repairing an analogy, if problems are encountered (cf. section 6.6.4.2).

Only few systems, like ACME, are able to combine all three types of constraints when forming an analogy. According to Boden this ability makes ACME's behaviour psychologically plausible. While ACME is only able to elaborate a mapping when a source situation is given, ARCS can also find an appropriate source situation. The behaviour of ARCS is reported to be significantly similar to human behaviour in psychological studies (cf. [Bod91, p. 177]).

6.5.3 Induction

Operator: Induction

Scope: The subject of this section is the recognition of new regularities among knowledge items. The verification of these rules (hypotheses) is made in the evaluation process. Their adaptation to changing or previously unknown or unused information will be described in section 6.6.4.1.

Precondition:

- Hints to regularities arise
- The current knowledge is insufficient for problem solving

Purpose of call: Extension of the domain knowledge

Input:
- a focus on the kind of regularities which are especially important, together with experiences[26] *or*
- a hint, e.g., an example for a regularity

Knowledge used: (domain knowledge)

Knowledge generated: new regularity (rule)

Tasks generated: Try to validate the induced result

Contributions from AI:

Basically the terms *induction* and *generalization* can be identified in AI research. Two types of generalizations can be distinguished:

[26] Usually, hypotheses recognition is done on a set of ground instances among which regularities need to be recognized.

- an existing rule is made more general, i.e., it shall cover a larger set of instances.
- based on a concrete behaviour a general rule is induced.

In AI the term *induction* is often used to refer to the recognition of regularities between attributes in a set of knowledge items.[27] The regularities uncovered thereby are then used in turn for the classification of new elements.

Example 6.5.3.1
`has_wings(eagle).`[28]
`can_fly(eagle).`
Induction: **IF** `has_wings(`x`)` **THEN** `can_fly(`x`)`

Induction needs to be distinguished from *(conceptual) clustering*. The goal of *clustering* is to form a partitioning of a set based on similarity between elements.[29] We will study *clustering* in section 6.6.3.2.

The simplest kind of induction is the generalization of existing regularities, because these hint to an adequate form of a rule. Such a primordial rule can be generated in an analogical transfer. This type of induction has already been illustrated in the PUPS-model in section 3.6.2.2.

6.5.3.1 Generalization of Existing Rules

An analogical mapping used in analogical transfer can be regarded as a rule. Due to the fact that this mapping is biased by idiosyncrasies in the source and in the target situations, it is rather specific. Therefore, when generating a rule which shall apply in a wide range of situations it is necessary to remove these peculiarities. This is usually done by generalization, similar to the approach taken in the PUPS-model (cf. section 3.6.2.2). Some approaches to analogy use this method for making better use of their results. The approach used in the GRAPES-system is very similar to the PUPS-model, as it is mostly based on Anderson's ideas. In this system a transfer is described by a production rule. These rules are basically of the form `IF` *situation* `THEN` *solution*. GRAPES has two possibilities for generalizing its production rules: it may drop conditions on the situation or it may convert constants into variables.

A similar approach has been presented by Becker in JCM (cf. [Hal89, p. 47]). In JCM transfers are also represented by rules:

$$\{\langle \texttt{wears}^4\ \texttt{Cyrus}^2\ \texttt{BB}^2\rangle\texttt{:4}\ \langle \texttt{member}^4\ \texttt{BB}^2\ \texttt{suspenders}^4\rangle\texttt{:3}\ \langle \texttt{property}^4\ \texttt{BB}^2\ \texttt{red}^4\rangle\texttt{:3}\ \langle \texttt{location}^4\ @^4\ \texttt{Peoria}^4\rangle\texttt{:1}\ \langle \texttt{time}^4\ @^4\ \texttt{Thursday}^4\rangle\texttt{:1}\} \Longrightarrow \{\langle \texttt{member}^4\ \texttt{Cyrus}^2\ \texttt{fireman}^4\rangle\texttt{:4}\}$$

In this notation the numbers denote the criticality (i.e., the importance) of nodes (e.g., `Cyrus`) and kernels (e.g., $\langle \texttt{wears}^4\ \texttt{Cyrus}^2\ \texttt{BB}^2\rangle$). These criticalities are refined by further experiences with the application of this rule.[30] If the criticality of a node approaches 0 then it is treated like a variable. If the criticality of a kernel approaches 0 then this corresponds to dropping the condition. However, this approach is dissimilar to the approach taken in the GRAPES-system, because the generalization is not only based on the transfer, but it is also influenced by future problem solving experiences.

While the systems JCM and GRAPES use arbitrary features of the situations for generating rules, Winston's MACBETH-system uses only a subset of the available features. It is able to select relevant

[27] Often the examples are partitioned into sets and a description of set membership shall be generated. However, set membership can also be regarded as an attribute.
[28] We use a predicate-like notation for attributes, because this allows for a simple notation of the rules found.
[29] However, we can use induction for generating a general characterization of these subsets.
[30] This is an adaptation process, therefore we will describe it in more detail in section 6.6.4.1.

features, because it possesses a causal model of its domain. This knowledge is used to focus the transfer and generalization processes on the important features. To this purpose the intersection of the facts, which support the answer in the source and in the goal situation of the analogical transfer is formed. Facts which are only used as intermediate steps are removed. The conjunction of the resulting facts yields the premise of the rule.

However, often no regularity is available for generalization. Consequently, in these situations induction must start with the recognition of regularities. This will be the subject of the next section.

6.5.3.2 Recognition of Regularities

In this section we will assume that all regularities take the form:

`IF` *formula1* `THEN` *formula2*,

where each variable in *formula1* also occurs in *formula2*.

Usually, *formula2* consists only of one predicate, the classification searched for. Most induction algorithms are based on an attribute-list representation (e.g., ID3, cf. [RW91]). However, the results of these algorithms can also be simply transferred into the rule notation employed here (e.g., `IF` att1=val1 `and` att2=val2 `and` ... `and` attn=valn `THEN` attm=valm).

Basically two types of rules can be distinguished:

- exact rules
- probabilistic rules[31]

While in the first case the rules must hold for all examples, in the second case they need only hold for a substantial subset of the examples. With respect to probabilistic rules again two subclasses can be distinguished: either probabilistic reasoning can be used for the construction of the rules only (e.g., ID3), or it can also be used for the application of the rules (e.g., [CW91]). In the second case we need to augment the rules with their probability for being correct:

`IF` *formula1* `THEN` *formula2* `WITH` p

In this notation p denotes the probability that *formula2* is true if *formula1* is true.

Besides the logical rules we discussed up to now, some systems also try to recognize numerical regularities among multiple parameters (so-called *quantitative discovery* or *function finding*). Traditionally, this is a subfield of *scientific discovery*. More recently new approaches arose in the context of Knowledge Discovery in Databases (KDD). For example, $pV = RT$ is such a numeric relation.[32]

Numerical data can also be discretized in order to use it in the search for logical rules. Three different approaches are used for discretizing numerical data:

1. A threshold is selected and the attribute values are simply partitioned into values above and below the threshold.
2. The range of the attribute is partitioned into a given number of intervals of the same size.
3. The range is partitioned such that the (information theoretic) entropy gets maximized (cf. [CW91]).

Basically, there exist two different approaches to the recognition of regularities:

1. Recognition and generalization of regularities in the examples

[31] Here we will concentrate on approaches based on probability (and information) theory as these are very common among the uncertainty based approaches. However, most of what is said about probabilistic approaches holds true for other uncertainty based approaches, too.

[32] This relation from thermodynamics describes the connection between the pressure (p), the volume (V), and the temperature (T) of an ideal gas. R is a physical constant.

2. Test of possible rules

These two approaches can be regarded as search in the instance space (1) and search in the rule space (2).[33] Out of these the first approach is more common. Both methods are used in the search for logical regularities and in the search for numerical relations.

When searching the instance space more and more general regularities are formed. The starting point for this cycle is the description given by the data. In the most usual case, all the examples are partitioned according to the conclusion they make true. These examples can already be regarded as primordial premises. Then they are generalized as far as possible while trying to preserve the partitioning. However, some algorithms even allow that the premises may overlap. When generalizing premises some of the rules may become redundant and are eliminated. Some approaches to classification are based on probability (or information) theory. We will describe some basic ideas of these approaches in section 6.5.3.4.

An approach based on search in the instance space has been proposed for the discovery of numerical relations (cf. [WW91]), too. This method is employed in the KEPLER-system. It is based on the variation of two variables at a time and matching the resultant observation to a set of prototype functions. The resulting binary formulas are then stepwise combined to form a multivariate formula.

When searching the rule space all the possible rules are systematically formed as hypotheses and then compared to the available data. Typically the rule space is restricted by additional constraints. One example for this method is the BLIP-system (cf. [Wro89]). Here, so-called meta facts are given, which are used as prototypes of logical relations. These prototypes are instantiated to rules and the resultant rules are then tested. This approach can be efficiently used in BLIP (and in MOBAL), because these systems are based on a function-free horn logic. This approach is also used in *quantitative discovery* ([WW91, Sha91]). In these approaches a fixed set of function prototypes is matched against the observations. A more sophisticated approach is used in the BACON-system (cf. [LSB87]). Here the search in a larger rule space is guided by heuristics. However, according to Shaffer ([Sha91]) this does not improve the quality of BACON's results.

This second approach is basically limited by the available rule schemes. But search in the instance space is also limited by the available predicates (attributes), functions and connectives. In section 6.6.3.2 we will examine methods for alleviating this restriction.

6.5.3.3 Selection of Attributes

Typically, regularities hold only between few of the existing attributes of an item. Therefore, efficiency considerations suggest the restriction of the search for regularities to a subset of the attributes. The selection of a useful subset is a typical attention filtering problem.

However, to date most approaches to the identification of regularities ignore this problem. The most notable exceptions from this rule are those approaches, which are based on *constructive induction*. These methods are able to generate new attributes at need and to remove attributes which do not contribute to the success of the induction procedure from the representation. We will return to this subfield of induction in section 6.6.3.2.

Systems, which are based on the generalization of existing regularities (cf. section 6.5.3.1), can use the given regularity as a hint to the important attributes.

An approach, which does not fit into any of the two categories above, is used in the LIVE-system (cf. page 61). This system simulates a planner, which learns its knowledge about the effects of its actions by observation of the environment. For induction of the planning knowledge it does not use all differences between the situations before and after an action, but it restricts itself to a subset of objects and relations, which are closely related to the action. One can imagine the description of these two situations as graphs, where the nodes are objects and the relations are arcs. Beginning with the objects mentioned in an action (e.g., arm) the system compares both graphs. This search

[33] They can also be interpreted as bottom-up vs. top-down.

is terminated if different arcs or nodes are encountered, but is made in multiple directions. The basic motivation for this approach is, that differences which are caused by differences already found shall be neglected. Additionally, LIVE possesses the ability to refine the induced regularities, if they turn out to be insufficient.

6.5.3.4 Probabilistic Approaches

The detection of regularities becomes especially difficult, if the information, which forms the basis for the induction process, may contain errors. Problems become especially large for approaches which try to induce exact rules, as noise can make it completely impossible for these approaches to induce any (correct) rules.

However, these approaches can be extended to handle such situations, too. This is demonstrated by the JCM-system. Here, the criticality of a node or a kernel can be lowered so that an erroneous fact need not to be matched exactly. Additionally, any erroneously introduced conditions can be effectively removed by lowering their criticality with later experience.

Although these possibilities exist, probabilistic approaches are more apt for handling such situations. As we mentioned already, these approaches can be further subdivided into approaches using probability theory only for the generation of rules and approaches using it in the application of rules as well.

The ID3- and CN2-algorithms (cf. [RW91]) are examples of the first approach. The ID3-algorithm is based on the top-down construction of decision trees. In each step of tree construction a test of the attribute which promises the largest increase in information is added to the tree. The sequence of tests made on a path from the root to one of the leaves can be regarded as a premise of a rule. Contrary to this approach the CN2-algorithm is based on the generation of complete rules. In each step a rule, which promises the highest gain of information is added to the rule set. While the ID3-algorithm tries to give a correct classification of the given examples, the CN2-algorithm also allows rules which do not give a correct classification. In order to avoid an over-specialization of the rules generated by ID3, the constructed decision tree often gets pruned (cf. [CW91]).

The approaches based on the minimum description length principle (MDL-principle) are based on information theoretic considerations, too (cf. [Ped91]). The basic idea of this approach is to construct a theory $\mathcal{T}$ (in this context $\mathcal{T}$ is a set of rules) such that the term $l(\mathcal{T}) + l(z_1, \ldots, z_n \mid \mathcal{T})$ becomes minimal. In this notation $l(\mathcal{T})$ denotes the length of a machine-readable representation of $\mathcal{T}$ (in bits) and $l(z_1, \ldots, z_n \mid \mathcal{T})$ denotes the number of bits necessary to represent the observations $z_1, \ldots, z_n$ based on the theory $\mathcal{T}$ (that is only the deviations from the theory $\mathcal{T}$ must be represented). This approach can even be used for function-finding as Pednault showed in [Ped91].

An approach which uses probability theory even for applying the rules is given by Chang and Wong in [CW91]. Their approach is based on defining a correlation between attribute values and classifications. This is done by constructing a standard normal distribution, where the expectation value is computed based on the independence of class-membership and attribute values. Based on this standard normal distribution and the examples the probability can be computed for attribute values and class membership being positively or negatively correlated or independent. This probability can be transformed into a so-called *weight of evidence*, that an element with attribute-value g_k belongs to a class p. This measure has the advantage that the evidence for an element belonging to class p is given by the sum of the evidences for belonging to class p based on each attribute value. This allows to handle missing attribute values rather easily. An element is regarded as belonging to the class for which the largest evidence gets computed.

6.5.3.5 Induction of Causal Relations

A causal relation is a special kind of regularity: according to [LISG91, p. 466] one can assume that an event A *causes* an event B, if the occurrences of A and B are correlated and A precedes B. Causal

relations are especially important for reasoning, because they are central to the understanding and planning of events.

One can distinguish two interpretations of the word 'cause', which are in common use:

1. Whenever an event A occurs, event B will occur, too.

2. An event A makes an event B more likely to occur.
   (Rain makes car accidents more likely.)

These two interpretations roughly correspond to the two aforementioned types of regularities (strict and probabilistic). Probabilistic approaches are especially helpful if noise exists in the environment or either a cause or an effect is not observable (cf. [Paz93]). Causal induction differs from other approaches to induction mainly because it must take the aspect of time into account. This is necessary for deciding whether A causes B, B causes A, or both are caused by an event C. However, most approaches to induction avoid the explicit representation of time. Instead they model the sequence of events by the sequence of the event descriptions in the input stream, or by giving the system the ability to manipulate the environment and observe the results.

We examined already two systems which try to learn causal relations: DIDO and LIVE.

DIDO (cf. page 61) tries to build a class hierarchy for describing the different possible situations and to learn for each possible situation the results the different possible actions have and their probability of occurance. This system makes some basic assumptions about its environment: every action causes only one state change, which is immediately visible. Moreover, actions can be performed independently and no preparation is required for setting up a situation. Due to these simplifying assumptions the main problem for DIDO is the construction of the class hierarchy.

In the LIVE-system considerably less simplifying assumptions are made. As we saw already on page 61, the LIVE-system is based on the notion of prediction sequences. Therefore, this system is able to take into account not only the immediately preceding action, but also earlier actions and states. This is only done if no difference can be perceived between a state, in which an action has been performed successfully and a situation in which the action failed. The method which is employed by LIVE for comparing two situations was already described on page 81. DIDO as well as LIVE are restricted to environments in which all changes are due to actions of the learner.

Contrary to this the OCCAM-system (cf. [Paz93]) is a system which learns causal relations from observations of its environment in which several agents may exist. Additionally, the system allows that a cause may have no effect (noise), that it may have multiple effects, or the same action may have different effects. OCCAM possesses multiple components for learning causal rules.

The least knowledge-intensive approach is called SBL. This component uses a clustering algorithm for aggregating similar action-result pairs into clusters. Action-result pairs are generated by combining each action with each state-change. Rules are generated through generalization of the examples in the cluster. This is done incrementally as new examples are added to a cluster. If identical constants appear in the examples then they are replaced by the same variable, inducing an equality constraint. This constraint is removed, if later on an example is added to the cluster for which the constraint does not hold. Each of the rules thus generated is tagged with a confidence. This confidence is updated in every cycle based on the observed and predicted state changes.

Additionally, OCCAM possesses a component for theory driven learning (TDL) of causal rules. The TDL-component uses so-called causal patterns for the induction of causal rules. A causal pattern consists of three parts: cause, effect, and disposition. Objects which occur in the action and in the effect are represented by the same variable. Additional attributes, which are important with respect to the causality are enumerated under disposition. When the system forms a new rule, rules which correspond to existing causal pattern are preferred.

OCCAM is also able to learn new causal patterns. It does this by employing an eager approach (cf. [Paz93, p. 184]). For each rule generated by SDL it generates a causal pattern by retaining only the equality constraints on the rules as cause and effect and adding the other attributes to

the disposition slot. If another pattern exists with the same equality constraints both patterns are merged. TDL learns more reliable rules from fewer examples than SBL.

As a third method OCCAM uses EBL for learning (cf. [Paz93, p. 183]). Indeed, the three components are used in the order EBL, TDL, SBL if a prediction failure occurs. That is the more knowledge-intensive approaches are used first, falling back to a more data-driven method if the prior one fails. According to Pazzani (cf. [Paz93, p. 181]): "the rationale here [behind this approach] is that EBL is expected to produce accurate rules from fewer training examples than TDL, and TDL produces more accurate rules than SBL."

6.5.3.6 Knowledge and Induction

Generally, by using supplementary knowledge the quality of the regularities found can be improved and the number of examples needed for inducing them can be reduced. Nevertheless, most approaches to induction in use today do not use any background knowledge.

A major exception from this rule are approaches which are based on analogical reasoning. This is understandable, because performing a successful analogical transfer requires a large amount of knowledge. A good example for this is the MACBETH-system, because here causal relations play an important role in finding an analogical mapping as well as in forming a general rule, which summarizes the analogical transfer.

Again, this emphasizes the importance of causal relations. But other types of knowledge can also be important for the induction of regularities. For example, Hong and Mao (cf. [HM91]) describe an example in which their procedure did not find an adequate set of classification rules. However, if a certain rule was provided at the outset their procedure generated a compact set of classification rules.

Cai et al. ([CCH91]) use domain knowledge in a very simple way. They use a concept hierarchy for generalizing regularities: A regularity is generalized by replacing a concept by one of its generalizations in the concept hierarchy. A similar approach has already been used in the AM-system (cf. [Len82a]).

Other approaches to the use of knowledge in induction are based on the reduction of the search space. For example, Mooney ([Moo93]) proposes the IOU-method (**I**nduction **O**ver the **U**nexplained). Here, in addition to positive and negative examples of the concept, a generalization of the concept searched for is given. This additional information can then be used for simplifying the search for a concept description. A related approach was given by Hall ([Hal88]). Here, EBL is used for explaining a design as far as possible. Only those parts of the design, which cannot be explained thus are then subject to the induction process. The rules generated this way can then be used for explaining other designs, simplifying new tasks. This approach is similar to the approach used by Pazzani.

Pazzani and Kibler ([PK92]) describe an extension of the FOIL-algorithm — FOCL — for using knowledge in rule induction. It allows the use of various types of knowledge:

- Rules (i.e., a domain theory in the sense of EBL) can be used.
- An initial concept description can be used. (This must neither be a strict generalization nor a specialization of the concept that shall be characterized.)
- Various constraints (e.g., types, all arguments of a predicate must be bound to different variables, etc.)
- Predicates which shall be learned as intermediate steps.

The authors report that with the help of this knowledge considerable reductions of the search space could be made.

6.5.3.7 Influence of the Representation Language

The representation language determines which rules can be represented and consequently which rules can be learned.

The types of rules, which can be learned by the different systems, vary considerably: always a conjunction of predicates can be represented (e.g., in the form of decision trees). Some systems also allow negation and disjunction (e.g., BLIP and MOBAL). Sometimes it is also possible to represent equality constraints, e.g., in the form of non-linear variables (e.g., OCCAM). However, most approaches to rule induction are not able to cope with functions, but are restricted to a function-free horn logic respectively to attribute-value-lists.

Additional restrictions may be imposed by the search procedure. In the case of search of the rule space such restrictions may be imposed by meta predicates. A restriction of the form of the possible rules is often called *bias*. Because the bias of a system is basically of syntactical nature, it can hardly be described on the knowledge level. Therefore, it is virtually impossible to predict what hypotheses a system generates as a result of an example presentation, without knowing the exact details of the employed representation and algorithm. This is expressed in the following conjecture by Dietterich:

Conjecture 6.5.3.2 ([Die86, p. 300], Bias Conjecture)
None of the syntactic biases listed above[34] *can be captured as a single set of logical axioms that can be ascribed to a learning system as its background knowledge.*

This strong influence of the representation on the generated results can also be regarded as an advantage. By adequately changing the employed representation (e.g., by introducing additional attributes and values, so-called *constructive induction*) the learning capabilities of a system can be considerably improved. We will examine possibilities for changing a representation (language) in more detail in section 6.6.3.

Summary of AI Contributions to "Induction"

Two basically different starting situations for rule induction can be distinguished: in the first scenario a primordial regularity has already been generated (e.g., in an analogical transfer); in the second new regularities shall be found in a set of facts. The first task can also be regarded as an adaptation of the existing regularity (cf. section 6.6.4.1).

Two basic types of rules can be distinguished: strict and probabilistic rules. The usage of probability theory can either be restricted to the generation of rules or it can be extended to the application of the rules. If probability theory is only used for the generation of rules, then obviously some information gets lost.

Most approaches to the identification of regularities concentrate on implications among attributes. The identification of numerical relations has received considerably less attention. Consequently, numerical attributes are usually discretized for performing rule induction. We identified three different methods for doing this (cf. page 80).

Two basic approaches for finding rules are generally used: search in the instance space and search in the rule space. The approach used in the TDL-component of the OCCAM-system can be interpreted as a combination of both approaches, because it is able to generate rule models (causal patterns). This is an element of search in the rule space, although the system basically uses search in the instance space (as most systems do).

Most induction approaches to date use only little knowledge in the course of rule induction, although knowledge could be used in many different subtasks of rule induction. For example, it could be used for identifing appropriate attributes, a topic which has hardly been addressed in rule induction up to now. In systems that already use knowledge, it is usually present in the form of a taxononmy or as preexisting rules. This knowledge can be used in a multitude of ways. It

[34] At this point in the paper Dietterich had already given the following list of biases: Occam's razor, restricted language (only allow rules from a logically incomplete language), conjunctive descriptions, maximally general descriptions, maximally specific descriptions, least disjunction, and one disjunct per lesson.

can propose a way of generalizing a rule (taxonomy), it can reduce the search space by identifying unimportant subsets of the facts (IOU), inappropriate forms of rules (FOCL), or it can help to interpret the given facts in a different way (existing rules).

The induction of causal relations has received even less attention than the use of knowledge in induction. However, we regard this as a very important topic, as causal knowledge (or causal models) may help in a multitude of tasks in a creative problem solver (e.g., induction of new rules, analogical transfer, etc.). Consequently, this ability is probably very important for closing the *discovery cycle*.

Comparison of AI Contributions to "Induction" with Psychology:
The PUPS-model proposes that, if a transfer is necessary, first a generalized rule shall be generated which is then instantiated for forming a transfer. On the other hand, most AI approaches generate a generalized rule after the analogical transfer has been performed. This has the advantage that rule induction can take into account any modifications made during the transfer process. However, it does not allow the induction of a rule based on a single event as humans sometimes do. On the other hand explanation based generalization (EBG) is able to form generalizations based on a single experience.

When general rules shall be learned from examples (e.g., classification rules) then many people use the holistic strategy (cf. page 28). This approach is basically identical to *anti-unification*, an approach for generating classification rules, which is widely used in AI. On the other hand, the selective scanning strategy has no such counterpart in AI. This is due to the fact that AI methods refrain from restricting themselves to a subset of the available attributes if no substantiation for this exists. Here again (as in planning/theorem proving), we have a situation where people employ heuristics which simplify a task, but increase the probability of error, while AI tries to keep on the safe side. In psychological studies the relevant attributes are usually given to the subjects, therefore these studies do not provide much information for a strategy of attribute selection, too. An exception is the study performed by Wisniewski and Medin (cf. [WM91]). There the authors suggest that the selection of attributes is based on the theories people have. However, this topic is still neither in psychology nor in AI well explored.

Many approaches to rule induction are compatible with human behaviour: For example, experiments conducted by Qin and Simon ([QS90]) showed that the behaviour of the function finding system BACON is similar to the behaviour of the subjects in their experiments. According to Boden (cf. [Bod91]) the ID3-algorithm, while having some characteristics which are psychologically not plausible, is basically motivated by human behaviour.

With respect to probabilistic approaches we saw that humans make some systematic errors. Many of these errors can be explained by assuming that they try to imagine the relevant sets and estimate their relative sizes. Here problems in recalling the adequate information may account for many mistakes made. Other errors can be explained by the confusion of portions and probabilities, e.g., the probability that a dice shows 1 and the portion of all events in which the dice shows 1 (both may be radically different if the start of the sequence is already known). Because AI systems do not have problems recalling facts from memory and are often based on finely tuned probability models, one can conclude that AI methods are better able to cope with this type of problems than humans. At least up to now nobody could outline many advantages the heuristics employed by humans would confer over the application of probability theory.

Contrary to this, AI approaches to causal induction are scarce to date. If one compares the heuristics suggested for causal induction in the PUPS-model with the approaches outlined above, it becomes evident that no system uses the *identity heuristic*. The *heuristic of previous action* is extensively used in the OCCAM-system (in the TDL- and SDL-components). The DIDO-system is also based on this heuristic. The *principle of minimal contrast* is used in the LIVE-system if the *heuristic of previous action* gives no unambiguous result.

Most AI systems that use knowledge in induction do this in one of the following two ways (cf. [WM91]): Either they preprocess the available data using knowledge-intensive techniques and use the results for induction, or they perform induction on the available data and use the resulting

rules in knowledge-intensive techniques (like EBL). Contrary to this, when humans perform induction tasks their prior theories and available data tightly interact. Additionally, information about prior attempts of rule application are used. Up to now there does not exist an AI approach to rule induction which integrates knowledge and data that tightly.

This comparison shows that there is still much room for extensions and improvements of AI approaches. We want to conclude this section by saying that there are still some induction abilities which humans possess which have not been mastered by any AI system to date. Besides causal induction and the use of heuristics in induction, AI systems are still inferior to human abilities when it comes to the use of knowledge in induction. The latter may be due to the fact that AI systems to date do not possess much knowledge compared to a human.

Machine learning algorithms often show — similar to humans — a bias for some kind of rules, that is they only find certain types of rules. Humans usually show this effect in a restricted form: they are, in principle, able to learn any kind of rule, but they find some regularities much more easily than others. For example, Anderson (cf. [And89a]) mentions that people usually have considerably less problems learning conjunctive concepts than other types of regularities.

6.6 Restructuring of Knowledge

6.6.1 Introduction

In psychology the term "restructuring of knowledge" denotes a non-trivial transformation of the knowledge the subject has. A transformation can be a rearrangement of the aspects involved (e.g., reorder the importance of the design criteria) or it can be a change in the representation of a fact (e.g., "the glass is half-full instead of the glass is half-empty"). Additionally, the deduction or induction of knowledge is often regarded as restructuring in psychological literature. We dealt with this topic already in the previous section, because we think it is more adequate to distinguish between the generation and the transformation of knowledge. Therefore, we will use the term "restructuring" only for the rearrangement of knowledge and for changes to the representation of facts. A special form of rearrangement is the change of the indexing structure of the knowledge, because it does not change the knowledge itself but only its accessibility. Therefore, we can distinguish the following three categories of restructuring:

I. changes to the accessibility of knowledge
   reindexing, forgetting[35]

II. restructuring on the syntactic level
   reformulation, concept formation, constructive induction

III. restructuring on the semantic level
   adaptation, addition/deletion/modification of constraints (subgoals)

Alternatively, one could distinguish the different operators according to the type of knowledge on which they operate:

i. restructuring of background knowledge
   reindexing, (forgetting,) concept formation, constructive induction, reformulation, adaptation

ii. restructuring of problem solving ideas
   reformulation, adaptation

[35] We dealt with this operator already in section 6.4.3, because it has strong connections to the activation of knowledge. However, it would be equally well justified to discuss this operator here.

iii. restructuring of problem specifications
reformulation, addition/deletion/modification of constraints (subgoals)

As this list shows, some operators can be applied to different types of knowledge. Therefore, we will use the first categorization in this section. An interesting aspect of this distinction is that it emphasizes the importance representational changes have to creativity. This is due to the fact that the success of reasoning in humans as well as in AI systems is strongly influenced by the form in which the available knowledge is represented. In section 6.6.3 we will examine the effects the representation has on the reasoning capabilities in more detail.

6.6.2 Changes to the Accessibility of Knowledge

As we have seen in the previous sections the ability to retrieve the right knowledge at the right time is of central importance to the success of the creative process. Among others the following abilities contribute to this: association, forgetting, reindexing. Of these operations association and forgetting have already been described (see section 6.4), therefore we will now turn to reindexing. Reindexing can be regarded as restructuring, because it makes previously inaccessible knowledge accessible and changes the order of the retrieved knowledge items.

6.6.2.1 Reindexing

As it is usually intractable to perform an exhaustive search of memory, it is important that a well-chosen set of indices for retrieval of information from memory is available. This set may change over time depending on the tasks the problem solver shall solve.

Operator: Reindexing

Scope: The reindexing-operator is necessary to ensure that the memory is properly organized and indices for retrieving important information are available at any instant.

This operator only changes the organization of the available knowledge. It does not generate additional knowledge, nor does it delete any knowledge.

Precondition:

- Additional information shall be incorporated into memory.
- The current organization has proven inadequate.

Purpose of call: Improve the accessibility of the knowledge.

Input: (Information that shall be incorporated into memory.)

Knowledge used:

- Information about the success of previous retrievals.
- Statistics about the helpfulness of previously retrieved information.
- Statistics about the current organizational structure of the memory.

Operators used: —

Knowledge generated: (New indices may be generated.)

Tasks generated: —

Contributions from AI:

The question, which indices shall be used for organizing memory and in which hierarchical order cannot be answered in full generality, as it is usually admitted that the usefulness of an index strongly depends upon the performance task (cf. [SHJ+94]). However, according to Kolodner (cf. [KP90, p. 981]), a negative characterization is possible:

- *A priori* or anticipatory indexing is insufficient, i.e., new indices must be generated during run-time.
- The primary goals are insufficient as a base for generating indices.

Often the features used as indices are partitioned into two categories based on the amount of processing needed for generating them:

Surface features: Features, that are obvious from the problem description, e.g., inclined plane.

Deep features: Features that need some amount of inference, e.g., an energy-conservation-problem.

Experts seem to rely more on deep features, while novices base their judgements on surface features (cf. [ES90, Sch93]).

In some studies the view is expressed that a well-organized memory is based on *predictive features*, however, what is required from a feature in order to be predictive varies from author to author. For example, Seifert et al. (cf. [SHJ+94]) require that a predictive feature predicts potential problems and can be used for identifying relevant plans in memory. Consequently, the predictiveness of a feature depends on the domain, the task, and the type of processing used. This dependency is mediated by the causal structure of the domain. However, because this structure is not explicitly available, Seifert et al. use in their RUNNER-system a heuristical approach for monitoring the predictiveness of an index.

Elio and Scharf (cf. [ES90]) use a simpler definition of *predictive* in their study: a *predictive feature* is a feature that frequently co-occurs with other features. They implemented their ideas in the EUREKA-system. This system solves physics word problems using a means-end-problem-solver. Additionally, problem solutions are stored in a P-MOP-network (MOP=**M**emory **O**rganization **P**acket). This network is hierarchically organized: parts that are common to several problem solutions are organized by one P-MOP, while the dissimilar parts are stored on sub-P-MOPs, which are sons of the more general P-MOP. As a problem solution may have different parts in common with different previous solutions, a problem solution may contribute to many P-MOPs. If the system searches for the next step, it first tries to retrieve an adequate step (i.e., an equation) from its P-MOP-network, by using the currently known facts about the problem as an index. Only if this fails, the means-end-problem-solver is used. When searching for information in the P-MOP-network, only predictive indices are used for retrieval (cf. [ES90, p. 600]).

This approach to indexing can also be regarded as *difference-based indexing* (cf. [Kol93]), because only those features are used as indices, which distinguish between different problem-solutions.

Two basic approaches to *reindexing* can be distinguished: failure-driven and success-driven reindexing.

Success-driven reindexing. This approach is used in the EUREKA-system. Each successful problem solving attempt results in a problem solving trace. This is then integrated into EUREKA's P-MOP-network. A P-MOP is splitted if an incoming problem solution has some commonalities with it, but also differences. The commonalities remain on the P-MOP, while the differences are organized by new sub-P-MOPs. Additionally, whenever a sub-P-MOP organizes the majority of the problem solutions organized by its parent P-MOP, it is integrated with its parent. The indices that lead from a P-MOP to the sub-P-MOPs are formed by the differences of the two P-MOPs. For example, if the sub-P-MOP organizes only that subset of the problem solutions organized by its parent, for which holds `desired(body1,accel-x-axis)`, then this feature can be used as an index. By the iteration of the splitting of P-MOPs and reintegration, deeper features move slowly to the top of the P-MOP-network (cf. [ES90, p. 598]).

Failure-driven reindexing. Other systems reorganize their memory only when a failure indicates, that the current memory organization is insufficient. An example of this approach is the RUNNER-system. Here, the indices should predict problems and opportunities for plan-execution. If this is not correctly made, a reorganization is necessary. The RUNNER-system uses so-called *appropriateness*

conditions for predicting when plan execution is possible and appropriate. For our purposes we can identify *appropriateness conditions* and indices. When reorganizing its memory this system checks and handles the following three conditions (cf. [SHJ+94, p. 48])[36]:

- An appropriateness condition that turns out to be always true is dropped.
- If a plan fails because some sub-plan of it fails, and that sub-plan failed because some appropriateness condition did not hold, then *promote* that condition to the status of an appropriateness condition for the super-ordinate plan.
- If a plan is frequently found to have false appropriateness conditions in situations where the plan is needed, and the conditions are under the agent's control, consider including the conditions in an *enforcement plan* that maintains them, so that the conditions can then be assumed true for that plan.

That is, the memory structure is improved by monitoring the usefulness of the current indices and taking corrective action (e.g., promoting an index).

A similar approach is used in the META-AQUA-system (cf. [Cox94]). This is a system for story-understanding. If the system stores a completely understood story it may find some information it earlier failed to retrieve. In such a situation the system indexes this information such that in the future it will be retrieved in similar situations. This is done by forming an appropriate generalization of the information to be stored and the information found.

An approach which can be used for success-driven as well as for failure-driven indexing is *explanation-based indexing* (cf. [Kol93]). In this approach an explanation for a failure or for a solution is generated and generalized as far as possible while preserving correctness. The features, which occur in this generalized explanation are then used as indices.

Comparison with Psychology:
The reindexing of information is a common activity of humans, too. For example, Selfridge (cf. [KW93]) claims that people tell stories to reindex them under new generalizations that have been learned since the story was first acquired.

Weisberg holds the view (cf. [Wei89]) that becoming competent in a field is necessary for being creative in this field. The preparation stage is presumably very important for this "becoming competent". An important accompaniment of this development of skills is that the available information becomes organized in a principled, useful way (cf. [ES90, Sch93]). Especially, experts show a memory organization that is more based on deep features than that of novices.

A possible model for explaining how a memory can become organized by deep features, which also tries to be psychologically sound, is given by Elio and Scharf with the EUREKA-system (cf. [ES90]). Seifert et al. ([SHJ+94]) also give a model of memory organization that is supported by psychological experiments. They emphasize the predictiveness of the features that are used as indices.

In section 3.6.6 we expressed the view that the *reminiscence phenomenon* can be explained by indexing with respect to existing indices during breaks and by generating new indices. This explanation is supported by the afore-mentioned evidence. However, there are two important differences of computer programs to human behaviour:

- First, computer programs usually do the reindexing immediately. We believe that this will be no longer possible, if the knowledge bases will become very large.
- Second, up to now, we do not know of any system which combines reindexing with the ability to form new concepts and regularities.

We think that, if this two inabilities of computer programs will be repaired, AI systems will probably show *reminiscence phenomenons* just as humans do.

[36] More exactly, at present the system handles only the first two conditions, while plans exist to extend the system for handling the third condition.

6.6.3 Restructuring on the Syntactic Level

By *restructuring on the syntactic level* we refer to a change of the representation of the problem solver's knowledge, that does not include a change of the semantic content of the knowledge base (cf. section 6.6.1).[37] This can be done by changing the representation of some facts within the current representation language. However, this is only possible, if the current representation language allows for multiple, semantically equivalent formulations of a fact. If this is not possible, then the representation language must be extended first.

Haase (cf. [RP93, p.49]) regards such an extension as a reconfiguration of the search space. If such a reconfiguration is made ingeniously, then the resulting search space can be searched more efficiently by the available problem solving techniques. Consequently, a modification of the representation language can be of decisive importance for the success of a problem solving attempt. This becomes especially obvious in the context of rule-induction. Here, it has been shown that the introduction of additional attributes/concepts can considerably improve the quality of the regularities found (cf. [RS90, WM94, Wro94]).

For *automated discovery*-systems it is very important to reorganize the knowledge base once in a while (e.g., by introducing additional representational elements like concepts), as this may conserve (or improve) the structure of the knowledge base and thereby simplifies other tasks on the knowledge base.

Haase (cf. [RP93]) points out, that one important reason for the success of AM was, that it had (and preserved) a *tight* representation scheme, i.e., a syntactic change often resulted in an interesting semantic change. A tight representation can only be conserved by an adequate extension of the representation language during the discovery process. Such an extension should introduce concepts that are (basically) indistinguishable from older concepts.

Due to these reasons, modifications to the representation play an important role in Lenat's discovery model (cf. level 1, 3, and 4 in figure 5.1).

In the following two sections we will first deal with the change of the representation within an existing representation language, then we will turn to modifications of the language.

6.6.3.1 Reformulation

Operator: Reformulation

Scope: The purpose of this operator is to state existing knowledge in a new form, as permitted by the existing representation language. In section 6.6.3.2 we will deal with methods for modifying the representation language.

This operator has some similarities with the *reindexing*-operator, which makes existing knowledge accessible in new ways.

Precondition: Problem solving has reached an impasse or an extension of the representation language has occurred.

Purpose of call: Change the representation of a knowledge item. This may be the specification of a goal of the problem solving process, any intermediate results, etc.

Input: Existing knowledge item and domain knowledge.

Knowledge used: Background knowledge about different representations.

Operators used: —

Knowledge generated: —[38]

Tasks generated: (Modify the representation language.)

[37] As we will see, we can not always strictly adhere to this principle. However, we will use it as our main guideline.

[38] This operator does not generate any knowledge, but changes the representation of existing knowledge.

Contributions from AI:

One possibility for restructuring is reformulation. An example has already been shown in our match problem (cf. page 5). Here, the reformulation of the problem itself, based on the winners of the matches instead of the losers, was of decisive importance for finding the best solution.

Due to the change in the knowledge base that happens, a reformulation step is similar to an inference step. However, a reformulation step can be more regarded as some kind of meta-step: it is akin to a jump within the search space[39] or to moving the problem around within the search space.

Often the opportunity for a reformulation of a problem situation only arises if previously some defaults are redrawn. Consequently, one can regard Truth Maintenance Systems (TMS) and Backtracking as aids to reformulation. However, as these techniques basically are adaptation techniques, we will discuss them in section 6.6.4.1.

A different type of reformulation step, which more literally corresponds to our definition, is used in rewriting (cf. [Ave91]). Rewriting systems are usually based on *completion*. The aim of these techniques is to prove theorems involving equality. The basic idea of this proof technique is the replacement of equal by equals. This is done in so-called *rewriting steps*. These can be regarded as prototypical instances of reformulation steps, as not the knowledge but only its form is changed. Additionally, an ordering is used for ensuring that the rewriting (reformulation) steps are helpful. This ordering can be regarded as meta-knowledge about the structure of the search space. Rewriting systems are also able to perform inference steps. This is done using a technique called *critical pair generation*.

Modifications of the representation language (cf. section 6.6.3.2) lead to opportunities for reformulation in a natural way: as soon as a new representational element (e.g., an attribute, a concept) is introduced into the representation language, the existing knowledge items can be reformulated in a way that makes use of these elements. If a new attribute is introduced into the representation, then, usually, the corresponding values for all items are assigned immediately. This is possible, because the value-assignment is straightforward as the attributes are usually defined using other attributes. The indexing of existing knowledge under newly introduced concepts has already been discussed in section 6.6.2.1.

After the representation language has been extended an important opportunity for reformulation, besides those already discussed, arises. This is the reformulation of previously found regularities. Here, the change of the representation language also changes the biases and therefore gives rise to the possibility of finding more adequate regularities. An example for this is the MOBAL-system (cf. [Wro94]). After introducing a new concept the system tries to characterize it, by finding regularities it is involved in. Often, these new rules subsume existing rules and express the knowledge contained in them in a more adequate way. The fact that the regularities thus generated are more adequate for describing the knowledge becomes especially obvious by the fact, that they have smaller exception sets than the original ones (prior to the generation of the concepts).

Comparison with Psychology:

In the context of changes to the representation language a basic difference between human behaviour and the behaviour of computer programs is obvious: while computer programs try to exploit any reformulation opportunity that arises immediately, humans do this only as need arises[40]. We believe that this is due to the fact that these programs work with much smaller knowledge bases than humans do. Consequently, if AI systems shall scale up they need to restrict themselves to the reformulation necessary at any given instant.

If one compares, what has been said above, with the reformulation operators mentioned in section 3.6.5, then it becomes obvious that operation 1 (modification of the accentuation) is partly covered by the induction of regularities. Operation 4 (change in the perceived function of parts) seems to be very important in the context of design. However, AI systems usually have no com-

[39] This emphasizes the close relation to the modification of the representation language, which can be regarded as a reconfiguration of the search space (cf. [RP93]).

[40] Or when they get the opportunity to do so. This may be important during the incubation stage.

parable operation. This is due to the fact, that they either employ a representation that allows for the description of functional, behavioural, and structural aspects (cf. page 71) or they have no possibility of representing multiple functions. In the first case different functions are equally well supported and the system does not need such an operation, in the second it is not possible.

However, this behaviour is incompatible to the behaviour shown by humans, as in any context they prefer a specific function of an artifact. This behaviour also seems more adequate, because usually in a given context only one function is of importance. This focus leads to a considerable reduction of the search space.

Operations 2 and 3 (combination of several parts into a new whole and disintegration of a whole into parts) seem to be especially important with respect to problem solving techniques based on abstraction. However, we do not know of a system which explicitly possesses comparable operations.

Reformulation can also happen, by transferring the problem into a different domain, where more knowledge is available. An example of this is the reformulation of the multilated checkers problem as the match-making problem (cf. page 26).

The relatively low interest reformulation has received in AI up to now seems to be due to two reasons:

1. AI systems are usually not able to support multiple alternative representations of their knowledge.

2. Systems which are able to extend their representation language are usually only capable of performing limited and well defined problem solving tasks.

We believe that the importance of this operation becomes more widely accepted, when these restrictions are alleviated.

6.6.3.2 Modification of the Representation Language

The topic of this section are modifications of the representation language, which are based on experiences of the problem solver/learner.

These modifications (e.g., concept formation) are very important with respect to the creative process, as they supply a basis for reformulation and can facilitate other important tasks of the creative process (e.g., induction).

In this section we will describe several possible changes of the representation language like concept formation, the creation of additional attributes (slots) and values (so-called *constructive induction*), and the elimination of these representational elements. We will discuss all these modifications of the representation language in the same section, because the basic ideas underlying these changes are similar.

Operator: Modification of the representation language

Scope: We will describe the formation of new concepts, attributes, and values based on existing ones. This corresponds to level 1 (define domain concepts) and level 3 (augment representation) in figure 5.1. We will not discuss level 4, as we do not know of an AI system, which is able to introduce new representations. Additionally, we will study the elimination of representational units, thereby discussing in more detail some issues which we mentioned already in section 6.4.3 (forgetting).

Precondition:

- The current representation seems no longer adequate.
- A subset of the knowledge elements shall be subject to closer examination.
- Other induction mechanisms are not successful enough.
- (Elimination) A concept/attribute did not prove useful during problem solving.

Purpose of call: Try to adapt the representation better to the current problem solving task.

Input:

- Examples which shall be clustered into a concept (hierarchy).
- An existing concept (hierarchy) together with examples or other hints which point to a useful modification.

Knowledge used: Information about the usefulness of the various concepts/attributes.

Operators used: —

Knowledge generated:

- Descriptions of the new representational elements.
- (Information about the classification of existing knowledge elements according to the new representational elements.)

Tasks generated: (Index existing information with respect to the new knowledge elements.)

Contributions from AI:

In the following discussion, whenever we refer to *concept formation* we will mean the introduction of a concept or an equivalent change of the representation language. For example, the AQ17-HCI-algorithm (cf. [WM94]) can introduce additional attributes into its attribute-value-list representation, which are defined by formulas which are composed from other attributes and may only have one of two values $(0, 1)$. This can also be interpreted as each attribute-value-list (entity), which has a value of 1 for this new attribute, belonging to a concept defined by the expression defining the attribute, while those having a 0 do not belong to the concept.

Wnek and Michalski also allow the assignment of a real value from the interval $[0, 1]$ to the attribute based on a similarity measure. This can be interpreted as the proportion to which this entity belongs to the new concept. Often an example is classified as being more or less typical for a concept (cf. [HY89]).[41] This shows that although attribute-value-lists have no special representation capability for representing concepts, they can nevertheless be represented.

However, most studies which combine attribute-value-lists and concept formation use a different approach: conceptual clustering. That is they construct a hierarchy of clusters; the clusters on one level of the hierarchy form a partition of one cluster on the level above. In this hierarchy additional information is stored about the entities belonging to each cluster and about the frequency with which each attribute-value-pair occurs in a cluster.

On the other hand, logic- or frame-based representations are able to represent concepts within their initial representation capabilities. Nevertheless, they can benefit a lot from the introduction of concepts (cf. [Wro94]).

Two different activities need to be distinguished during concept formation (cf. [Fis87, p. 142]):

Aggregation: Grouping together different knowledge items, which shall form a new concept (or attribute, etc.). This is also called an *extensional definition* of the concept.

Characterization: Give a structural description of the concept, i.e., give a set of rules which can be applied to any knowledge item for deciding whether the item belongs to the concept. This is also called an *intensional definition* of the concept.

Because the characterization of a concept essentially is *learning from examples*, it can be performed using rule induction as described in section 6.5.3. Consequently, we will concentrate in this section on the clustering step. However, some approaches tightly interconnect both activities (e.g., conceptual clustering).

[41] This can be regarded as an implementation of the human behaviour to regard some examples as being more prototypical for a concept than others. Michalski gives a different approach to the implementation of concepts with prototypical examples in [Mic90].

When we speak of the elimination of a concept, attribute, or value, then this can be regarded as forgetting in the sense of section 6.4.3. That is, the representational element need not be deleted from the knowledge base as a whole, but only from the active knowledge.[42]

Although many different approaches to the modification of the representation language exist, their motivations can be distinguished into two main categories:

1. A better organized representation of the available knowledge is desired.
2. Modifications of the representation are expected to have a positive effect on the performance in the current task.

6.6.3.2.1 Improving the Organization of Knowledge

In this case usually a concept hierarchy is desired which organizes the various knowledge items in the knowledge base.[43] It can be shown (cf. [Wro94]) that the introduction of new concepts together with a reformulation of the knowledge in the knowledge base improves the quality of a knowledge base (in the form of function-less horn-logic rules) if the introduced concepts adhere to the following criterion:

- At least two sufficient conditions for the concept exist, i.e., at least two rules exist where the predicate c, which denotes the new concept, occurs on the right hand side.
- Either c occurs in at least two rules as the only predicate on the left hand side
- Or there exists a rule, in which c occurs together with a different predicate in the left hand side.

Most approaches to concept formation (especially conceptual clustering) are based on grouping similar objects together, i.e., they try to increase intra-cluster similarity and decrease inter-cluster similarity (e.g., COBWEB, [Fis87]; UNIMEM, [Leb87]; ARACHNE, [ML91]; INC, [HY89]). Naturally, this results in concepts which can only be probabilistically characterized. Consequently, it is often difficult to interpret the resulting clusters in a meaningful way.[44]

Systems like COBWEB or INC use a top-down approach to the construction of the concept tree, i.e., whenever an element e needs to be introduced into the tree, they start at the root and evaluate the results of the following five activities[45] using their quality measure (*category utility* in COBWEB):

- add e to an existing class
- create a new class (only containing e)
- combine two classes into a single class
- divide a class into several classes
- promote a class (to a higher level)

The action, which scores the best according to the quality measure is then accepted. Therefore, these approaches can be regarded as hill-climbing in the space of concept hierarchies. Some systems (e.g., ARACHNE) impose additional restrictions besides the quality measure on the horizontal and vertical placement of concepts within the concept hierarchy. According to [ML91] this makes ARACHNE less sensitive to the ordering of examples.

[42] Sometimes a representational element is only obstructing a single subtask of the problem solving task (e.g., the induction of a regularity). In this case *elimination* refers to utilization filtering, i.e., this element is not used during this subtask, but will remain in the active knowledge.

[43] The vast majority of approaches to concept formation (especially conceptual clustering) work incrementally, i.e., a new object is integrated into the concept hierarchy as soon as it turns up.

[44] According to McKusick and Langley (cf. [ML91]) this is especially true of COBWEB. They claim that their system (ARACHNE) improves upon this situation.

[45] Many systems only use the first four activities.

However, there also exist some approaches which are based on the structural descriptions of the objects (e.g., CLUSTER). Consequently, these concepts can be more easily characterized in an understandable way.

The approaches mentioned above are usually based on the assumption that the underlying representation is a simple attribute-value-list. However, Thompson and Langley ([TL91]) did show that these approaches can be extended to structured objects, too.

Due to the emphasize COBWEB-like approaches give to a high correlation of the features these approaches can also be used for supporting a task: rule induction. Indeed, clustering approaches are often used in this way. Suppose some values are missing from a object, but enough remain, such that a classification of the object as belonging to a certain cluster can be made, then the missing values can be expected to be identical to the typical values of the cluster. In this interpretation clustering can also be interpreted as the generation of probabilistic rules. Consequently, the problem of overfitting can arise just as with rule induction techniques (cf. page 82). Therefore, Fisher suggests to prune the cluster hierarchy (cf. [Fis89]).

6.6.3.2.2 Supporting a Performance Task

Just as a concept hierarchy, which is organized based on feature correlation, can be used as a set of probabilistic rules, opportunities for a change of the representation arise in the context of rule induction. Indeed, the term *constructive induction* usually refers to the generation of additional attributes in this context.

In AQ17-HCI (cf. [WM94]) a new attribute is only introduced, if it is able to distinguish better between positive and negative examples than any of its components. This is quantified by a measure called *pattern strength*. Correspondingly, an attribute which occurs in none of the generated rules or only occurs in rules which have low strength is removed from the representation.

AQ17-HCI is able to recognize three different types of patterns in a preliminary rule set:

Value pattern: Values that often co-occur in a class description, e.g., `blue` $\vee$ `red` $\vee$ `white`.

Condition pattern: A conjunction of two or more elementary conditions that frequently occur in a rule set for a given class[46], `(x=20) and (y=large)`.

Rule pattern: A rule pattern is a rule or a subset of the rules that characterize a given class, e.g., `[(x=20) and (y=large)] or (x=7)`.

These patterns are used for extending the representation. While value patterns introduce new values for an attribute, condition and rule patterns define new attributes.

In this context the induction of new attributes helps to simplify the rules. Thereby, it makes it possible to learn more complex relationships with the same effort. This enables the induction algorithm employed to learn rules which previously would have been to complicated. Wrobel calls this "moving the learnability cliff".

This behaviour is similar to the BR-3-system (cf. page 61), which introduces new quantum numbers (which can be regarded as attributes of a particle), if this is necessary for explaining the observed reactions. An additional problem of the BR-3-system is that the attribute values for the objects are not given by the attribute definition, but must be found by the system. Similarly, some members of the BACON-family of programs are able to introduce material constants into the knowledge base for explaining observations (cf. [Haa89]).

The DIDO-system (cf. page 61) learns reactions of the environment to its actions. These experiences are stored in a classification hierarchy. Here, a class (i.e., a concept) is defined only by those predicates that characterize a situation. A new class is introduced as soon as a new situation is observed and a class is eliminated, if the results of the actions are not different from the results observed in its superclass. Additionally, an outcome of an action may be eliminated, if its probability falls below a certain threshold. The OCCAM-system (cf. page 83), which has a similar performance

[46] Remember, that AQ17-HCI's performance task is the characterization of pre-classified examples.

task, uses a different approach. Using an UNIMEM-like approach it forms clusters of pairs of situations. From each of these clusters it induces a causal pattern.

An approach in which those knowledge items that support the same reasoning are grouped together is illustrated by the BLIP- and MOBAL-systems (cf. [Wro89, Wro94]). There, the user can notify the system about exceptions to the applicability of rules. If the set of exceptions becomes to large, the system tries to introduce a new concept that describes the range of applicability of the rule more adequately. However, Wrobel suggests that the concept should only be introduced into the representation of the system if it complies with the criterion mentioned on page 95 and it should be removed as soon as this condition no longer holds.[47]

Contrary to the conceptual clustering approaches discussed in the preceding section the concepts thus generated are not part of a concept hierarchy. The only connections between concepts are given by the rules which hold among the concepts. Similarly, in other approaches attributes and newly introduced values are usually not integrated in a concept hierarchy. A notable exception from this rule are the slots in EURISKO. Here, the slots themselves are represented as concepts in the hierarchy.

An important aspect of concept formation in the context of problem solving is the concentration on those features, that are relevant to the current task. Such a recognition of the relevant features can be performed using **E**xplanation **B**ased **L**earning (EBL). Consequently, Mooney (cf. [Moo91]) interprets EBL as concept formation.

A second important aspect of concept formation in the context of problem solving is the usage of deep features, i.e., features which are not part of the problem description, but can be generated from it. Yoo and Fisher (cf. [YF91]) tackle this problem with a system called EXOR. This system forms a tree of partial problem solving traces. Each node contains a trace that is more specialized (i.e., contains additional rules) than its parent. All predicates of a problem solving situation (including the inferred ones) are used for predicting the usefulness of a partial problem solution.

6.6.3.2.3 Concept Formation as a Performance Task

While most of the approaches described above are based on finding a group of examples and then try to find an intensional description of this set (or interleave both activities), sometimes the starting point is an intensional description of a concept.

The most prominent examples for this are the AM and EURISKO-programs. These programs start by generating intensional descriptions of concepts and then look for examples for these. Based on these examples they induce additional properties of the concepts.

Typical motivations for the AM-program for introducing a new concept are:

- Only few examples can be found. Define a generalization of the concept (cf. [Len82a, p. 171, Heuristic 39]).
- A lot of examples can be found. Define a specialization of the concept (cf. [Len82a, p. 171, Heuristic 40]).
- Define concepts that are related to interesting concepts. Typical relations used are: generalization, specialization, symmetry (define the complement), intersection (define the intersection of two interesting concepts), etc.

EURISKO was also able to introduce new slots. However, its motivations for doing this are basically identical to those for introducing new concepts:

- A slot contains too many values.
- A slot has become very important (i.e., it has high worth).

For defining new slots basically two methods exist in EURISKO (cf. [Len83b, p. 278]):

[47] He notes, that the quality of the rules generated in future rule induction tasks is improved, if this criterion is respected.

- Specific information about the contents of a slot is extracted and stored in a new slot. For example, given a slot `extreme-examples` EURISKO defined (among other slots) `average-worth-of-extreme-examples`.

- Based on *two* slots a, b and a relation R, a new slot s is defined, having the semantic $k_2 \in s(k_1)$[48] iff $R(a(k_1), b(k_2))$.
  For example, if k_1, k_2 are heuristics, R is the relation *implies*, a denotes the slot `then-conjecture` and b denotes `if-truly-relevant`, then the new slot s has the semantics `can-trigger`, i.e., after execution of heuristic k_1, k_2 is applicable.

Summary of "Modifications of the Representation Language"

Conceptual clustering systems usually try to optimize a quality measure based on the distribution of examples within the concept hierarchy (e.g., *category utility* in COBWEB). Basically, this approach is independent of a performance task, however, such a coupling may be introduced. Then the task can contribute additional features used for clustering (cf. [YF91]) or can identify some features as being unimportant (cf. [Moo91]).

However, when modifying the representation in the presence of a performance task often the measure is derived from the performance task itself (e.g., *pattern strength* in [WM94], probability of the reactions in [SM93], etc.).

This measure is usually coupled with a certain (more or less arbitrarily chosen) threshold determining whether an additional representation element shall be introduced or dropped. However, sometimes an independent measure is used for recognizing the need for an extension of the representation (e.g., in MOBAL the number of exceptions of a rule, in the BR-3-system an impasse in explaining observed reactions is reached, etc.).

A knowledge-intensive approach to concept formation is taken in the programs AM and EURISKO.[49] These programs generate a task for introducing a new concept (or a new slot) whenever the opportunity arises. However, the task is executed only if it is regarded as being interesting enough. This estimate of the *interestingness* of a concept/slot is done by heuristics. Many of them are domain-specific. Additionally, these programs monitor the interestingness of their concepts continuously. This is done by similar heuristics. Contrary to other approaches, EURISKO was not only able to use knowledge (heuristics) for generating new concepts, but also for generating new attributes (slots). Most systems, which are able to introduce new attributes, only use empirical techniques.

Comparison of AI Contributions to "Modifications of the Representation Language" with Psychology:

Many approaches to concept formation are inspired by psychological studies. This is especially true of approaches to *conceptual clustering* (cf. COBWEB, [Fis87]; INC, [HY89], etc.).

In section 3.6.4 we distinguished two different forms of *macro formation*, which are interesting with respect to concept formation:

1. Formation of a class from elements with common properties, thereby neglecting unimportant details.

2. Encoding of complex internal models (e.g., car instead of: motor, four wheels, ...).

All approaches described generate concepts of the first form. In this case the decision, what is an unimportant detail depends on the future use of the concepts. In most AI systems the relevance of the attributes is a-priori given. The other systems use one of two approaches:

- The importance of an attribute may be determined empirically by measuring its usefulness. This approach is taken by most systems that employ *constructive induction* (e.g., [WM94], [RS90]).

[48] $s(k_1)$ denotes the list of entries on the slot s of concept k_1.

[49] According to Lenat (cf. [Len82a, p. 58]), AM possessed 238 heuristics. Of these, 59 heuristics were used for evaluating the interestingness of a concept. We do not have any numbers regarding EURISKO, but expect that here the number of heuristics was much higher.

- The relevant attributes can also be identified using a background theory (cf. [Moo93]).

Based on psychological experiments Wisniewski and Medin ([WM91]) suggest that in humans both techniques strongly interact.

The second form of *macro formation* has received considerably less attention in AI. It is distinguished from the first type by the fact that here an abstraction is performed. This may be regarded as a reversal of EBL. However, it remains unclear what makes humans perform such abstractions. It seems to be a plausible hypothesis, that the abstraction is triggered by the function (i.e., the role in the reasoning) that this set of components play. From this point of view one could regard BLIP and MOBAL (cf. [Wro89, Wro94]) as approximations of this behaviour. The abstraction can then be regarded as a transition from a structural to a functional description.[50]

It is commonly accepted that creative people possess a high tolerance with respect to diverging details (cf. section 2.3.2). Approaches to concept formation, which are based on probabilistic measures (e.g., conceptual clustering), also have this property. In the concepts generated by these approaches there often does not exist a single attribute, which has the same value for all elements. Contrary to this are approaches, which are based on structural commonalities. Typically, in these approaches no element may belong to a concept, which diverges in a single attribute from the description.

According to Rhine (cf. section 3.6.4) attitudes can also be regarded as concepts. These are second order concepts, which result from combining first order concepts, where one of the concepts includes a judgement (e.g., unpleasant). Attitudes determine what someone is striving for, or what someone wants to avoid. From this point of view attitudes can be regarded as goals and — consequently — are very important for control. However, up to now no AI system did use concepts in this form. EURISKO can be regarded as an approximation of this view, because its control was based on heuristics and such a heuristic can be regarded as a combination of an attitude (goal) and an action (means to achieve the goal). Additionally, this system was able to generate new heuristics (attitudes) and to arrange them in a specialization/generalization hierarchy.

Especially, in the field of *creative design* it would be very important for a system to be able of generating new goals (evaluation criteria), when an opportunity arises. However, up to now existing systems are only able to modify their goals along predetermined dimensions (cf. section 6.6.4.3).

6.6.4 Restructuring on the Semantic Level

In this section we will deal with restructuring operations that change the content of the knowledge base. While all the operations described in this chapter have some importance to the creative process, the operations described in this section are of central importance to creativity. This view is strongly supported by psychological literature that expresses the opinion that restructuring is central to the creative process (e.g., [SK74, Ulm68, Wei89]). Here, restructurings are often described as small steps that open the way for new possibilities of solving a problem. Based on the different types of knowledge that may be subject to adaptation three different forms can be distinguished:

1. Adaptation of background knowledge
2. Adaptation of partial problem solutions
3. Modification of the problem specification

We will distinguish between different kinds of adaptations employed, because the technique employed depends upon the knowledge that must be changed. This distinction has not been necessary in the psychological part of this study (see section 3.6.3), because in psychological literature the only type of adaptation that is well covered is the adaptation of general rules.

[50] Note, that this is not a behavioural description (cf. page 71), because only *one* of the possible functions that these components may serve is used.

6.6.4.1 Adaptation of Background Knowledge

In most systems the adaptation of knowledge is restricted to the adaptation of general rules. Therefore, this will also be central to the discussion presented here. This special topic is also well explored in psychological research (cf. section 3.6.3).

Operator: Adaptation of the knowledge base

Scope: Methods for adapting rules to contradictory evidence are presented.

Precondition: A fact is not consistent with the current state of the knowledge base.

Purpose of call: Integrate the fact into the knowledge base.

Input: a new fact

Knowledge used: Existing rules and facts in the knowledge base. (Information about the origins and reliability of this knowledge.)

Operators used: —

Knowledge generated:

- Modifications of existing rules and facts in the knowledge base.
- Integration of the new item (possibly in a modified form).

Tasks generated: —

Contributions from AI:

In the context of deductive databases two different reasons for an adaptation of the content of the knowledge base are distinguished (cf. [Som94]): *updating* and *belief revision*. While *updating* refers to a change of the knowledge base due to a change in the world, *belief revision* denotes a change due to a previous error in the knowledge base. However, as most approaches to adaptation do not make this distinction we will not make it here, too. The motivation for a problem solver to perform these adaptations is that in order to arrive at an adequate problem solution the knowledge base used should reflect the state of the world as correct as possible.

Belief revision and *updating* are similar to *non-monotonic reasoning* insofar as all these techniques often require that previously made inferences must be retracted in order to ensure consistency of the knowledge base. In *non-monotonic reasoning* the retraction of previously made inferences is usually due to the use of default assumptions in the reasoning chain, which do no longer hold. An important tool for managing the necessary retractions are **T**ruth **M**aintenance **S**ystems (TMS). Tightly connected to TMS is the problem solving approach of making tentative assumptions and retracting them if they do not lead to a solution. This technique is called *backtracking*. As the withdrawal of an assumption is often needed for finding the right adaptation, *backtracking* and *TM-systems* can be regarded as important tools for making adaptations.

Contrary to the approaches described above, many systems do not require that the resulting knowledge base is consistent or this is a trivial corollary to the consistency of the single regularities with the ground facts. Consequently, we will concentrate on the adaptation of single rules to new facts.

As we have already seen in the *selective-scanning-strategy* (cf. table 3.4) taking into consideration an additional knowledge item may lead to one of the following two types of adaptations:

Generalization: The conclusion of the rule holds, but not the premise.

Specialization: The premise of the rule holds, but not its conclusion.

Specializing a rule is often called *discrimination*. If a condition is added to the rule then it is called a *discriminating condition*.

While most systems generalize/specialize only those rules which are contradicted by a fact, some systems impose additional restrictions which may lead to the adaptation of further rules and facts. For example, in deductive databases the consistency of the database after the adaptation is usually required, or when generalizing classification rules it is sometimes required that the adapted rules still allow for a unique classification of each example.

The latter restriction is made in algorithms like ID3 (cf. [RW91]).[51] These adhere to this restriction by successively partitioning the example space. (One can regard this as specializing a rule and introducing additional rules that cover the remaining examples.)

A different approach for adhering to the same restriction is the *complementary discrimination technique* developed by Shen ([She93]). In this approach a single rule is split into two siblings,[52] as soon as it makes a wrong prediction. Let us call the condition of applicability of the initial rule `I`. Furthermore, this technique adapts the siblings together such that they cover all the examples for which condition `I` holds and for each example only one of the rules is applicable. This is done as follows: If an incorrect prediction is made by a prediction sequence (cf. page 61), then a discrimination condition `D` is added to the premise of the rule. Consequently, the premise of the rule now takes the form $\texttt{I}\wedge\texttt{C}\wedge\texttt{D}$, where `C` stands for any previously added conditions. The premise of its sibling is changed to $\texttt{I}\wedge\neg(\texttt{C}\wedge\texttt{D})$.

However, many algorithms allow that rules may give contradictory classifications for some examples (e.g., CN2; [RW91]).

Altogether, we could find four principal methods for performing generalizations of a rule (see also section 6.5.3):

1. Drop a condition from a conjunction (or add a condition to a disjunction).
2. Replace a constant by a variable.[53]
3. Weaken a relation among entities in a relation.[54]
4. Replace a concept by one of its generalizations.

Approaches 3 and 4 need some kind of background knowledge (in 4 a concept hierarchy and in 3 the restrictiveness of the various relations), consequently most systems apply only methods 1 and 2. These two methods can also be applied in a continuous form as Becker illustrated in the JCM-system (cf. page 79). This system attaches a criticality to each node and each kernel. The lower the criticality is, the less important are the corresponding parts of the rules for a match. If the criticality of a node approaches zero this corresponds to turning a constant into a variable, while a criticality of zero for a kernel corresponds to dropping the condition.

Similarly to the generalization operators mentioned above, there exist symmetrical specialization operators (e.g., adding a condition to a conjunction).

When rules are applied, usually several opportunities for adaptations arise. In the JCM-system feedback on the success of the rule applications are used for adapting the criticality of the nodes and kernels, and for adapting the confidence in a rule. While most systems change a rule immediately if contradictory evidence shows up, some systems simply store the instances to which the rule must not be applied. This can either be done by storing this set of instances directly (e.g., the exception sets used in MOBAL and BLIP), or by adding blocking conditions (e.g., the `UNLESS`-clauses used in MACBETH-system, cf. Winston in [Hal89]).

An important problem when adapting rules is blame- and credit-assignment. That is, if a chain of rule-applications leads to an (un-)successful result, than it must be decided which rule should be adapted and within this rule which condition should be adapted. Some systems use the

[51] Usually, these algorithms are simply regarded as rule-induction algorithms. However, any incremental algorithm can be regarded as forming a basic result and then performing adaptations.

[52] According to Shen (cf. [She93, p. 153]) an extension to more than two rules is possible.

[53] This can be regarded as a special case of anti-unification (cf. [RW91]).

[54] For example, by dropping an equality constraint, replacing an `AND`-Relation by an `OR`-relation, etc.

Bucket Brigade Algorithm for solving this problem. Here, each rule which contributes to a chain of inferences receives some credit for this. Successful and unsuccessful rules are recognized by summing up the credit over a large number of inferences. A similar approach is taken in the MOBAL-system (cf. [Wro94]). Here a confidence is attached to each rule, based on the ratio of successful and unsuccessful inference chains, it took part in. (The confidence of input facts can be set by the user.) If an adaptation needs to be selected, then the one, which leads to a minimal loss in confidence, is made.

Comparison with Psychology:
Many similarities exist between the adaptation techniques used in AI systems and the techniques proposed in psychological literature for explaining human behaviour (cf. section 3.6.3). An example of this is the PUPS-model. Here a rule is adapted by adding a discriminating condition to the precondition-slot. This approach is similar to the introduction of UNLESS-clauses in the MACBETH-system.

However, an important distinction exists between most AI approaches to adaptation and human behaviour. As we mentioned in the psychological analysis, humans usually do not change the hypothesis they hold as soon as they get contradictory evidence. Instead a certain amount of evidence must sum up. Some AI systems imitate this behaviour by a certain tolerance to contradictory evidence. For example, the JCM-system changes the criticality of the nodes and kernels only incrementally. Another approach is taken in the MOBAL-system, here contradictions to the rules are collected, until enough evidence sums up. If enough evidence against a rule has been found, a restructuring is performed. This can result in a minor modification of the rule, or it can result in a completely new rule set along with the introduction of new concepts. This seems to be a rather good simulation of human behaviour. However, no system we know of bases its decision to perform an adaptation on the availability of other hypothesis, as humans do. This is due to the fact, that rule-based systems typically are not able of holding and evaluating several competing rules.

An important aspect of creative adaptations is the ability to overcome states of fixation. For this the ability to retract assumptions earlier made is of great importance. In AI systems this ability is often implemented with the help of a TMS or using backtracking. The widespread use of these techniques reflects the emphasis they get in psychology. Overcoming sets includes more than the retraction of inferences. It also includes the ability to replace the currently used reasoning technique by a different one, if the current technique is not successful enough. This ability is paralleled in *derivational analogy* (cf. [Hal89]), because this technique introduces a *perseverance threshold* for deciding when to abandon transfer and return to weak methods. A similar proceeding has been proposed in the *explicit-planning method* by Munyer (cf. [Hal89, p. 55]). However, most systems do not possess this ability as they do not have several inference techniques at their disposal.

An aspect of adaptation that has not been examined in AI systems up to now is the *interpretation of features*. Subjects show this behaviour in psychological experiments (cf. section 3.6.3). This is due to the fact, that AI systems (at least those that have been examined) make the implicit assumption that the unique determination of the value of a feature is possible. Additionally, many systems are based on the assumption that their models are consistent. Consequently, they do not use the technique of *shifting to evidence that supports alternative theory-based beliefs*. However, there exist systems that are able to do this. For example, case-based interpretation programs like HYPO construct argumentations for and against a claim (cf. [Kol93]). The CYC-system also has the ability to construct argumentations in favor and against a fact. Additionally, it is able to compare both argumentations. It uses this capability for handling defaults (cf. [GL94]).

6.6.4.2 Adaptation of Problem Solutions

The adaptation of a problem solution is often necessary after performing an (analogical) transfer, because usually the mapping employed cannot adequately reflect all the differences between the source and the target situation. Therefore the generated solution often has some minor errors, which must be corrected before the solution can be used. Especially in Case-Based Reasoning (CBR) the

importance of adaptation is emphasized (cf. [Kol93]).[55] Because hypothesis formation can also be regarded as a design problem (cf. [Kar93]), the adaptation techniques described in this section are also important for forming consistent theories.

Operator: Adaptation of Problem Solutions

Scope: Several techniques for adapting a preliminary problem solution and controlling these adaptations are presented.

Precondition: A preliminary problem solution exists, i.e., it addresses a problem, but does not completely solve it.

Purpose of call: Construct a complete problem solution

Input:
- A preliminary problem solution
- A problem specification

Knowledge used: (background knowledge)

Operators used: —

Knowledge generated: Final problem solution

Tasks generated: —

Contributions from AI:

For fixing small problems with a solution three different approaches are commonly used:

1. The ability which lead to the construction of the preliminary solution may be recursively applied to the remaining inadequacies. This approach is often employed in design (cf. [BGP94, Nav91]).
2. Use weak methods, i.e., search methods like hill-climbing and means-ends-analysis.
3. Use specialized adaptation techniques. This approach is often used in CBR.

In some approaches adaptation and transfer are integrated. For example, derivational analogy (cf. [Hal89, p. 71]) combines the transfer of the solution-path with search. If an inference step from the source solution cannot be directly instantiated, then additional inferences are made for generating a situation, which allows for the application of this step. If the effort for creating such a situation exceeds a so-called *perseverance threshold*, than the analogical transfer is abandoned.

In the enumeration above, methods 1 and 2 denote general problem-solving methods. We did already deal with these methods in other sections, consequently we will concentrate on the third item, which subsumes the techniques specific to adaptation. These techniques cannot be used as independent problem solving techniques, but require additional abilities for constructing preliminary solutions.

When adapting existing (or transferred) solutions to a new situation, often two basically different approaches are distinguished (cf. [RG92]):

- Mutation
- Combination (Merging)

[55] Sometimes a distinction is made between adaptation and repair (cf. [Kol93]). In this case the term *adaptation* denotes the modifications that are made before the result of the transfer is accepted as an preliminary solution, while the term *repair* is used, if the modification is based on the outcome of a mental simulation or a test of this solution. However, the techniques employed for repair and adaptation are similar. Consequently, we will not make this distinction here.

The combination of several solutions into one new solution is only possible with limited effort, if the parts for which the individual solutions are used are non-overlapping (cf. [KW93]). With respect to the area of Case-Based Reasoning, Kolodner states, that up to now no system exists that is capable of real merging. Instead, when using several cases for problem solving, usually one case is used for deriving a basic solution, while the others are used for making fixes to it.

Often approaches to mutation are further distinguished into controlled and random mutation. However, most symbolic approaches employ only controlled (or plausible) mutation. Random mutation is typically used in sub-symbolic approaches like genetic algorithms.[56] Some symbolic approaches combine plausible mutation with a random component. For example, AM and EURISKO employed this approach.

In [Kol93] Kolodner gives a comprehensive overview over specialized adaptation techniques. She distinguishes the following ten adaptation methods:

1. **Substitution methods** are used for replacing role-fillers in the preliminary solution by new role-fillers, such that the result is more appropriate to the current situation.
   (a) *Reinstantiation:* A role-filler in an old case is replaced by a role-filler from the new problem. According to the distinction used in this study this should be regarded as a transfer technique.
   (b) *Parameter adjustment:* This technique is used for the adaptation of numerical values. The adaptation can either be based on numerical differences (e.g., the number of people coming when adapting the amount of necessary ingredients for a meal) or on differences in discrete attributes (e.g., JUDGE uses indications for the amount of violence involved, when it adapts a sentence).
   (c) *Local search:* Starting with the original value a replacement for a slot-filler in the preliminary solution is searched in the abstraction hierarchy of the system.
   (d) *Query memory:* A specification for the needed item is generated with the help of known constraints. This specification is then used for initiating a recall. The recall can be made from the abstraction hierarchy, the case base, etc.
   (e) *Specialized search:* Specialized heuristics are used for recalling, i.e., where to look for an appropriate item is decided based on the slot and the original value.
   (f) *Case based substitution:* This is a special case of *query memory* where the search is performed in the case base.
2. **Transformational methods:** while substitution methods only change role-fillers, these methods can also change the structure of a case.
   (a) *Common-sense transformations:* These are simple heuristics of rather general nature (e.g., `delete-secondary-component` = delete an overconstrained component if it is not very important).
   (b) *Model-guided repair:* Here a causal model is available, and changes to the solution are triggered by problems in generating a successful explanation for the solution.
3. **Other methods:** methods which cannot be grouped under the two headings above.
   (a) *Special-purpose adaptation and repair heuristics:* These are specialized heuristics for adaptation. According to Kolodner (cf. [Kol93]) most methods for structural adaptations of a problem solution are implemented as special-purpose heuristics.

[56] Some supporters of sub-symbolic approaches propose the view that true creativity can only be achieved by random mutation. However, in psychological research often the view is held that it is of decisive importance to the creative process that useless combinations are not made (cf. [SK74, Bod91]).

(b) *Derivational analogy:* We mentioned already, that this method is more adequately described as a method implementing non-specific transfer than as an adaptation technique.

Prior to applying an adaptation technique it must be decided which part of the preliminary problem solution shall be adapted and which technique should be used for doing this.

The first step in recognizing necessary adaptations is the identification of a problem with the preliminary solution.[57] In [Kol93] the following types of hints to problems are described:

- Differences between the previous and the current problem specification point to a need to fix parts of the old solution.
- Inconsistencies between the proposed solution and stated goals and constraints of the new problem are identified and point to a need for fixes.
- A checklist identifies standard problems, that, if present, must be fixed.
- Adaptations done previously, point to a need for compensatory adaptations (e.g., a previous adaptation may disturb the balance of a solution that must be preserved; this must be fixed by additional adaptations).
- The results (and their analysis) of carrying out (or projecting) a solution points to problems that must be fixed.

However, a mistake does not uniquely determine what change must be made. An example will illustrate: JULIA is a system that proposes menus based on given constraints. Let's suppose it found a solution with a main dish that contains beef, but then it is told that a vegetarian will also come. Potential adaptations would be to exchange the beef for something vegetarian, to choose a different main dish, or to search for a completely new menu.[58] Additionally, the assignment of a problem cause to a problem is often more complicated than in the example above, e.g., if a problem with a complex device turns up.

When choosing a part of the solution for adaptation three aspects need to be considered:

Connectedness: How many constraints connect this element of the solution with other parts of the solution?
The more constraints exist, the more complex the adaptation task will be.

Centrality: Is the element central to the solution?
For example, the main-dish is more central to the solution than a side-dish. The more central a part is, the more complex it is to find a replacement for it.

Closeness of substitution: How similar are possible replacements to the current element?
Usually, it is easier to replace a part by a similar element than by a dissimilar one.

In general, simple substitutions should be preferred over more complex ones and substitutions should be preferred over transformations. As the enumeration above shows, the selection of the component that shall be adapted and the available adaptation techniques tightly interact.

If a problem has been identified, the problem solver must identify appropriate adaptation techniques in memory. Several identification techniques are in common use ([Kol93, p. 456]):

- An adaptation strategy is proposed together with the identification of the part that should be adapted. This approach is often implemented using checklists or case-based adaptation.[59]
- Some items are associated with close substitutes.

[57] For a general description of approaches to recognizing problems see section 6.2.

[58] A somewhat more creative system would consider to retract the invitation to the vegetarian.

[59] In case-based adaptation the case-memory is searched for a case with a similar problem and the adaptation strategy used in this case is proposed.

- Applicable adaptation strategies are indexed by the situations in which they are useful. This indexing can either be based on an existing situation description (e.g., in JULIA, [Kol93]) or a special characterization can be generated specifically for this purpose (e.g., in SWALE, [SL89]).

For choosing among possible adaptation strategies Kolodner proposes several preference rules:

- Prefer specific adaptations (i.e., adaptations that change the current case only a little bit) over general ones.
- Simple adaptations should be preferred, e.g., local search, parameter adjustment, specialized search.
- Adaptations with a high probability of success should be preferred.

Comparison with Psychology:
As we mentioned in section 3.6.3 virtually no specialized adaptation technique for problem solutions is described in psychological literature. One exception from this rule is the necessity of overcoming states of fixation during problem solving. This is often described as a very important activity for creativity. This ability has two aspects: first, wrong assumptions must be retracted, second, a wrong problem solving approach must be abandoned and replaced by a different one. We mentioned these two aspects already in the preceding section. Both have counterparts in AI: while TMS and backtracking can be regarded as techniques for implementing the first aspect, methods like derivational analogy or Munyer's explicit planning method (see page 102) implement the second aspect, as they include the ability to abandon a problem solving technique if its application is not helpful. However, systems that integrate multiple problem solving techniques are still rare.

While in psychological research the techniques used for the adaptation of problem solutions to new situations has received only little attention, this area has received a lot of interest in AI, especially in CBR. Some authors (e.g., Kolodner, Schank) emphasize the role adaptations play in the generation of creative products. Especially, complex adaptation techniques like transformation methods are likely to produce creative results (cf. [Kol93]). The application of adaptation methods in unusual situations (so-called out-of-place-adaptation) is also regarded as being able to generate creative solutions (cf. [KW93, SL89]). If one compares this with the recommendations Kolodner gives for the application of adaptation techniques, than it gets obvious that such adaptations are usually not applied, as simpler ones (which are usually also more promising) are preferred. However, this behaviour is psychologically sound, as it is reported that usually people fall back on more complex problem solving approaches only if simpler ones give unsatisfactory results (cf. [SK74, p.79]).

The greater the changes to a previous solution are, the more important gets the evaluation of the result, as the appropriateness of the result can be less justified based on the appropriateness of the original solution. Consequently, the application of complex adaptation techniques and out-of-place-adaptation requires that the results get thoroughly evaluated. A second reason that makes good evaluation techniques necessary is, that the larger the allowed transformations get, the more possible modifications exist. Therefore, evaluation of the available possibilities gets more important as only the plausible modifications should be further examined. The importance of good evaluation mechanisms for creativity is also emphasized in psychological literature (cf. [Bod91, SK74, Ulm68, Wei89]).

Usually, a problem solution which is generated by the combination of several solutions is regarded as involving more creativity than a solution which is built by making modifications to a single solution. However, as we mentioned above, no AI system to date is really able to merge several solutions into a new one. This will probably be a topic for future research.

6.6.4.3 Modification of Problem Specifications

The modification of the problem specification (i.e., the constraints that define an acceptable problem solution) is regarded as the central criterion for creativity in *creative design* (cf. [KW93]). The

importance of this ability for the creative process is also emphasized in psychological research (cf. [McL92]).

However, as *creative design* is a relatively new field within AI only few systems exist up to now that are capable of adapting problem specifications.

Operator: Adaptation of problem specifications

Scope: Several techniques for adapting problem specification and the reasons that may lead to an adaptation are described.

Precondition:

1. Problems in the reasoning arise (e.g., inability to generate or evaluate).
2. Evaluation yields interesting results.
3. An exploration of the problem space shall be performed.

Purpose of call: Exploit interesting properties and opportunities connected with the current problem.

Input: Reasoning problems or evaluation results

Knowledge used: Background knowledge about constraints in the current domain.

Operators used: (mental simulation)

Knowledge generated: New constraints

Tasks generated: —

Contributions from AI:

One of the first *creative design* programs was CYCLOPS (cf. [Nav91]). This system has the ability to add constraints to its initial problem description and to relax constraints to allow the exploration of a wider range of possible designs. According to Navinchandra, the basic idea of this approach is, that it is usually impossible to define all relevant criteria for a new design before beginning the construction.[60] Consequently, Navinchandra distinguishes between prior criteria, emergent criteria, and post-design criteria (cf. page 49).

The CYCLOPS-system has been applied to the domain of landscape design. The program is controlled by an A*-algorithm, which guides the search for a partial design. During this search the design gets successively completed. This search process is combined with a case library, which is used for evaluating partial designs and repairing problems. A partial design can remind the system of an earlier case, which is retrieved and examined for additional important criteria. These criteria are then added to the current problem specification (emergent criteria). The additional criteria can influence the evaluation of the current design either positively (opportunities) or negatively (potential problems). Additionally, the system can relax constraints, e.g., when it is required that the houses face south, a design, in which the houses deviate a little bit from the south, may also be accepted.

However, the relaxation of problem descriptions is a much older theme in AI. Every program for mediation begins with an overconstraint problem (find a solution which is acceptable to the conflicting parties) and must subsequently relax the constraints on the solution to find a compromise (cf. [Kol93]).

Usually, the constraints that make up a problem specification do not all have the same importance for the (design) problem. Additionally, as Kolodner and Wills ([KW93]) point out, each problem domain has its standard constraints, which are usually not made explicit in a problem description. However, making them explicit and perhaps more concrete (i.e., operational) can help the problem solving process in various ways. For example, they can be used as additional indices

[60] This view is shared by other authors, e.g., Fischer ([Fis92]), Logan and Smithers ([LS92]), etc.

into the memory of the reasoner, they can be used for distinguishing between several domains[61], they are needed for explicitly manipulating them, etc. However, the programs, that exist today are restricted to manipulate constraints of minor importance.

A system that is able to change its problem specification is hard to evaluate with respect to its success in problem solving. Consequently, Kolodner and Penberthy ([KP90]) distinguish *primary* and *secondary constraints* and their program is only allowed to change secondary constraints. However, the ability to modify constraints of higher importance is necessary for achieving a high degree of creativity. We will return to this topic in the comparison with psychology.

Several reasons may trigger a modification of the problem specification. In [KW93] these reasons are distinguished into three main categories:

1. Inability to evaluate
2. Inability to generate
3. Opportunism

Out of these, opportunism is probably the most important source of modifications, as most studies emphasize it. Consequently, Kolodner and Wills propose an opportunistic control for a creative reasoner. However, if both an adaptation of the problem solution and a modification of the problem specification is possible they prefer the adaptation, as this is usually less costly.

Opportunities for a modification of the constraints can be rather easily recognized by a comparison of the current design with a previous solution. Consequently, most systems for creative design possess a case-based component.

The reasons mentioned above may lead to different types of modifications. An inability to evaluate a design points to the need of making the existing constraints more concrete (i.e., to operationalize them). However, we know of no system yet, which possesses this ability, although Kolodner and Wills report evidence for the use of this technique by humans (cf. [KW93, p. 52]).

The inability to generate an adequate design is usually regarded as being caused by an overconstrained problem specification. Consequently, this leads to dropping a constraint or relaxing it. The relaxation of constraints has already been described in the context of the CYCLOPS-system above. Dropping a constraint is often regarded as an opportunistic process: constraints which could not be satisfied in the generation process are dropped, while those which could be satisfied are preserved. The inability to generate an adequate design can also lead to the introduction of additional constraints. If a problem with the current design is discovered, additional constraints are added for avoiding these problems in the future. An example of this approach is illustrated in the IDEAL-project (cf. [BGP94]). Here, a model-based approach is used for discovering problems with an existing design (e.g., the coffee will not stay warm), understanding (i.e., explaining) them, and finding an additional constraint (e.g., the coffee needs to be warmed).

The opportunistic approach is based on the identification of possible problems and advantages of a (partial) solution. Possible problems can be identified by mental simulation or by recalling problems with similar designs. Similarly, advantages can be identified. However, for enabling the system to perform such an identification, the problem solver must either possess the necessary domain knowledge (then this activity can be regarded as an operationalization of the knowledge) or the problem solver has been told of these criteria in previous cases. In this case this ability allows the problem solver to avoid making the same mistake twice, or to find solutions the user will be more comfortable with.

While the modification techniques mentioned above are based on the construction of a design, it is also possible to explore and stretch the problem constraints simply for identifying what will be acceptable. Any loop-holes found in this process can then be exploited. Kolodner and Wills ([KW93]) illustrate this with an example from the domain of architecture: given the task to "design a building between two buildings" a designer might ask how close the middle building may be to the

[61] This gets necessary if a problem solver shall be competent in several domains at the same time.

two adjacent buildings. By taking closeness to the limit, the problem may be modified to "connect the two buildings together".[62] A system capable of doing such *constraint stretching*, would then either use its background knowledge for deciding this question or would consult the user.

As the examples given in this section show, this is still a very immature field within AI.

Comparison with Psychology:
In section 3.6.3 we noted already, that the modification of the problem specification is also regarded as an important activity in psychology. In spite of this importance, no detailed description of the techniques humans use for this could be found in literature. The descriptions of this activity given in AI literature are not very detailed, too, as the development of these techniques is still in its early stages. However, the techniques described above can be regarded as being psychologically sound as they are confirmed by observations of human designers which have been conducted by Kolodner and Wills (cf. [KW93]).

The modification of the problem specification gives rise to a major problem: that of evaluation. It is hard to assess whether the solution of a modified problem can still be accepted as a solution, if constraints which originally belonged to the problem specification are relaxed or dropped. Indeed, this may vary from person to person. The example of the Sydney Opera House nicely illustrates this point (cf. page 7). This is especially important for a computer program, because the readiness of humans to accept a creative solution from a program is even lower than accepting such a solution from a human. This point is well illustrated by the critiques of EURISKO's victories in the TCS-competition. The program did creatively — and successfully — deviate from the usual and was consequently accused of cheating. But as Boden points out "creatively breaking the rules, or even bending them, could always be called 'cheating'" (cf. [Bod91, p. 110]).

This aspect gets the more important, the more basic the modified constraints are. The most basic constraints in a domain are called *standard constraints* in [KW93]. Boden calls the search space, which is defined by such a set of constraints, a *conceptual space*. She distinguishes two types of creativity: the exploration of a conceptual space and — more importantly — the transformation of conceptual spaces. Especially, if conceptual spaces are transformed during the creative process, the result is usually not readily accepted by humans. General acceptance may even need a considerable amount of time. Such a transformation is similar to the *changes of paradigm* defined in [Kuh73].

Even if less fundamental constraints are modified, it is important that the modifications are justified, as this is an important criterion for distinguishing between creativity and schizophrenic production (cf. [SK74, p. 31]). Consequently, programs that are able to change their problem specifications must either possess much background knowledge about their domains and their problems or they must rely on user-support for making such adaptations.[63]

Due to the problems accompanying the modification of problem specifications, an adaptation of the problem solution should always be preferred to a modification of the problem specification, if both possibilities exist. This preference has also been observed in human behaviour (cf. [KW93]).

6.7 Control

While the preceding sections gave an overview over various operators which are especially important to the creative process, in this section we will propose a control mechanism for integrating these operators into a useful whole.

6.7.1 Introduction

While individual operators have been thoroughly examined in AI literature as well as in psychological literature, only little material is available about the control structure of a reasoner which has to

[62] Note, the similarity of the heuristic employed with AM's heuristic 218, given on page 63.
[63] The cases in CYCLOPS that make additional criteria available to the program can be regarded as pre-coded user-support.

integrate the diversity of operators discussed in the preceding sections. Consequently, we will not discuss existing implementations of control structures in this section. Instead we will discuss a model for a control structure that satisfies the diverse constraints highlighted in the preceding sections. Naturally, wherever it is possible we will contrast this model with relevant material from psychological and AI literature for demonstrating that it is sound and implementable.

In the psychological part of this study we devoted a separate section to evaluation (section 3.7). However, as evaluation and control do strongly interact in our model we will not discuss evaluation separately, but as a component of control. In section 6.7.5 we will concentrate on the evaluation of the results of operator application, while other aspects of evaluation will be discussed throughout this section.

The control of problem solvers is well known to be a complex and large area of research. Many of these complexities are relevant with respect to creative problem solving. For example, in (creative) design usually several (sub)goals may interact, thereby making nonlinear planning necessary. Further, the complexity of the problems which need to be solved will usually make abstraction necessary. However, we will not discuss these problems here, as they are well beyond the scope of this study. Instead we refer to the standard literature on this subject (cf. [Her89]). The discussion in this section is limited to the special problems a creative problem solver will encounter. These are:

- The problem specification may change during the course of problem solving.
- Opportunities must be recognized and exploited.
- The control component must be able to learn, as a creative problem solver must be able to acquire expertise in a new area.

6.7.2 The Model of Control

In section 6.2 we already introduced the distinction between *problem symptom*, *problem cause* and *problem solving task*. This distinction forms one of the most pronounced differences between the control architecture proposed here and other architectures.

The architecture of the control component can be further subdivided into internal management and activities that must be carried through or triggered. In this section we will only discuss the management activities. The other activities will be discussed below, each in a section of its own.

The basic management task of the control component is to manage active problem symptoms, problem causes, problem solving tasks, and the relations between them. Figure 6.1 gives an example of such an relation.[64] Throughout this section we will assume that problem solving tasks are given declaratively, as this prerequisite is necessary for performing the adaptation of problem specifications, which we identified as an important activity for creativity. Consequently, we will use the terms problem solving task, problem specification, and goal interchangeably throughout this section.

Problem symptom and problem cause are related through an explanation. A set of problem causes may be connected to a problem symptom by an arbitrary and/or-tree. This is due to the fact that several causes may interplay for producing a problem symptom or different alternative explanations may exist for a symptom, between which a discrimination is not possible based on the available information. Problem cause and problem solving task are connected through a remedy. Here again, several possible remedies may exist, which may have a probability of success smaller than one.

There exist other relations besides these, that must be handled. For example, if a problem symptom occurs, it may no longer be possible to work on the task which generated the problem symptom. Consequently, this task must be suspended until the problem symptom does no longer exist. Additionally, a task may be subdivided into a number of smaller tasks, between which relations may exist, too. Moreover, several possibilities for task decomposition may exist, leading to

[64] We will discuss methods for attaching problem causes to problem symptoms and problem solving tasks to problem causes in section 6.7.5.2.

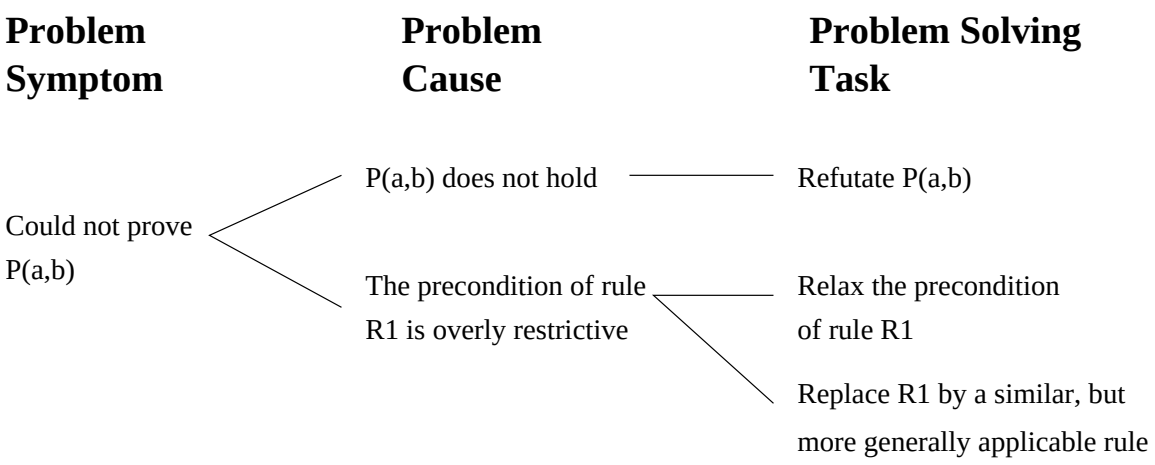

Figure 6.1: A simple example of the relation between problem symptom, problem cause, and problem solving task

nearly arbitrary and/or-relations between a task and its subtasks.[65] However, the relation between problem symptom and problem solving task can also be uniquely determined. For example, if a task is provided by the user, then problem symptom and problem cause may even be identical.

Basically, there are three types of events which need to be handled by the dependency management.

- A new task is generated by task decomposition.
- A derivation graph of the form problem symptom/problem cause/problem solving task has been generated.
- An operation has been completed successfully.

If problem symptoms, problem causes, or problem solving tasks are made available to the management component, then it is useful to check whether these have already been encountered before, as it is especially important to remedy problems (symptoms or causes) that occur frequently. Problem solving tasks which can (help to) remedy several problems at once are of especial importance, too (opportunistic control).[66]

In section 3.3 we pointed out already that a human problem solver can be expected to have at least the following two basic motivations:

1. Tendency to seek new information
2. Tendency to maintain a consistent world view

In this section we assume that a creative problem solver, which can be expected to have some functional autonomy, will have several basic goals, including the two mentioned above. A newly introduced problem cause may also be related to such a standard goal. This must be recognized by the management component and registered.

If an operation has been completed successfully and a problem cause or a problem symptom has been removed, than problem solving tasks and causes, that are no longer connected to an open

[65] Possibilities for handling task decomposition are discussed in more detail in section 6.7.3.2.

[66] Lenat ([Len82a]) reports that basing the task selection scheme in AM on the reasons that support a task has been helpful to AM's performance.

problem can be deleted.[67] Moreover, tasks that were suspended earlier, can now become active again.

The managed dependencies have considerable influence on the control flow, as they play a decisive role in task selection. We will see this in more detail in section 6.7.3.1. However, these dependencies also have a different important role: they set the frame for changing problem specifications. A problem specification can be changed as long as its solution will still remedy the original problem symptom. Naturally, this criterion is only directly applicable for problem specifications generated by the problem solver itself. However, we believe that a similar context can also be constructed for externally provided problems. However, this makes additional knowledge necessary.

As we pointed out already, besides handling management tasks the control component must also perform some actions of its own. These are carried out in each control cycle in the following order:

Selecting actions: A task is chosen and decomposed into subtasks until a level is found on which an operator can be applied to it. Then an adequate operator is chosen.

Operator application: The chosen operator is applied to the task.

Evaluation: The results of the operator application are evaluated. The evaluation is made (at least) along the following dimensions: were problems encountered? was the goal reached (or approximated)? shall the generated information be added to the knowledge base?

Learning: Based on the results of operator application the ratings of the operators and information used is adapted. Additionally, new operators and control rules may be learned. A change of the evaluation criteria may also take place.

The basic model of control which has been proposed in this section combines more traditional aspects of the control of problem solvers with *functional autonomy*. We think that the latter is very important for enabling a problem solver to improve its own problem solving capabilities, which in turn is a very important ability for a creative problem solver. The abilities we require of a control component may be rather complex, however, we believe that such a component is realizable, because its various parts have already been implemented in other systems.

6.7.3 Choosing Actions

The first step in the control-cycle must establish a current goal and an operator application which may bring the problem solver closer to its goal.

6.7.3.1 Selecting Tasks

Usually, a large number of tasks will be available for execution. Out of these one must be selected for further work. Some of these tasks may strongly depend on each other. For example, one may need data a different task will produce. This may allow for the exclusion of some tasks. However, after this step there will usually still be an abundance of tasks. For ordering these several important criteria could be identified:

1. Some tasks may have a dependency, which recommends a certain ordering.
2. Some tasks may be more urgent than others.
3. Some tasks may be expected to produce more interesting results than others.
4. Some tasks may be expected to produce results of higher utility than others.
5. The behaviour of the problem solver should be focused.

[67] They are not necessarily removed from the entire system, instead they can be helpful for testing newly acquired heuristics.

These criteria (and perhaps some others) can be combined into a heuristical measure for selecting among the competing tasks. In each cycle the most interesting task is chosen and further decomposed. Consequently, the control component will not spend too many resources on the decomposition and evaluation of tasks that are probably not very interesting. Rating tasks on a rather abstract level makes some experience necessary. As a creative problem solver must be able to move to new areas of competence, the control component itself must be capable of learning.

Now we will examine the criteria we propose for task-assessment in more detail:

Recommended orderings. Even if no strict dependency between two tasks exists, one ordering of the tasks may be preferred over the other. For example, one task may be expected to produce a data item, which falls into the range over which the second task shall induce a regularity. Although, it is possible to induce a regularity first over the existing data and adapt it later when additional data becomes available, it may save some effort to produce the data first and induce the regularity later.[68]

These orderings are sometimes called *micro-strategies*, as they only give a partial ordering. Causse and Lebbe show that micro-strategies can also be represented declaratively (cf. [CL95]).

Urgency. Some tasks may need more rapid response than others. For example, a problem posed by the environment (e.g., by an impatient user) which demands for a reaction by the system should be regarded more urgent than a problem generated by an internal goal of the system.

Interestingness. Some tasks may promise more interesting results than others. Throughout this study we could identify several hints supporting the hypothesis that the interestingness of a fact can be mostly identified with its information content. However, one must clearly distinguish between the expected information content of a result and the information content of the actual result. Expectations for the information content of a result can be based on existing knowledge supporting possible outcomes. The more clearly one of the possible outcomes is favored by this knowledge (e.g., based on the reliability and the number of argumentations for this result) the less information content this result should be expected to have. Consequently, the uncertainty of the result can be regarded as a measure for predicting the expectation value of the information content. However, after a task is carried out things are different. If it turns out that the actual outcome of a task differs from the expected, then this is a very interesting event, too. Here, it is the more interesting, the better supported the *wrong* outcome is.

Besides the uncertainty of the result there are additional criteria which influence the interestingness of a task. Among these are the interestingness of the objects the task refers to, the actions it may involve, etc.[69] The interestingness of an object is influenced by the interestingness of the information that has been uncovered about it.

Utility. The utility of the tasks may differ just as the utility of their expected results does. For example, a regularity in a domain to which many of the problems that are given to the problem solver belong, is more utile than a regularity in a domain which has no relations to the typical problems of the problem solver.

Focus. This is a deviation from the purely information guided criteria mentioned above. Consequently, we can regard this as a handicap to the system. However, we think that this is rather important for a system which needs to interact with a human, as it results in a more plausible reasoning chain and enables the user to follow this chain. Two basic types of foci can be distinguished:

Task-oriented: Tasks that are close to the previous task in the tree, which resulted from the task decomposition, are preferred.

[68] We have already seen an example of such a recommended ordering in the OCCAM-system (cf. page 84). Here, more knowledge-intensive methods are used before more data-driven methods. The IDS-system (cf. [NL93]) for *scientific discovery* can also be interpreted according to this principle. Here, the different techniques are used in the order *taxonomy formation*, *qualitative discovery*, *quantitative discovery*, as the more abstract technique structures the search space for the more detailed technique.

[69] Such a scheme is used in the AM-system for rating tasks. The worth of a task T ($worth(T)$) is given by the formula $\| \sqrt{\sum R_i^2} \| \times (0.2 \cdot worth(A) + 0.3 \cdot worth(F) + 0.5 \cdot worth(C))$ where A denotes the action involved, C the concept and F the facet (the slot of the concept); R_i denotes the worth of the reason. However, as Lenat points out, the exact form of the formula was not important.

Data-oriented: Tasks that refer to the same (or related) information (e.g., the same regularity) as the previous did are preferred.

A combination of these techniques, together with other focus-heuristics, have been used in AM for generating a plausible chain of reasoning (cf. [Len82a]) with some success. From a technical point of view a focus may confer an additional advantage on a problem solver: it may increase its *locality*, thereby improving its performance.

Some additional criteria like the *opportunity* for performing a task and the expected success of this task are best judged based on the applicable operators, their current costs of application and their expected success. Consequently, we will defer the discussion of these criteria until section 6.7.3.3.

If no worthwhile task exists problem recognition tasks may be generated. this is similar to the suggest heuristics used in AM.

6.7.3.2 Task Decomposition

While rather simple problems can be solved by directly applying an operator to them, this is usually not true for more complex problems.[70] Consequently, if no operator application is found which may solve the current problem, it must be decomposed into simpler problems until an operator is applicable to the resulting problem specification.[71]

Several different approaches to deriving subgoals can be distinguished:

1. *Decomposition patterns* for a goal may exist.
2. *Decomposition strategies* can be used.[72]
3. *Acquisition strategies* can be used for acquiring additional problem solving knowledge (including additional operators).

A decomposition pattern is a scheme, which can be used for replacing a goal by a number of subgoals. If all these subgoals are achieved, the original goal is achieved. These patterns are usually provided to the problem solver at the beginning, however a case can also be interpreted as a scheme. Consequently, as new cases are acquired new schemes can be learned.

Decomposition strategies are a more complex technique for decomposing goals. These strategies are procedures for achieving a goal by achieving subgoals. The main difference to decomposition patterns is that the generation of additional subgoals depends on the evaluation of the results of the previous ones. For example, if the first subgoal has been successfully achieved, the second subgoal is simply put into the scheduling list. Otherwise an additional subgoal may be inserted into the scheduling list for repairing this failure. This has the consequence that only one subgoal at a time is available for scheduling, while a decomposition pattern makes all subgoals immediately available.

Acquisition strategies are different from the two techniques mentioned above, insofar as they do not strive for achieving the goal immediately, but for improving the problem solving capabilities of the system first. Acquisition strategies play a role similar to the situation assessment stage in CBR (cf. [Kol93]). These strategies can either strive for the acquisition of additional data or for the acquisition of additional problem solving knowledge (e.g., heuristics). However, the latter will usually establish the need for additional data, too. (This relation is depicted in figure 5.1.)

[70] However, there is no such thing as the largest size of a problem that can be solved without decomposition. For example, in CBR cases can be of arbitrary size, thereby enabling a system to reinstantiate any previous problem solution without decomposing the problem.

[71] Here, we discuss task decomposition, as if all tasks would be on the same agenda. However, realistic implementations will probably diverge from this model due to efficiency reasons. For example, the control component may be recursively invoked or different control components may exist for different levels. The latter approach was used in EURISKO-system, which was even able to build new rule-interpreters if needed.

[72] Some of the operators discussed in the previous sections can also be described as decomposition strategies, as they can be decomposed into more primitive operator applications.

Consequently, these strategies only add additional goals to the agenda, but do not achieve the original goal in the first place. We will return to this kind of strategies in section 6.7.5, as they are also employed if evaluation detects an impasse in problem solving.

6.7.3.3 Selecting Operators

After problems have been decomposed far enough that operators are applicable to the subproblems, for each subgoal an appropriate operator must be selected. Usually more than one operator will be applicable.[73]

When an adequate operator instantiation must be chosen, additional constraints besides those present in the task selection may be present. For example, additional data dependencies, which have not been obvious from the task description, may exist. Additionally, a focus can be interpreted as a constraint on the preferable operator instantiations.

Besides the criteria already discussed in the context of tasks, there are basically two criteria for choosing among possible operator instantiations, which can only be adequately evaluated for a specific operation. We mentioned already *opportunity*, *expected success*, and *expected costs*[74] of a task. On page 105 we identified already some criteria for judging the cost of an adaptation: connectedness, centrality, closeness. Opportunity can be interpreted as the deviation of the expected costs from the usual in the current situation. For example, if an experiment setup already exists, the costs of performing the experiment is dramatically reduced.

Clearly, the problem is to compute expectation values for success and costs. As far as we know there is no system to date which uses such estimates.[75]However, using such estimates may proof very helpful for comparing the usefulness of the available operators, especially in a system which possesses a wide range of different operators, as the one proposed here does. Rules for generating expectation values could either be given to the system at the outset, or the system could derive them from observations of its own behaviour. Such an information would confer an additional advantage on the system (besides the selection of appropriate operators): it could also be used for finding hints to new problems (e.g., an operation that took shorter than expected may hint to a possibility for simplification). Additionally, the probability of success of an operator application must be identified. However, this will usually not be given as a single number, but instead as hints to the applicability of operators. These hints come in two forms:

- plausible move generators (if in situation `sit1`, do `act1`).
- censors (in situation `sit2` refrain from doing `act2`)

The EURISKO-program demonstrated that both types of heuristics can be induced successfully from the program's behaviour. We will return to this point in section 6.7.6.

The hints thus generated for the possible operator applications can then be combined into an expectation value for the task. This in turn provides additional information for choosing among the different tasks.

As we mentioned already in the preceding sections, the selection between the various possible tasks and operators should be based on a conservative strategy. That is, the actions which are most promising and coupled with the least effort should be chosen at any moment. Risks should only be taken, if necessary. That is an existing solution should be adapted instead of constructing one from scratch, the problem specification should be adapted only if this is necessary, etc.

[73]Remember, that we discussed categories of operators in the previous sections. Consequently, there will usually exist several alternative methods in each category. Additionally, the arguments for the operator must be adequately chosen (e.g., the knowledge which shall be used by an inference operator).

[74]The costs of a task can again be interpreted along several dimensions. For example: consumed memory, cpu-time, number of user interactions, etc.

[75]However, EURISKO and AM monitored the costs and successes of their operator applications and used these results as a basis for further action.

Different techniques can be used for finding operators (respectively operator instantiations). The following enumeration gives a short (although not complete) overview over possible techniques:

- Operators can be augmented with information about when they should be applied. We approximated this already with the preconditions given for every operation in the previous sections.
- A checklist can be used for identifying applicable operators.
- Operators can be proposed together with the tasks.
- An explicit representation of the goal hierarchy exists in which each goal is associated with methods for achieving it.
- Operators are attached to the most general concepts in the concept hierarchy to which they are applicable.[76]

  ⋮

The selection scheme seems to be at odds with the goal of creativity, as this is commonly identified with performing unusual or extraordinary actions. However, we believe that creativity mainly originates from other dimensions, e.g., from differences in the evaluation of interestingness, from additional criteria and task generators, etc. This view is also supported by results we mentioned in the psychological part of this study. For example, we mentioned that creative people usually evaluate their achievements very critically (see page 34), that they are able to learn new strategies (see page 36), etc. Additionally, creative people are usually productive, which is only possible, if they work rather goal-oriented and not if they follow every possible path.

6.7.4 Operator Application

After an adequate operator instantiation has been chosen, the operator must be applied. For doing so, a problem solving context must either be built or selected from the existing contexts. Here, an additional advantage of providing the system with a focus becomes visible: if the system refers to the same PSC during several successive operator applications, no context switches are necessary, leading to an improved performance of the system.

Besides creating an adequate environment for operator application also a decision must be made about the resources available to the operator. The amount of resources made available will usually depend on the estimation of the costs of the task, the number of alternatives available, the importance of the task, etc. In a system which possesses several alternatives to an operator application, the restriction of the resources available during execution of the operator is useful, as it may be more effective to abort an operation that consumes to many resources and try a different one than to wait until the operation terminates. Such a *perseverance threshold* is also used in *derivational analogy* (cf. [Hal89]). If too much effort is needed for reinstantiating a derivational trace, it is aborted. The AM-system also limits the amount of resources its operators (i.e., its heuristics) may consume. This limitation restricts memory and cpu-time-usage and is based on the interestingness of a task, i.e., the more interesting a task is, the more resources it may consume.

6.7.5 Evaluation

The task of evaluation is to provide feedback to the control-loop based on the results of operator applications. For doing so, evaluation must perform three different functions:

Preliminary assessment: The usefulness of the result of the operator application must be judged.

Blame/credit-assignment: The reasons for the result being (not) useful must be detected.

[76] Linking items with close substitutes (cf. page 105) can also be regarded as an instance of this approach.

Rating update: The ratings (validity, utility, etc.) of information used during execution of the operator needs to be updated.

We will discuss these three functions in the given order. However, in real implementations they closely interact. Additionally, any subtask of evaluation, which would take too much time or is not urgent can be explicitly represented and handed over to the control component for later scheduling.

6.7.5.1 Preliminary Assessment

An operator will only yield useful results, if it could be executed successfully. Consequently, the first step in the preliminary assessment phase is to check for any problems encountered during operator application. We discussed problem recognition already at length in section 6.2. Problems in the reasoning process will usually be detected during operator application, as they will lead to a termination of the execution. Some of the problems in the knowledge base will also be recognized this way. However, for some problems in the knowledge base and for problems in the environment evaluation will usually need to check for explicitly.

Some of these checks can be made immediately (e.g., was the phenomenon already observed before?), while others may be deferred, because they would take up too many resources or are not related to the current goal (e.g., try to explain an unknown phenomenon, which has no obvious connection to the current goal).

After a check has been made for any problems encountered, the evaluation of the results of the operator application takes place. This may also include any intermediate results produced. Whether evaluation of results can take place after a problem occurred dependents on the severance of the problem.

It is clearly not sufficient to test whether the results thus produced solve the current problem solving task. Instead it must be estimated whether they approximate a problem solution. For example, in search problems it is common to use heuristics for estimating goal distance, and any operator application reducing this estimate can be regarded as successful.[77] Similarly, the result of an analogical transfer will usually not solve the problem directly, but will only be a preliminary result. For some problems an exact solution will not even be possible, leading to a need for changing the problem specification. An important problem of estimating goal distance is that the goal description may include a discrete test (e.g., house faces south), while the test for approximation needs to be based upon a continuous attribute (e.g., deviation from the south). Usually, these relations are precoded (cf. [Nav91]), however sometimes it is possible to generate heuristics for estimating goal distance automatically (see page 122). Sometimes it may also be useful to chose overly criteria for generating additional opportunities for learning. We will return to this aspect in section 6.7.6.

An additional complexity is very important in the context of creative problem solvers: even if the whole result is not useful with respect to the current problem solving goal, it must be recognized if certain aspects are extraordinarily well achieved (e.g., the designed artifact is very cheap). These aspects need to be examined during credit-assignment, as they may lead to the opportunistic introduction of new criteria (build it using plastics). Consequently, the evaluation method employed must be able to evaluate different aspects separately.

Basically, three different approaches for judging the validity of new information exist:

1. Theory based
2. Test based
3. Analogy based

[77] However, it may be necessary to make operator applications that increase this estimate, for deriving a solution.

Theory based. Two different kinds of theories may be used for estimating the validity of new information: domain theories and meta theories. Domain theories can be used for justifying information which has been transferred using analogy. Hall mentions that this approach is often used in analogy (cf. [Hal89]). On the other hand a meta theory would contain rules like "deductively generated information receives the same validity as the information used for producing it", while induction based information would receive a somewhat lower validity, the details depending on the examples and the generalization techniques used.

Test based. The validity of information can be assessed based on experiences gathered in using the information. These experiences may either be generated during normal use (cf. section 6.7.5.3) or during specifically generated test tasks. We already discussed the later approach at length in section 6.4.2.

Analogy based. An estimate of the validity of information can also be based on analogous reasoning. That is a design is regarded as a valid solution to the current problem, if a similar design was a valid solution to a similar problem (cf. [Kol93]).

All these activities may be executed as separate tasks if they would consume too many much resources.[78] If any problems or opportunities are recognized during these tasks, this may lead to the resumption of an earlier problem solving task. If it is recognized that the result does not solve the problem, it depends on the estimate made in this phase of evaluation whether more resources will be devoted to this line of reasoning or not. Additionally, it may be decided to try an adaptation of the problem specification, if either opportunities were recognized or problems during generation or evaluation of a solution arose (cf. page 108).

6.7.5.2 Credit/Blame-Assignment

The task of the credit/blame-assignment phase is to "explain" why the result of the operator-application was as (un-)successful as it was. That is, in case of an error it must be found out, whether any incorrect knowledge was used, whether the operator was used in the wrong situation, etc. Likewise, in case of a successful operator application, it must be recognized what contributed to the success.[79]

Basically, this problem is similar to the problem of diagnosis. This emphasizes its inherent complexity. As credit- and blame-assignment may only take up a small amount of resources, it is obviously impossible to perform a complete diagnosis. This and the reason that an exact diagnosis is usually not possible due to a lack of knowledge makes it necessary to allow for mistakes in credit/blame-assignment. This has been done by supporting multiple, alternative causes.

Three basic approaches to credit/blame-assignment can be distinguished:

- Theory-based approaches
- Heuristic approaches
- Combined approaches

Theory-based approaches use a meta-theory of the problem solver's own behaviour for credit- and blame-assignment. These approaches are still rather scarce to date. The RAPTER- and the META-AQUA-system both are examples of this meta-reasoning approach (cf. [CF94]). For example, given the two goals of inspecting arms and refueling the RAPTER-system may try to achieve the former first with the result of arriving late for refueling. In such a situation it constructs a reasoning chain like the following one (cf. [CF94]): failure to refuel *caused-by* failure to synchronize arrival at the fuel station with station's opening hours *caused-by* ... *caused-by* use of arbitrary-choice strategy to decide execution order *caused-by* absence of static ordering constraints in method-1 of prep-journey.

[78] Such an activity would be interpreted as a verification stage by an outside observer.

[79] Even if the application was successful, it may be possible to identify incorrectly used knowledge, for example, if the operator constructed a whole inference chain any elementary step (e.g., resolution) which is not connected with the goal can be regarded as a mistake.

Heuristic approaches are much more common. The *Bucket Brigade Algorithm* is probably the best known approach of this kind. It seems that AM and EURISKO use a similar approach, however the available literature does not allow a final statement about this point. A more refined heuristic of this kind is the heuristic of *minimal loss of confidence* used in the MOBAL-system. As we described these already in section 6.6.4.1, we will not further discuss them.

Combined approaches integrate the use of a theory with heuristic decisions. The system FAILSAFE-2 nicely illustrates this point (cf. [BM94]). This program uses a so-called *impossibility theory* for explaining what is wrong. For example, if the goal is to put block A on top of block B and block A is on the table, then this theory yields the result that the mistake in the current situation is that block A is on the table. Consequently, any situation which is not identical to the goal situation is in error. Therefore, this system uses different criteria for detecting erroneous situations (see page 120). Blame-assignment is finally made by a heuristic saying that the operator application that made the erroneous condition come true (i.e., block A on the table) should be blamed.

After the causes for the reasoning problems have been identified adequate tasks must be generated for repairing them. Usually, this is done by using the causes as an index to a set of tasks. Sometimes it is not possible to identify well-defined reasons for a problem. In these situations the remedies are rather general, too. For example, if a local maximum has been encountered, then try something totally different, is a heuristic which has already been used (among many others) in the EURISKO-system (cf. [Len83a]).

Besides repairing the current solution it is also possible to learn from the causes of the failure how to avoid similar ones in the future (learning from failure) and how to converge to the correct result with less effort (learning from success). We will return to this topic in section 6.7.6.

6.7.5.3 Update Ratings

The information produced in the previous two phases can now be used for updating the ratings (i.e., validity, interestingness) of the available information. In principle, this can also be regarded as learning, as it will influence future behaviour. As many different adaptations will usually be performed in this phase, we restrict ourselves to a few examples.

- The confidence in a rule used in a successful inference chain may be increased.
- The utility of a successful operator may be increased.
- The interestingness of a concept which is mentioned in a newly found conjecture is increased.
- The cost of an operator may be corrected according to the resources consumed in the current application.

Similarly, the estimates may be decreased upon negative results.

Instead of adapting simple numerical estimates it is also possible to characterize more clearly the situations in which an operator will be successful. However, this leads us directly to the topic of the following section.

6.7.6 Learning to Control

The information generated in the credit/blame-assignment phase cannot only be used for repairing mistakes, but for improving the control component as well. This acquisition of control knowledge (opposed to the acquisition of domain knowledge) is the topic of this section.

Usually, three different approaches to learning are distinguished:

- On-line learning
- Off-line learning

- Discovery

In on-line learning control knowledge is acquired during solving a single problem, i.e., the acquired knowledge can already be used for speeding up the solution of this problem. Contrary to this, in off-line learning the knowledge is acquired *after* the problem is solved, i.e., it can only be used for solving subsequent problems. In discovery the generation of control knowledge is not based on problem solving traces, but more generally the system tries to establish relations between control decisions and the overall system behaviour.

The control model employed has the advantage that it easily supports all three kinds of learning, as they only seem to be a matter of perspective in this model. For example, the acquisition of control knowledge after an operator application can be regarded as off-line learning on the abstraction level on which the operator solved its problem. However, viewed from a higher level of the task hierarchy it may as well be regarded as on-line learning. Moreover, if the task was not generated for solving a given problem, but for providing a basis for learning control knowledge, it can also be interpreted as discovery.

Most approaches to the acquisition of control knowledge are based on repeating successful actions and avoiding unsuccessful actions. As we saw in chapter 3 this is very similar to the behaviour human subjects exhibit. A first approach to the formalization of this idea has been made by Lenat (cf. [Len83b, Len82b]) with his *theory of heuristics*. There he defined *heuristics* as "compiled hindsight". According to this theory heuristics are useful, because the function *appropriateness*(action, situation) is a continuous function in both variables (cf. [Len83b, p. 263]). Naturally, this is only a rough approximation of reality. Consequently, Lenat introduced the so-called *second order corrections* stating that the space of task domains is inherently quantized and therefore *appropriateness* is a piece-wise continuous function. However, according to this theory it is useful in a complex, knowledge-rich, incompletely-understood domain to behave as if *appropriateness* would be continuous.

Similar to credit/blame-assignment three basic approaches to the generation of heuristics can be distinguished: heuristic, theory based, and combined. A theory based approach is employed in the RAPTER-system. After performing a theory based blame assignment it simply uses the derived cause for recalling an adequate repair strategy for its control knowledge. Heuristic approaches have been extensively studied in the EURISKO-system. An instance of a combined approach is the FAILSAFE-2-system.

Two basic types of heuristics can be distinguished:

- Plausible move generators (Generalizations of successes)
- Pruning heuristics (Generalizations of failures)

Usually, failures are much more common than successes. Consequently, most heuristics usually learned are of the second kind. This is especially true for systems performing on-line learning, as during problem solving the only successes are the completion of the goal or a subgoal. On the other hand it is rather simple to generate additional opportunities for learning when doing learning from failures, by choosing overly general failure conditions. This approach is used in the FAILSAFE-2-system.

The FAILSAFE-2-system ([BM94]) is a forward-chaining planner. All control knowledge used in the system is knowledge learned during problem solving. Two types of failures are distinguished: over- and underconstrained search. The following four indicators for underconstrained search are used:

1. No operators apply.[80]
2. All operators censored.

[80] This state is regarded as under-constrained, because a perfect problem solver would not consider paths that lead to such situations.

3. A protected goal is violated.

4. Too many states have been expanded without achieving the goal.

Similarly, the search is regarded as being overconstrained if either no state can be expanded or too many states have been expanded without achieving the goal. If any of these situations is encountered, blame-assignment, using the combined approach described in the previous section, is used for identifying the problematic state.

The FAILSAFE-2-system acquires control knowledge in several forms. The most important type is the acquisition of censors, i.e., preconditions to the applicability of an operator. These are learned if the search appears underconstrained. The censor is generated by performing goal regression using the description of the operator that lead to the blamed state and the failure identified by the impossibility.[81] If the search appears over-constrained, censors may be relaxed, i.e., even censored operators may be applied. If this relaxation leads to the goal, an exception to the censor is generated by summing up all preconditions of the operators that lead from the censored state to the goal. This sequence of operator applications is also recorded as a macro. Additionally, the system can learn goal-ordering rules, if it recognizes that for achieving a subgoal it is useful to destroy an earlier achieved subgoal. It learns a rule stating that goals of the current form should be achieved prior to achieving goals of the other form.

In general, we can identify the following types of control knowledge, that can be acquired by a problem solver:

- Preconditions for operators
- New operators
- Strategies
- Preference criteria

Preconditions for operators. The censors introduced by the FAILSAFE-2-system are examples of this approach. They are added for avoiding fruitless search or protecting useful aspects of a situation. A more general approach is suggested by the *theory of heuristics*: preconditions should be modified for increasing the usefulness[82] of an operator. This also includes the modification of preconditions for blocking the application of an operator in situations in which it would consume too many resources or can not yield an adequate result.

New operators. The most commonly used approach to the generation of new operators is the formation of macro operators (cf. [BM94]). A first step along this direction is to recognize that some operators may trigger the application of others. EURISKO has been able to form this concept autonomously (see page 97). Moreover, EURISKO has been able to form genuinely new operators (heuristics), e.g., using analogy, generalization, and many other techniques.[83]

Strategies. A strategy may either be acquired as a coherent piece of knowledge or as a set of rules that relate situations with the possible actions that may be taken. *Derivational analogy* is an implementation of the first approach, while EURISKO realized the second.

Preference Criteria. Preference Criteria can be used in three different contexts:

- Selection of knowledge
- Selection of tasks

[81] Basically, the conditions in the failure description that are made true by the operator are replaced by the operators preconditions. An example of a censor is $\mathit{current\text{-}goal}(\mathit{on}\ ?X\ ?Y) \wedge (\mathit{holding}\ ?X) \wedge (\mathit{clear}\ ?Z) \wedge (\neq ?Z\ ?Y)$.

[82] Using the theory of heuristic we can identify the usefulness of an operator (heuristic) with the integral over its appropriateness function (for all situations).

[83] Lenat ([Len83b]) also mentions meta-heuristics for guiding this generation process, e.g., if the depth of the tree formed by the heuristic along a relation R is large, then generalization should be used, otherwise techniques like analogy can be expected to perform better.

- Selection of operator applications

The selection of knowledge is mainly a topic of reindexing, which we covered in section 6.6.2.1. However, from the more general point of view that we can take now, we can see that the adequacy of the retrieved knowledge considerably depends on the current task and situation.[84] Consequently, these should either be used as additional indices or as criteria for ordering the retrieved knowledge. Naturally, such an ordering among the retrieved knowledge depends on the weighting of the various indices. This weighting can also be refined with experience. The ARIES-system (cf. [Hal89, p. 68]) uses such an approach for improving the similarity measure, it uses for selecting adequate source situations for analogical transfer, with experience.

The selection of an operator for solving the current task is also based on a similarity measure, trying to measure the adequacy of the operator for solving the task. Consequently, a similar approach can be used for improving this measure with experience, in addition to adapting the ratings of the operator.

For an optimal selection of tasks the recognition of interdependencies between the various tasks is very important. An example of a technique for learning explicitly such interdependencies are the goal-ordering rules learned by the FAILSAFE-2-system.

An additional important criterion for selecting a task is its expected distance to the goal. This is especially important, if alternative paths to the goal may exist. For search often heuristics are used for estimating goal distance. An important subclass of these heuristics are the effective admissible heuristics, as they lead to the generation of (nearly) optimal solutions, depending on the type of algorithm they are used with. Prieditis ([Pri93]) gives an approach, which is capable of generating such heuristics automatically. His approach is based on exhaustively searching for an adequate abstraction that yields such an estimate. Its implementation was capable of rediscovering several known heuristics and even some previously unknown ones.

Such a generation of heuristics is a typical discovery task. It would be generated as soon as suboptimal problem solving behaviour is recognized.

Another important discovery task for the acquisition of control knowledge is the generation of additional criteria (concepts) for the selection of operators or tasks. Naturally, these criteria would be derived from the basic goals of the problem solver, like efficiency. Consequently, one can identify this approach with the formation of attitudes (cf. section 3.6.4).

As we have seen above, approaches to learning, which rely on a theory, usually require less search than purely data-driven approaches. This is in accordance with the view proposed by Boden (cf. [Bod91]), that a meta-theory of the problem solver (and its capabilities) is very important to creativity. Because the abilities of the problem solver change over time, it is insufficient to provide it with a static meta-theory. Instead it must be able to form one on its own. Some advances have been made along this direction, e.g., Porter and Kibler ([PK86]) have shown that it is possible to learn by observation a STRIPS-like operator description for a given opaque operator (so-called *relational modelling*). According to Bhatnagar and Mostow ([BM94]) it might possible to extract an impossibility theory automatically from such a STRIPS-like operator description.

Naturally, learned control knowledge must be tested and improved with experience, just like any other knowledge. This may either be made with explicit test tasks[85] or with further experience. Additionally, if not enough information is available for the induction of control knowledge this may be acquired in additional tasks. An approach for generating additional problem situations from given ones is described in [PK86].

[84] The influence of the context on the ease with which knowledge is retrieved is well documented for men (cf. [And89a]).

[85] Based on the notion of heuristics as "compiled hindsight" it might be interesting to test the usefulness of the acquired knowledge on the tasks that lead to its generation.

6.8 Memory Organization

In the preceding sections we introduced various requirements the different abilities of a creative problem solver pose on the organization and the contents of its memory. In this section, we will start with a review of the types of knowledge that are necessary or useful for the different reasoning capabilities. Then we will discuss the implications this has on the structure of the knowledge base and on the representation language used. It will turn out, that the results of this analysis are compatible with the model that has been proposed in section 3.2 based on the results of the psychological analysis.

6.8.1 Knowledge Types

Most of the knowledge a problem solver possesses belongs to one of the following three categories[86]:

- Ontology
- General knowledge
- Cases

The *ontology* defines the basic terms used for describing the contents of the knowledge base. While it is usually regarded as being static, we pointed out the need for changes of the representation language in section 6.6.3.2. For making changes to the ontology the basic building blocks of the representation language should themselves be described in the ontology. We will return to this topic in section 6.8.3, when we will discuss requirements on the representation language.

The *general knowledge* can be regarded as an accumulation of facts and rules. However, based on the interrelations of these knowledge items, a partitioning into different theories may be possible. Additionally, it may contain models of certain devices, situations, etc. These may be regarded as instantiations of the theories. We emphasized the importance of models in the context of creative reasoning in sections 6.4.2 and 6.6.4.3.

The various operators together with the description of their range of applicability belong to this type of knowledge, too. This knowledge cannot be separated from the other knowledge, as it may also serve as input to the other operators (e.g., adaptation). For using the operators in this way a structured representation is necessary (cf. [Len83a]). If the operators themselves are opaque or not easily intelligible to the system, models of their behaviour need to be formed, e.g., using relational modelling ([PK86]), prediction sequences ([She93]), etc.

While the general knowledge describes basic principles that hold in the environment, *cases* record concrete experiences. According to Kolodner ([Kol93]), a case consists of a problem specification and the corresponding solution. Optionally, it may contain an outcome, i.e., the result of carrying out the solution in the world. However, some approaches are based on complete problem solving traces, i.e., a record of the inference steps a reasoner carried out, the alternatives it considered, why it chose one alternative over the other, etc. Between these two extremes a whole continuum of representations exists, all of which we will subsume using the term *case*. Cases may be put to many different uses depending on the information they contain:

- Suggest a solution
- Provide information for induction tasks
- Project the outcome of a solution
- Suggest new strategies (e.g., in *derivational analogy*)

[86] We choose the distinction between these categories based on the abstractness of the knowledge.

- Improve existing knowledge

  ⋮

The importance of cases in reasoning is well documented for human experts (cf. [Sch93]). This significance is basically due to the fact that they help to reduce the cognitive load of the reasoner and that they enable him to come to a solution even if a complete theory is missing. According to Kolodner (cf. [Kol93]), these advantages can also be realized using CBR. These characteristics are also important with respect to creativity, as a creative reasoner needs to handle complex tasks (i.e., alleviating its cognitive load enables him to tackle more complex problems) and must be able to derive solutions in domains without a well-developed theory.

Besides the types of knowledge mentioned above, an indexing structure is necessary for making the efficient retrieval of knowledge possible. Additionally, a good indexing structure should also retrieve the most adequate knowledge in a situation, e.g., predictive features should receive especial attention in indexing (cf. section 6.6.2.1). Consequently, the indexing structure itself can be interpreted as encoding knowledge.

Based, on this interpretation indexing can be regarded as an instance of a more general concept: *knowledge about knowledge*. We will call this kind of information *annotations*. In this chapter we identified several important forms of annotations:

Source: Where did the information come from?
If it was derived, the operators and the information used is important.
If it was told by an external source (e.g., the user), then this is also important.

Age: When has the information been entered or derived?

Applications: When has the information last been used (and to what success)?

Usefulness: In what situations is the information believed/known to be useful?

Reliability: How reliable is the information?

Interestingness: How interesting is the item (and consequently other information related to it)?

Criticality: How important is a certain aspect in a match?

⋮

6.8.2 Structure of the Knowledge Base

The knowledge base of any system that has to deal with complex problems can be expected to be huge. Some authors estimate that human experts have at least 50 000 chunks of relevant domain knowledge. The possession of such a wealth of knowledge probably is an important precondition for creativity (cf. [Fis92]). Additionally, what is usually called *common-sense knowledge* seems to play a very important role in creativity, because when adapting problem specifications a realistic evaluation of the consequences of such a change in a larger context must be made. Up to now only one serious attempt to the encoding of *common-sense knowledge* exists: the CYC-knowledge base (cf. [LG90, LGP+90, GL94]).

Due to the large amount of knowledge necessary and the possibility that some parts of the knowledge base may be inconsistent with other parts, a structuring of the knowledge base is necessary. Here, we will discuss two different approaches to the structuring of knowledge bases: modules (cf. [AW94]) and contexts (cf. [GL92]).

The goal of modularization is to partition the knowledge base into consistent parts, while the knowledge contained in different parts taken together may be inconsistent. In [AW94] the authors propose an approach that allows different modules to be active at the same time, while still preserving consistency. Naturally, this requires that the active modules are adequately chosen.

Guha and Lenat ([GL92]) discuss the use of contexts in the CYC-system. Their primary use is to differentiate between different levels of abstraction (e.g., naive physics and more detailed theories). The distinction between different contexts extends into the ontology of the system, e.g., the same predicate may have different signatures in the various contexts. Lifting axioms define the relation between the different contexts. There is one further application of contexts in the CYC-system, which we employed already throughout this study. These are the so-called problem solving contexts (PSC). A PSC contains all the information which is recalled or inferred in a problem solving attempt. Its primary advantage for the problem solver is that it reduces the size of the search space. Basically, a PSC may contain all the types of knowledge which we identified in the previous section.

However, within a PSC the various knowledge items can be further differentiated according to the role they play:

Problem Specification: The specified problem describes the goal that lead to the generation of the PSC and that shall be solved. The other knowledge items that lead to the generation of the problem specification/problem solving task (see section 6.7) may also be important.

Activated Knowledge: Knowledge that is stored elsewhere, e.g., in the rest of the knowledge base or in other PSCs[87] and is regarded as being important with respect to the current problem.

Inferred Knowledge: During problem solving the executed operators will usually infer additional knowledge.

Hypotheses: During the problem solving attempt usually various alternative directions for solving the problem will be explored (e.g., alternative rules, partial designs, search paths, ...). These will compete for forming the basis of the ultimate result.

For holding various alternative hypotheses special precautions must be made, since if they are simply added to the knowledge base the result will not be consistent. Here, a modularization is also not adequate since it must be possible to contrast the various hypotheses for constructing discrimination experiments. Consequently, special representation capabilities are necessary for such hypothesized knowledge. These special measures may be the reason why so few systems take advantage of the benefits the use of several competing hypotheses may provide.

With respect to creativity the use of competing hypothesis can be interpreted as *divergent thinking*, an approach that is widely regarded as being beneficial to creativity.

After a problem is solved (positively or negatively) the important inferred knowledge in the PSC is added to the knowledge base. Additionally, the problem solving trace (or parts of it) may be added to the knowledge base as cases. Thus, knowledge playing the roles mentioned above may be encoded in the knowledge base.

6.8.3 Implications for the Representation Language

Based on the discussion in the preceding sections some basic properties can be identified that a representation language should have for supporting creativity.

A creative problem solver needs to acquire a lot of knowledge. As this knowledge must be efficiently used, a principled organization of the knowledge base must always be preserved. In order to do so, an extension of the representation language may be necessary (cf. section 6.6.3.2). As Wrobel points out ([Wro94]), it can be proved that the introduction of a new concept always leads to an improvement of the quality of the knowledge base if some structural constraints are satisfied.

The requirement of a principled representation is strongly related to what is sometimes called a *tight representation* (cf. [RP93]). In a *tight representation* a syntactic change closely corresponds to

[87] In section 6.7 we discussed the possibilities that a problem may lead to the generation of several alternative problem solving tasks and that a problem during problem solving may lead to the recursive generation of problem solving tasks. Consequently, PSCs may be recursively embedded in other PSCs or may introduce inferred knowledge from a related PSC.

a *meaningful* semantic change. As Lenat and Brown ([LB84]) point out this is very important for the success of the discovery process, as discovery steps are essentially of syntactic nature. Therefore, a *tight representation* must always be enforced. This can be done by introducing additional concepts, slots, etc. and using these additional representational units for reorganizing the available knowledge. As Haase points out (cf. [RP93]), it is important that such an extension of the ontology has well defined beginning and end. Further, the newly introduced representational units must not be principally different from those initially available.

If such an extension shall not be based on a pure trial-and-error approach, then it is necessary that reasoning about the various possibilities is possible. An important prerequisite for this is that the basic concepts of the representation language are described in the representation language. An early approach to this problem was the representation language language (RLL) (cf. [GL80]). Here, slots, inheritance-modes, functions, etc. are themselves represented as concepts in the system and can consequently be changed. CycL (the representation language of the CYC-system) also possesses mechanisms for meta-level statements (e.g., `allGenls` relates a concept with its generalizations).

A creative problem solver must be able to extend its problem solving knowledge, as well as its problem solving capabilities. Consequently, it must be able to extend its set of operators. As we discussed in section 6.7.5.2 the ability of meta-level reasoning is also very helpful for improving the set of available operators.

Meta-level reasoning is simplified, if the meaning of the represented knowledge is clear. Therefore, a declarative representation language should be preferred. Further, both in theoretical and practical work a certain consensus exists that a declarative language is best suited as the underlying representation formalism for constructing large knowledge bases. As we mentioned above the representation of uncertain knowledge and its degree of certainty is important. This can also be integrated into a declarative formalism (cf. [Hal94]). A more difficult problem is the representation and reasoning with potentially contradictive knowledge.

The importance of meta-level representations for creativity is also emphasized by Boden ([Bod91]) from a psychological point of view.

Chapter 7

Discussion

In this chapter, we will review the results of our study and will discuss some aspects that are related to the context in which creativity usually emerges. We will conclude this chapter with an overview of interesting research problems that have been uncovered in this study.

7.1 Criteria for Creativity

In chapter 4, we did already discuss the characteristics of creative achievements. The two basic criteria for a creative product are that it must be *novel* and *problem solving*. With respect to novelty we further distinguished between the non-standard instantiation of an existing pattern and the invention of a new pattern. The invention of a new pattern is usually regarded as being more creative than making an unusual instantiation. Naturally, this degree of creativity can be refined in various dimensions. For example, even a new pattern will usually contain some parts that are similar to already known patterns.

A different dimension is that of the comprehensiveness of the patterns. For example, a new design for a chair is usually judged less creative than a major scientific achievement that changes the way we interpret the world. An example of the latter is the theory of relativity, which redefined the meaning of fundamental terms like mass, time, and energy (cf. [Kuh73, p. 114]). Consequently, we think that creativity is most adequately regarded as a multidimensional continuum. Some authors simplify this continuum into a hierarchy of levels (e.g., Taylor distinguishes five levels of creativity, cf. [Ulm68, p. 53]).

Judging creativity has the additional complexity that in principle it must be judged with respect to the problem solver, as a pattern may be new to the problem solver, but well-known to an outside observer, or vice versa. However, creativity is usually judged by the environment. We will return to this problem when we will discuss the change of problem specifications.

This dichotomy between the frame of reference of the problem solver and that of the outside observer is one reason why we do not expect to see any programs widely accepted as being creative soon. Current AI-systems have little knowledge compared to humans, so that even if they produce something novel with respect to their own frame of reference, this will probably be well-known to humans and, consequently, not be accepted as being creative.

7.2 The Nature of Creativity

Usually one does not decide a priori to generate a creative solution. Instead, one tries to find an acceptable solution and whether the solution is creative or not is a quality which will be assigned to it a posteriori. However, certain steps (e.g., restructuring) are generally regarded as being essential to creativity, although they do not guarantee creative results.

But, as our review of psychological literature showed, there is no single ability which is decisive creative results. Instead, creativity usually arises through the constructive interaction of several abilities together with the proper knowledge. Among these abilities, standard (i.e. non-creative) problem solving abilities can be regarded as a baseline for a creative problem solver, which is in accordance with the *boundary value hypothesis* (cf. page 8). However, in some situations these abilities need to be complemented by non-standard abilities. Of these non-standard abilities especially the importance of *restructuring* is emphasized. This term subsumes such abilities as *modification of problem specifications* or *reformulation*. These abilities are widely accepted as being important. However, to date they are implemented in only a few systems.

An approach to restructuring which would be rather simple to implement arises in the context of concept formation and conceptual clustering. Usually, these tasks are either performed step-wise (i.e., the results are adapted for each new example) or holistic (i.e., all examples are available at the same time and are taken into account simultaneously). In the step-wise scenario, an example for a restructuring step would be to use all examples available after a number of incremental adaptation steps and perform a holistic characterization of these. But no system we know of employs this approach.

A further characteristic of creativity is that the utility of the abilities and knowledge items available to the reasoner cannot be assessed in advance.[1] Consequently, the lack of some knowledge may even be advantageous. We discussed this based on the example of Hans Krebs (cf. page 10). Some knowledge he possessed was vitally important for his work. At the same time he also had some real benefits from *not possessing* certain knowledge. That is, for achieving a creative solution it is very important that the personal preferences, abilities, and knowledge fit with the requirements of the problem. The difficulty of deciding what knowledge, abilities, and attitudes are helpful or hindering in achieving a creative result (or any result at all) sheds some light on creativity as a social process.

Usually, complex and important problems (e.g., scientific problems) are tackled independently by various people. Each of them tries to get to a solution according to its attitudes using his own abilities and knowledge. Which of the approaches employed is best suited for solving the problem and, consequently, which competitor will be successful can only be decided a posteriori.

This problem solving approach of using different methods simultaneously in the hope that one of them will be successful can be implemented using multi-agent systems. From this point of view, the TEAMWORK-approach (cf. [AD93]) can be regarded as a successful implementation of this idea. In the TEAMWORK-method several experts tackle the same problem using different strategies (e.g., different critical pair selection heuristics). Intermediate results are compared regularly and the best intermediate results form the common basis for the next cycle. Here, the same phenomenon turns up: which expert achieves the best intermediate result can only be judged a posteriori; indeed, this may change from one cycle to the next. Moreover, experiments showed that sometimes contributions from various experts are necessary to achieve the final result (e.g., the proof of a theorem).

7.3 Requirements on Creative Reasoners

In this section, we try to answer the central question of this study:

> What qualities does a problem solver (be it human or artificial) need in order to be creative?

7.3.1 Abilities and Knowledge

Throughout this analysis, we emphasized that a creative problem solver needs a large range of different abilities (operators). Partially, this is due to the complex tasks a creative reasoner needs

[1] This is mainly due to the fact that either at the beginning no way of solving the problem is seen, or during problem solving a restructuring is made that changes the utility of the abilities and knowledge items.

to solve. Consequently, it needs various different abilities that work together in order to achieve a goal (e.g., *problem recognition*, *association*, *transfer*, and *mental simulation*). However, the main reason that makes this large range of abilities necessary is that if one problem solving method proves inadequate, a different method should be available. Due to this reason, a creative reasoner should have several alternative approaches available for executing a single operator like induction. Below, we will use the term *capability* for denoting any approach for implementing an operator.

Different capabilities may complement each other in two different ways:

Fall back: One capability may be used, if the application of the other is not successful.

Stage setting: One capability may provide information that is required for the successful application of the other.

A good example of the first interaction is the OCCAM-system ([Paz93]). It possesses three components for generating causal explanations: EBL, TDL, and SBL. Of these components, a more data-driven component is only used, if the more knowledge-based approach is unsuccessful (cf. page 83). The second interaction is well illustrated by the IDS-system (cf. [NL93]). This system also possesses three basic capabilities: taxonomy formation, qualitative discovery, and quantitative discovery. Taxonomy formation generates abstract state descriptions. Qualitative discovery then establishes generalized transition conditions between these states. Finally, quantitative discovery searches for numerical relations between quantities in different states connected by generalized transition conditions.

In an integrated system with a large number of capabilities usually a vast number of such relations may hold between the various abilities.

An ability that is believed to be of central importance for some types of creative problem solving (especially *creative design*) is the *modification of problem specifications*. This is due to the fact that often in creative problem solving the problem is initially insufficiently defined. In these cases, it cannot be easily decided what constitutes an acceptable solution and what does not. In particular, it may be necessary to drop some initial restrictions on the problem solution. Even if humans do this, it is often not accepted, as we illustrated using the example of the Sydney Opera House (cf. page 7). If computers do so, it is even more unlikely that this is accepted (cf. [Bod91, p. 281]). For example, Boden states that EURISKO was accused of cheating, due to the unusual fleets it generated for winning the TCS game (cf. [Bod91, p. 110]), "but creatively breaking the rules, or even bending them, could always be called 'cheating'".

Thus, if a computer program tries to generate creative solutions that involve the modification of problem specifications, it is very important that the program is able to explain why modifications had to be made.[2] We think that our decision to base our model on human behaviour is advantageous in this context, as the behaviour of a system based on this model will consequently be better understandable for a human.

For making acceptable modifications a large amount of background knowledge about the problem solving task is needed. It must be decided what consequences such a modification has for the user(is such a modification acceptable for him (e.g., does it exceed his level of authority in a business setting), does it solve his underlying problem, ...), on other related problems(does it invalidate any solutions already found for other problems), etc.[3] This problem gets additionally complicated by the fact that the full range of possible modifications and their consequences will not be initially given to the system, as this would require to predetermine all possible modifications. Instead, the system

[2]However, not each modification must be explainable a priori. "Otherwise no solution could be generated" is perfectly acceptable as a justification. Consequently, the system may make modifications to the problem specification simply to see what will happen.

[3]Current systems with the capability of modifying problem specifications take a simpler approach. In CYCLOPS (cf. [Nav91]) the possible modifications along with the corresponding changes to the evaluation of the result are explicitly encoded in the system. In CREATIVE JULIA (cf. [KP90]) constraints on the solution are divided into primary and secondary constraints and the system prefers changes to secondary constraints.

must be able to make realistic assumptions about this information. For doing so, a large amount of common-sense knowledge is needed. The only project that tries to provide such capabilities is the CYC-project (cf. [GL94]). However, it is neither clear whether this project will succeed with respect to its own goals nor whether it will solve the problem outlined above.

Due to the complications connected with modifications of problem specifications, domains where problem specifications are initially more exactly defined and, consequently, modifications of problem specifications are less usual, are better suited for creative computer systems. We think that technical engineering domains are domains of this kind. But even, if no modifications of problem specifications are made a creative problem solver should have a large amount of knowledge available, as it may provide the problem solver with alternatives in reasoning if it gets stuck. For example, the larger the range of source situations, the more possibilities for using the *analogy*-operator exist.

7.3.2 Meta-abilities

We used the term *meta-abilities* to denote abilities/characteristics that can be combined with any of the usual abilities.

We could identify two important meta-abilities:

- Heuristic reasoning
- Learning

Heuristic reasoning. Often people use error-prone methods for achieving a result.[4] In AI, this approach is usually avoided, if an exact method exists. However, we think that this approach is very important in order to deal with complex problems, as it is often much easier to produce a preliminary solution and repair it afterwards than to produce a correct solution in the first attempt. Further, these heuristics often produce a correct solution despite the fact that they are not logically sound. The fact that they enable a reasoner to solve problems with less effort allows the reasoner to solve more complex problems. This is important, as problems that demand for a creative solution usually are rather complex.

Learning. Problems which are complex enough that their solutions will be commonly accepted as being creative will usually be too difficult to be solved at first try. Consequently, a creative reasoner needs to acquire competence for solving its problems. This competence cannot be available at the outset, as the problem demands (by definition) a novel solution.

The acquisition of competence extends into several dimensions:

- Reorganization of the memory (appropriate indexing, activation of useful knowledge)
- Refinement of the problem description
- Acquisition of necessary domain knowledge
- Acquisition of appropriate operators
- Acquisition of appropriate control knowledge (strategies, evaluation criteria, etc.)

According to the psychological model of chapter 3, the acquisition of competence should be predominant in the preparation stage. Based on the psychological part of our study we came to the conclusion that in this stage *functional autonomy* is especially important as the acquisition of competence essentially is a discovery task. Many different activities may contribute to the acquisition of competence. For example:

- Exploration of the range of acceptable solutions by tentative modifications of the problem description

[4] We discussed this in section 6.5.1 "inference".

- Analysis of initial, unsuccessful solution attempts
- Solution of restricted problems[5]

This leads us to our central hypothesis:

> *Creative problem solving can be understood as a process in which problem solving and discovery are tightly integrated.*

One can interpret this statement as saying that the abilities of problem solving architectures can be greatly enhanced by the addition of discovery capabilities.

But it can also be interpreted the other way round: "discovery architectures can be greatly enhanced by the addition of problem solving capabilities and by providing them with external problems the range of their discoveries may be greatly extended". We believe that the lack of confrontation with external problems is one major reason for the ultimate demise of such programs as AM. The task of AM was to satisfy its curiosity according to its preprogrammed rules and in the end it had done so. Consequently, it could not find any more interesting tasks. This is analogous to mathematics (AM's domain). Again and again problems have been posed from other domains (especially from physics) leading to new mathematical questions, which sometimes even lead to the discovery of completely new disciplines in mathematics. Without this kind of input mathematics would never have achieved the state it is in today. Comparable stimuli have been missing in the AM-program.

7.3.3 Evaluation

Some characteristics of creative reasoners (with respect to evaluation) are:

1. They exhibit field-independence (i.e., they are not fooled by surrounding stimuli).
2. They possess a large number of criteria for judging a result.
3. They show tolerance with respect to diverging details.
4. They are able to recognize emergent value.

We think that the first characteristic is related with creativity as a social process. This criterion implies that a creative reasoner evaluates its results with respect to its own frame of reference. Consequently, different reasoners will probably make different attempts, of which probably only one will succeed.

The second and the third criterion are tightly connected. If a reasoner is tolerant w.r.t. some criteria, then a critical selection is only possible if the reasoner possesses a large number of criteria. The tolerance w.r.t. diverging details that creative reasoners exhibit can also be interpreted as a relaxation of problem constraints (cf. *modification of problem specifications*).

The ability to recognize emergent value can be interpreted as a counterpart to the problem recognition operator we proposed in our model. It can also be interpreted as a form of learning as the criteria found are subsequently used for judging other alternatives. The ability to recognize emergent value may be a source for the large number of criteria a creative problem solver possesses.

7.3.4 Control

In section 2.3.2 we identified as an important quality of creative people that they are willing to devote much effort to a solution approach that they evaluated as promising. We believe that this is an important characteristic of any creative reasoner as the problems that must be solved creatively

[5] Porter and Kibler ([PK86]) already demonstrated the usefulness of generating and solving modified problems for acquiring competence in problem solving.

are usually rather difficult. However, at the same time it is generally believed that creative reasoner must be capable of *divergent thinking*, i.e., they are able to produce and evaluate many different alternatives. These two characteristics are not at odds w.r.t. each other, as it may seem. Instead, this implies that a creative reasoner evaluates many different alternatives, settles for one of those, and then follows this direction until he arrives at a solution.

A further characteristic of creative reasoners is that they prefer complex stimuli, i.e., they are attracted by more complex stimuli than ordinary reasoners and capable of handling them.

Above, we did already mention a major result of our comparison of existing AI systems and the requirements of creative reasoners: an integration of problem solving capabilities and discovery capabilities along with *functional autonomy* is necessary. Naturally, these requirements have also implications on the control of the creative process. We did try to consider these in our proposal for a control component in section 6.7.

7.4 Conclusion

In the course of this study (especially chapter 6), we developed a model for a creative problem solver. We expect this model to be rather successful if implemented, as it bears some deep similarities with the models developed in psychological research for explaining human creative behaviour. Additionally, our model is compatible with some approaches in computer science for modelling creative behaviour. Specifically, these are *automated discovery* (cf. [SHM91, KS88, Len83b]) and *creative design* (cf. [KP90, KW93, McL92]).

The central characteristics of our model are:

- A tight integration of a large number of abilities is proposed.
- Aspects from automated discovery (e.g. functional autonomy) and problem solving capabilities are tightly coupled.

The model we proposed in this study must not be mistaken as a program design. It is more adequately described as a research project, as our model is still hypothetical and, consequently, needs thorough validation. When transforming this model as a whole or certain parts of it into a program design, some modifications will be necessary. This is especially obvious with respect to the way we described the operators. For example, while we discussed the recall of an analogy source (section 6.4.1, "Association and Recall") and the elaboration of an analogy (section 6.5.2, "Transfer") separately, in an implementation these two will probably be tightly integrated, as many interactions between them exist that usually can be exploited to good advantage.

A major problem that we expect for an implementation of our model is that a large amount of knowledge is needed for any system that is expected to be creative in a realistic setting. Especially, for the adaptation of problem specifications much background knowledge is needed.

Consequently, in the near future autonomous creative systems are rather unlikely. Instead, systems for supporting human creativity are more realistic. Based on this assumption it may seem inadequate to restrict this study to abilities necessary for autonomous creative behaviour. However, we believe that the same abilities are necessary for support systems, although at a lesser degree of perfection. We will not discuss the additional abilities necessary for supporting creativity in order to keep this study at a reasonable size. Relevant material can be found elsewhere (e.g., [Fis92]). We believe that our decision to base this study on psychological models is advantageous for the construction of such systems, because they should behave in a way intelligible to the user.

7.5 Opportunities for Future Research

In the course of our analysis, it was possible to discover some deficiencies of AI approaches by contrasting their abilities with those exhibited by humans. In this section we will summarize these

deficiencies. They are also good opportunities for further research in AI. For the purpose of easy readability, we group these research problems according to the operator to which they relate.

Problem recognition. If a problem solver moves to a new field, its abilities to recognize problems early need to be adapted to peculiarities of this domain. For example, if someone computes salaries he will soon recognize that intermediate numbers exceeding a certain threshold point to problems, thereby enabling him to catch errors early on. We expect that such domain-specific problem criteria can be generated by a combination of blame-assignment, induction, and domain-specific instantiation.

Association and Recall. With respect to recall we pointed out the importance of semantic categories for the recall of information. However, all approaches using semantic categories we know of use hard-coded categories. We think that similar techniques as those employed for problem recognition can be used for establishing new (domain-specific) semantic categories.

To our knowledge no approach to association uses the full range of association keys described in psychology. Especially contiguity, part-whole relations, and similarity/contrast are usually not used for association. The secondary association laws (vivacity, recentness, and frequent repetitions) generally are also not used in AI systems.

Transfer. Humans often generate prototypical situations specifically for the purpose of transferring knowledge from them. Contrary to this, in AI analogy is only used with existing situations.

Induction. As we discussed in section 6.5.3, AI approaches to induction are inferior to approaches used by humans in many ways. The most prominent of these is the ability to make a plausible a priori selection of relevant features. While AI systems usually use all available features as a basis for rule construction, humans preselect features based on the theories they have. The same difference exists with respect to the form of rules.

Adaptation. If a rule does not seem to hold, people are able to change the interpretation of the features that are mentioned in the rule.

Although the problem of generating adequate hypotheses for repairing theories has already received considerable attention, we believe that it still deserves much work.

The merging of elements from different cases is still an open problem (e.g., no system to date is able to merge the concepts `truck` and `house` to yield `mobile home`).

Control. Generating new criteria for judging the appropriateness of a task and evaluating its results is very important when moving to a new domain. Basically, this is very similar to problem recognition and we believe that the same techniques may also be helpful in this context. In particular, the domain specific instantiation of attitudes may be helpful. For example, the basic goal of efficiency, when instantiated in the domain of programming, will at least yield the following two results: efficiency of the program and efficiency of generating the program.

Especially in the context of relaxation of problem constraints, a further problem arises: what does it mean to relax a constraint? For example, the constraint "house shall face south" to "house shall approximately face south" implies the change from a discrete criterion to a continuous criterion. We believe that such transitions can only be performed with the help of adequate background knowledge (i.e., the program must have an understanding what does it mean for a house to face south).

Appendix A

The VEGA Project

Now, we will relate the basic approach used in the VEGA project at the DFKI GmbH (cf. [BHH+92]) with the model proposed in this study. We will begin with a short outline of the VEGA project. Next, we will discuss in some detail differences between the VEGA approach and our model and the consequences this implies. As the VEGA project can also be viewed in the larger context of a *corporate memory*, we will end this appendix with a short discussion of corporate memories and the advantages they may provide for creative problem solving.

A.1 An Outline of the VEGA Project

While our model is based on the assumption that problem solving capabilities, discovery techniques, and the knowledge base should be tightly integrated, the VEGA project is based on a more modular model. The knowledge base and the discovery capabilities are integrated, while problem solving capabilities are located in (external) expert systems.[1] These expert systems can be plugged into the knowledge base provided by the VEGA system, for supplying them with the necessary knowledge.

The task of the VEGA system is the discovery of new knowledge, as well as the improvement of the representation of knowledge already encoded in the knowledge base. Naturally, the discovered knowledge must be validated, before it is used. This aspect is strongly emphasized in the VEGA approach and is envisioned as being tightly coupled with knowledge discovery.

It is expected that the VEGA system will be useful for the construction of new knowledge bases (i.e., during knowledge acquisition) as well as for the maintenance of existing knowledge bases. (These may be altered through the introduction of new knowledge by the expert system (e.g., cases) or by the knowledge engineer.)

The most obvious difference between our model and the VEGA approach is that we propose a tight integration of the problem solving capabilities with the other capabilities of the system, while in the VEGA approach these are only loosely coupled. Obviously, the VEGA approach has considerable advantages (e.g., the modularity of the approach, an improved reusability of the VEGA system across problem domains, etc.). However, we believe that it also has considerable disadvantages. We will return to this point later.

A.2 Relation of Our Model with the VEGA Approach

The VEGA system and our model have an important feature in common: discovery techniques play a central role. In the VEGA system they are also called *exploration techniques*. There, they are

[1] However, problem solving capabilities can also be regarded as discovery techniques, as they lead to the generation of additional cases. These can be integrated into the knowledge base. Indeed, this is the main technique in CBR systems for enriching the content of the knowledge base.

used for improving the knowledge contained in the knowledge base and extending it by inducing additional knowledge. In our model these techniques are used for two purposes:

- For improving the competence of the problem solver
- For restructuring the knowledge base and the current hypotheses

While the latter is very similar to the VEGA approach, the former does not have a correspondence in the VEGA approach, as the problem solvers are not part of the VEGA architecture. This has the further consequence for the VEGA approach, that although the necessary techniques are available, they cannot be directly used in problem solving although they would be helpful. Further, although the improvement of the knowledge base envisioned should improve the competence of the attached problem solvers, this cannot be done in a goal-oriented way, as the problems of the problem solver cannot be used as a basis for selecting further activities in the VEGA system, as it does not know anything about them.[2] However, as the VEGA system may interact with the user, he may provide the system with the necessary goal-orientation. Moreover, the VEGA system can also be regarded as a toolbox (cf. [ABH+95]), where the user may even select the specific operators applied to the available knowledge.

Discovery techniques are used in the VEGA project only in a restricted form, as compared to our model. However, still the same techniques may be useful. Consequently, our discussion of these techniques, their prerequisites and possibilities of improvement may also provide ideas for the VEGA project. Especially in those cases where the same techniques have been discussed in the VEGA project and in this analysis, additional insights may be expected, as both discussions are based on different viewpoints.

If one compares the VEGA system with plugged-in expert systems with our model, one may recognize a strong analogy: as the expert systems use knowledge from the main knowledge base maintained by the VEGA system, their local knowledge bases can be regarded as views on the global knowledge base enriched with knowledge inferred in the course of problem solving. So these local knowledge bases strongly resemble the problem solving contexts we propose in our model.

However, despite this analogy, the different forms of integration of the problem solving capabilities cause a lot of differences between the two models. The advantages of the VEGA approach have already been outlined in the previous section. Now, we will briefly summarize the advantages of our model:

1. By integrating problem solving capabilities into the model, the discovery process can be better focussed on information that can be expected to improve the problem solving capabilities. Additionally, problem solving capabilities may trigger a search for information (cf. section 6.2), leading to the discovery of information that would otherwise not have been found.

2. It has already been pointed out in the VEGA project proposal ([BHH+92]) that complex evolution techniques may be too time-consuming for being applied to the complete knowledge base. In this case, adequate parts of the knowledge base must be identified for applying the discovery algorithms to it. We believe that problem solving contexts are natural and adequate parts of the knowledge base for this task.

3. The context of problem solving provides additional opportunities for the discovery of knowledge (especially control knowledge for the problem solver). This may be further improved if additional problems are automatically generated for learning. That is, the problem solving capabilities may aid in the discovery and not only vice versa.

The tight relation between discovery and problem solving becomes most obvious, if one examines the task of a discovery system like the VEGA system. As the VEGA project proposal ([BHH+92])

[2] Besides, perhaps, the domain: a sub-project of the VEGA project (APPL-KB) defines a knowledge base, which shall be the main application domain of the VEGA system.

defines, a central task of the VEGA system is the discovery of interesting patterns in the knowledge base, where "interesting" is defined[3] as novel, useful, and nontrivial to compute. This definition is rather problematic in the context of the VEGA approach, as usefulness can only be adequately defined with respect to a problem solving task. Here again, the user is needed for providing the system with the necessary guidance.

Markovitch and Scott ([MS93]) examined the problem of the usefulness (utility) of knowledge in more detail. They define the *utility of a change in a knowledge base* as being dependent (among other parameters) on the problem solver and the set of problems it needs to solve. Consequently, if a discovery system searches for additional knowledge without taking into account the actual needs of the problem solver, the knowledge found may even be harmful (see also appendix B).

Despite the many differences between the VEGA system and our model which we emphasized above, there are also many commonalities. For example, both approaches rely on a declarative language and require that a large number of different knowledge types may be represented using this formalism. In the VEGA project the representation language is used for representing:

- task and domain knowledge
- validation goals and exploration hypotheses
- meta-knowledge (e.g., encoding interestingness).

These uses of the representation language are also common in our model. Additionally, we proposed the representation of some other types of knowledge:

- annotations of various kinds (usefulness, criticality)
- operators
- control knowledge (preconditions, goal orderings)
- problem solving traces
- causes for tasks

As the most important uses of the representation language are common to both the VEGA approach and our model, we expect that our discussion of the representation language and representational needs may provide additional ideas for the VEGA project.

A.3 Corporate Memory

The VEGA project may also be regarded in the larger context of a *corporate memory*. The idea behind a *corporate memory* is, that it contains the whole knowledge that exists in a corporation and thus supports all the activities in a company.

This scenario is especially interesting if viewed with creative problem solving in mind. As we pointed out, it is an important problem that constraints on the problem solution can be relaxed if necessary, and that this is done in a controlled way. For doing so, a lot of knowledge is necessary for judging the consequences of such a relaxation. We believe that a *corporate memory* provides this knowledge, as it relates the task to its context. An example will illustrate:

> Suppose an expert system is given the task of scheduling the transport of some goods from point A to point B, such that they do not arrive later than the next day at noon. Relaxing the constraint *at noon* may lead to the generation of the solution: transport the goods from point A to point C and from there to point B. As these trucks are needed anyway the solution would be less costly, however, the goods would arrive only at 2 p.m.

[3] In this study, we defined interestingness differently (probably more adequate for use in the VEGA project), relating interestingness with informational content. Additionally, we identified usefulness as being independent from interestingness and termed it utility (cf. page 124).

If the context of the task is not known to the expert system, then it has no possibility of judging this alternative. However, if sufficient knowledge is provided by a *corporate memory* then this judgement may be possible. For example, if the transportation of goods is the company's job and arrival of the goods until noon is guaranteed, then this probably is a bad idea. However, things are different, if these goods are needed for the company itself. If they are not needed any earlier than 3 p.m. then this constraint relaxation is advantageous and should be made.[4]

As this discussion shows a *corporate memory* may provide the necessary context for judging the relaxation of problem constraints. Consequently, we expect that *corporate memories* will play an important role in the introduction of creative problem solving abilities.

[4]However, there still may be problems the system itself is not aware of. Consequently, the altered problem description should be handed back to the user for confirmation.

Appendix B

The Information Filtering Model

The *information filtering model* was developed by Markovitch and Scott (cf. [MS93]). Contrary to other models of learning systems this model concentrates exclusively on the selection (or filtering) of information during the learning process. Therefore, it is tightly connected with the process of evaluation as it is discussed in this study. This model is based on the concept of *information flow* in a learning system that is outlined in figure B.1.

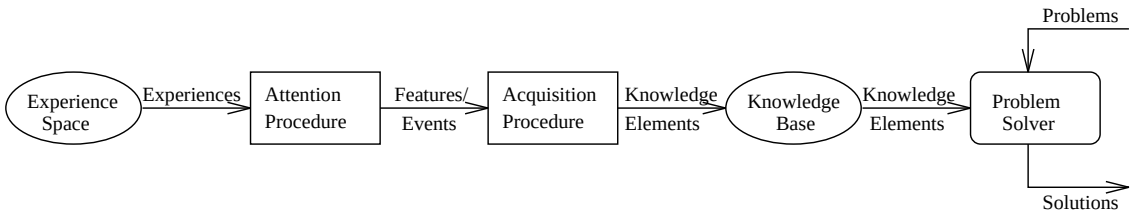

Figure B.1: Information flow in a learning system

Now, we want to give a short overview of the components of this model:

Experience space: The experience space contains all possible experiences. From this space training experiences are selected and provided to the learner. This selection can be made by the user, the learner, or the problem solver.

Attention procedure: The attention procedure selects certain aspects of the experiences (or certain experiences) and passes them on. For example, in a failure-driven learner the attention procedure would eliminate all successful cases.

Acquisition procedure: The acquisition procedure is the learning component in the system. It uses its input to generate changes in the knowledge base. This procedure also transforms (features of) experiences into knowledge.

Problem solver: The problem solver uses knowledge from the knowledge base to generate solutions to problems.

The purpose of information filtering is to permit only data of *positive utility* to flow on to subsequent stages. The utility of data can be assessed on different stages.[1] Data of negative utility

[1] In [MS93] Markovitch and Scott formally define the *utility of a change in the knowledge base*, the *utility of a knowledge element*, and the *utility of a set of training experiences*.

can considerably impair the efficiency of the problem solver as well as the quality of the solutions it produces. However, the utility of information depends on various parameters, e.g., the problem to solve, the current content of the knowledge base, the way the problem solver uses the knowledge base, etc. Information can be filtered on every stage. However, the later the filtering is done the more information is available for assessing its utility. On the other hand, the earlier the filtering is done the less effort is spent on information of negative utility.

Figure B.2 shows the five possible locations for information filters in a learning system.[2] However, most learning systems to date possess only one or two of these filters, if any.[3]

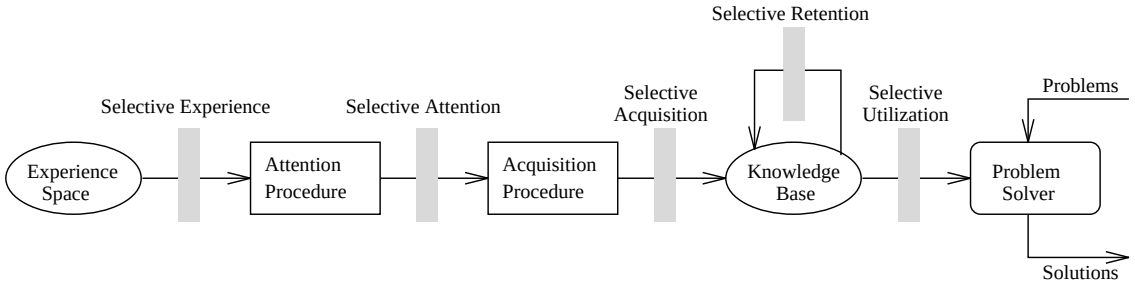

Figure B.2: The information filtering model

The following list gives an overview of the different types of filters:

Selective experience: This filter selects experiences from the set of possible experiences. This can be made either actively (by generating appropriate tasks) or passively (by throwing away experiences of negative utility). Three different approaches to implementing such a filter can be distinguished: error based (learning from mistakes), uncertainty based (a selection can not be made), and based on miscellaneous heuristics (e.g., in AM).

Selective attention: This filter tries to select the set of useful features from the set of all features. This is usually based on some predefined measures or heuristics.

Selective acquisition: This filter selects the knowledge which shall go into the knowledge base. Because no experience with using it is available, the evaluation is usually based on the experience which led to the generation of this knowledge item. For example, in PRODIGY a generated rule is judged based on the time savings it would have provided to the problem solver, if it would have been available during the problem solving experience. Here, filtering for better solutions and for greater efficiency can be distinguished. Filtering for better solutions is most often used in systems which have a classification task.

Selective retention: Contrary to the prior filters this filter can select knowledge based on experience with its utility in problem solving. For example, in PRODIGY the same heuristic is used as in the acquisition filter, but now the evaluation is based on the experiences since acquisition of the rule. Here filtering for better solutions and for greater efficiency can be distinguished, too.

Selective utilization: This filter provides a restricted view on the knowledge base to the problem solver. It is especially important if the utility of knowledge varies greatly as a function of the

[2] The attention and the acquisition procedure also reduce the information content of the experiences. However, they are not regarded as filters as this reduction is not made for eliminating information of negative utility, but only for transforming experiences into knowledge.

[3] According to [MS93], a notable exception from this rule is the PRODIGY-system. It contains experience, attention, acquisition, and retention filters. The only filter missing is an utilization filter. IB4 and MACLEARN each possess three different filters.

problem to be solved. For example, indexing schemes can be regarded as utilization filters, if the indices are selected such that only knowledge which is probably rather useful for problem solving is retrieved (see section 6.4.1.1 for details on the selection of indices).

Often these filters are implemented as precoded heuristics. However, they can also be realized as learning systems. In this case, they are called *secondary learners* (cf. [MS93, p. 142]). Especially, the retention filter is an obvious candidate for implementation as a *secondary learner*. It is often implemented by accumulating usage statistics and eliminating knowledge in the knowledge base, which did not perform well in prior problem solving experiences.

Appendix C

Glossary

This glossary is based on the definition of terms used in *scientific discovery* given in [SL90]. However, some adaptations and additions had to be made due to the larger context they are used in here (e.g., the problem solver may be aware of several different domains).

Ontology[1]**:** The ontology introduces the concepts which are used in the knowledge base. It also gives relations between these concepts, e.g., specialization/generalization, instantiation, etc. It can also include other characteristic information of a concept like the units of measurement, the values attributes can assume, etc.

Problem knowledge: Knowledge which is specific to the current problem. This includes the problem specification and available hypotheses for solving the problem.

Domain knowledge: All knowledge that belongs to the same domain as the current problem. For example, with respect to a problem in electricity all knowledge about electricity belongs to the domain knowledge. This includes the relevant parts of the ontology, earlier problem solving experiences, general laws, etc. The relation between a problem and a domain is mediated by the ontology. However, such a correspondence may be established on different levels of generality. Correspondingly, the relation between a problem and a domain is not unequivocal.

Background knowledge: The background knowledge contains knowledge which is not specific to the current problem. Among other things this may include beliefs about the environment, earlier problem solving experiences, meta-knowledge (e.g., why is the current problem important?). Just like in the domain knowledge different types of knowledge are contained in the background knowledge. Sometimes we will distinguish between *background knowledge for a problem*, i.e., knowledge, which is related to a certain problem and *background knowledge in general*, which includes all the permanent knowledge the problem solver possesses. The background knowledge for a problem differs from the corresponding domain knowledge insofar as it may contain knowledge from other domains, e.g., problem solving experiences in the domain of hydrodynamics, when the current problem belongs to the domain of electricity. The background knowledge in general, together with the problem knowledge and the knowledge about the current environment, makes up for all the knowledge available to the problem solver.

Theory: A theory is a body of knowledge which contains the necessary facts and rules for explaining phenomenons in (a part of) a domain. The domain knowledge may contain several competing theories, e.g., newtonian mechanics and relativistic mechanics.

Model: A model is defined by a description of a situation (experiment, device, etc.) together with the necessary rules for determining the development of the situation (carrying-out of the

[1] This is sometimes also called a taxonomy or a concept hierarchy.

experiment, behaviour of the device). Based on competing theories, alternative models may exist for the same situation. A model can be interpreted as an instantiation of a theory for a concrete situation.

(In)active knowledge: While the previous categories of knowledge were based on the type of the knowledge and its relation to the problem, the distinction between active and inactive knowledge is based on the reasoning process. During the reasoning process knowledge can become activated or deactivated, with the goal that at any moment the knowledge which is necessary for the progress of the problem solving process is active. While Anderson assumes a continuous range of activations, we distinguish only between active and inactive knowledge for the sake of simplicity. At the beginning of the reasoning process only the problem knowledge is activated.

Problem solving context: In principle, a problem solver can work on several problems at the same time. The active knowledge for a single problem forms a *problem solving context*.

Restructuring: In psychology the term 'restructuring' denotes a non-trivial transformation of the knowledge available to the subject. This includes changes to the form and to the contents of the knowledge. In the AI part of this study we distinguish between the construction of knowledge and changes to existing knowledge. There, we use the term *restructuring* only for the latter.

Ability/Operator: Throughout this thesis, we use the terms *ability* and *operator* mostly synonymously. However, we prefer the term *ability* when discussing human behaviour and we use the term *operator* when we discuss the model presented here or computer programs. One should also note that these terms are not used for referring to a single information processing activity, but for a whole group of techniques with similar purpose. Thus, the *operator induction* summarizes all techniques that may be used for performing induction.

Meta-ability: Some abilities or basic approaches can be combined with other abilities. We term such abilities *meta-abilities*. For example, learning is a meta-ability as it can be combined with adaptation for yielding new adaptation methods, it can be combined with inference in order to produce new inference methods (e.g., macros), etc.

Bibliography

[ABH+95] Andreas Abecker, Harold Boley, Knut Hinkelmann, Holger Wache, and Franz Schmalhofer. An Environment for Exploring and Validating Declarative Knowledge. DFKI Technical Memo TM-95-03, DFKI GmbH, September 1995.

[AD93] Jürgen Avenhaus and Jörg Denzinger. Distributing equitional theorem proving. In Claude Kirchner, editor, *Rewriting Techniques and Applications, 5th International Conference, RTA-93, Montreal, Canada, June 1993*, volume 690 of *Lecture Notes in Computer Science*, pages 62–76. Springer Verlag, Berlin, 1993.

[And85] John R. Anderson. *Cognitive Psychology and Its Implications.* A Series of Books in Psychology. W. H. Freeman and Company, 2nd edition, 1985.

[And89a] John R. Anderson. *Kognitive Psychologie.* Spektrum der Wissenschaft Verlagsgesellschaft, Heidelberg, 2nd edition, 1989. This is a german translation of [And85].

[And89b] John R. Anderson. A theory of the origins of human knowledge. *Artificial Intelligence*, 40:313–351, 1989.

[Ave91] Jürgen Avenhaus. Reduktionssysteme 1. Skript zur Vorlesung, Universität Kaiserslautern, Fachbereich Informatik, Postfach 3049, 67653 Kaiserslautern, 1991.

[AW94] Grigoris Antoniou and Ipke Wachsmuth. Structuring and modules for knowledge bases: Motivation for a new model. *Knowledge-Based Systems*, 7(1):49–51, March 1994.

[BGP94] Sambasiva R. Bhatta, Ashok K. Goel, and Sattiriaja V. Prabhakar. Innovation in analogical design: A model-based approach. In *Proceedings of the 3rd International Conference on Artificial Intelligence in Design (AID-94), Aug. 15–18, Lausanne, Switzerland*, 1994. (to appear).

[BHH+92] Harold Boley, Philipp Hanschke, Knut Hinkelmann, Manfred Meyer, and Michael M. Richter. Vega — knowledge validation and exploration by global analysis. Project proposal, German Research Center for Artificial Intelligence (DFKI), 1992.

[BM94] Neeraj Bhatnagar and Jack Mostow. On-line learning from search failures. *Machine Learning*, 15(1):69–117, April 1994.

[Bod91] Margaret A. Boden. *The Creative Mind: Myths and Mechanisms.* Basic Books, 1991. 303 pages. First published in Great Britain by George Weidenfeld and Nicolson Ltd. in 1990.

[Bod93] Margaret A. Boden. The logic of creativity. In D. Miller, editor, *Proceedings of the 1993 International Symposium on Logic Programming*, pages 3–11, Cambridge, MA, 1993. MIT Press.

[Bod94] Margaret A. Boden. Agents and creativity. *Communications of the ACM*, 37(7):117–121, July 1994.

[CCH91] Yandong Cai, Nick Cercone, and Jiawei Han. Attribute-oriented induction in relational databases. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, chapter 12, pages 213–218. AAAI Press/ The MIT Press, Menlo Park, CA, 1991.

[CF94] Michael T. Cox and Michael Freed. Using knowledge of cognitive behaviour to learn from failure. In J. W. Braham and G. E. Lasker, editors, *Proceedings of the 7th International Conference on Systems Research, Informatics, and Cybernetics; Baden-Baden, Germany, August '94*, pages 142–147, 1994.

[CL95] Karin Causse and Jacques Lebbe. Modeling identification strategies using the MCC methodology. In *Proceedings of the MLnet Sponsored Familiarization Workshop, Knowledge Level Modelling and Machine Learning, Heraklion, Crete, Greece, April 28–29, 1995*, pages 1–15, 1995.

[Cle88] John Clement. Observed methods for generating analogies in scientific problem solving. *Cognitive Science*, 12:563–586, 1988.

[Cox94] Michael T. Cox. Machines that forget: Learning from retrieval failure of mis-indexed explanations. In *Proceedings of the 16th Annual Conference of the Cognitive Science Society, Hillsdale, NJ.* Lawrence Erlbaum Associates, 1994.

[CW91] Keith C. C. Chang and Andrew K. C. Wong. A statistical technique for extracting classificatory knowledge from databases. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, chapter 6, pages 107–123. AAAI Press/ The MIT Press, Menlo Park, CA, 1991.

[DHS87] Friedrich Dorsch, Hartmut Hacker, and Kurt-Hermann Stapf. *Psychologisches Wörterbuch.* Verlag Hans Huber, Bern, 11 edition, 1987.

[Die86] Thomas G. Dietterich. Learning at the knowledge level. *Machine Learning*, 1:287–315, 1986.

[dK93] Johan de Kleer. A view on qualitative physics. *Artificial Intelligence*, 59(1/2):105–114, february 1993. (Special Volume: AI in Perspective).

[ECJS94] Ernest A. Edmonds, Linda Candy, Rachel Jones, and Basel Scafi. Support for collaborative design: Agents and emergence. *Communications of the ACM*, 37(7):41–47, July 1994.

[ES90] Renée Elio and Peternela B. Scharf. Modeling novice-to-expert shifts in problem-solving strategy and knowledge organization. *Cognitive Science*, 14:579–639, 1990.

[Etz88] Oren Etzioni. Hypothesis filtering: A practical approach to reliable learning. In John Laird, editor, *Proceedings of the 5th International Conference on Machine Learning, Ann Arbor, MI*, pages 416–429. Morgan Kaufmann Publishers, 1988.

[Fis87] Douglas H. Fisher. Knowledge acquisition via incremental clustering. *Machine Learning*, 2:139–172, 1987.

[Fis89] Douglas H. Fisher. Noise-tolerant conceptual clustering. In N. S. Sridharan, editor, *IJCAI-89, Proceedings of the 11th International Joint Conference on AI, August 20–25, 1989, Detroit Michigan, USA*, volume 1, pages 825–830. Morgan Kaufmann Publishers, 1989.

[Fis92] Gerd Fischer. Creativity enhancing design environments. In John S. Gero and Mary Lou Maher, editors, *Modeling creativity and knowledge-based creative design*, chapter 10, pages 235–258. Lawrence Erlbaum Associates; Hillsdale, New Jersey, 1992.

[FP91] Douglas H. Fisher and Michael J. Pazzani. Theory-guided concept formation. In Douglas H. Fisher, Michael J. Pazzani, and Pat Langley, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, chapter 6, pages 165–177. Morgan Kaufmann Publishers, 1991.

[GL80] Russell Greiner and Douglas B. Lenat. A representation language language. In *Proceedings of the First Annual National Conference on Artificial Intelligence at Stanford University, August 18 to 21*, pages 165–169, 1980.

[GL92] Ramanathan V. Guha and Douglas B. Lenat. Language, representation, and contexts. *Journal of Information Processing*, 15(3):340–349, 1992.

[GL94] Ramanathan V. Guha and Douglas B. Lenat. Enabling agents to work together. *Communications of the ACM*, 37(7):127–142, July 1994.

[Gos92] Thomas Goseberg. ACT, PUPS and spreading activation. In S. Wess, K.D. Althoff, F. Maurer, J. Paulokat, R. Praeger, and O. Wendel, editors, *Fallbasiertes Schließen — eine Übersicht (Band I–III)*, volume SWP-92-08 of *SEKI WORKING PAPERS (SFB)*, chapter 2, pages 13–28. Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 67653 Kaiserslautern, 1992.

[Haa89] Kenneth Haase. Automated discovery. In Richard Forsyth, editor, *Machine Learning: Principles and Techniques*, chapter 7, pages 127–155. Chapman and Hall, 1989.

[Hal88] Robert J. Hall. Learning by failing to explain: Using partial explanations to learn in incomplete or intractable domains. *Machine Learning*, 3(1):45–77, 1988.

[Hal89] Rogers P. Hall. Computational approaches to analogical reasoning: A comparative analysis. *Artificial Intelligence*, 39:39–120, 1989.

[Hal94] Victoria Hall. Uncertainty-valued horn clauses. Technical Memo TM-94-03, DFKI GmbH, Postfach 2080, 67608 Kaiserslautern, Germany, September 1994.

[HC91] Kurt A. Heller and Perleth Christoph. Kreatitivitätstests. In Kurt A. Heller, editor, *Begabungsdiagnostik in der Schul- und Erziehungsberatung*, chapter 3.3.4, pages 177–182. Huber, Bern, 1991.

[Her89] Joachim Hertzberg. *Planen — Einführung in die Planerstellungsmethoden der künstlichen Intelligenz*, volume 65 of *Informatik*. BI Wissenschaftsverlag, 1989.

[HM91] Jiarong Hong and Chengjing Mao. Incremental discovery of rules and structure by hierarchical and parallel clustering. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, chapter 10, pages 177–194. AAAI Press/ The MIT Press, Menlo Park, CA, 1991.

[HY89] Mirsad Hadzikadic and David Y. Y. Yun. Concept formation by incremental conceptual clustering. In N. S. Sridharan, editor, *IJCAI-89, Proceedings of the 11th International Joint Conference on AI, August 20–25, 1989, Detroit Michigan, USA*, volume 1, pages 831–836. Morgan Kaufmann Publishers, 1989.

[Kar93] Peter D. Karp. Design methods for scientific hypothesis formation and their application to molecular biology. *Machine Learning*, 12(1/2/3):89–116, 1993.

[KD88] David Klahr and Kevin Dunbar. Dual space search during scientific discovery. *Cognitive Science*, 12:1–48, 1988.

[KDF90] David Klahr, Kevin Dunbar, and Anne L. Fay. Designing good experiments to test bad hypotheses. In *Computational Models of Scientific Discovery and Theory Formation*, chapter 12, pages 355–402. Morgan Kaufmann Publishers, San Mateo, CA, 1990.

[Kod90] Yves Kodratoff. Combining similarity and causaltity in creative analogy. In Luigia Carlucci Aiello, editor, *ECAI90, 9th European Conference on Artificial Intelligence, August 6-10, Stockholm, Schweden*, pages 398–403. Pitman Publishing, London, 1990.

[Kok91] Sakir Kokabas. Conflict resolution as discovery in particle physics. *Machine Learning*, 6(3):277–309, 1991.

[Kol93] Janet L. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann Publishers, 1993.

[Kol94] Janet L. Kolodner. Understanding creativity: A case-based approach. In *Topics in Case-Based Reasoning, 1st European Workshop, EWCBR-93, Kaiserslautern, Germany, 1–5 Nov. 93*, volume 837 of *LNCS*, pages 3–20. Springer Verlag, 1994.

[KP88] Robert Kail and James W. Pellegrino. *Menschliche Intelligenz*. Spektrum der Wissenschaft Verlagsgesellschaft, Heidelberg, 1988.

[KP90] Janet L. Kolodner and Theresa Louise Penberthy. A case-based approach to creativity in problem solving. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society, Cambridge, MA*, pages 978–985, august 1990.

[KS88] Deepak Kulkarni and Herbert A. Simon. The processes of scientific discovery: The strategy of experimentation. *Cognitive Science*, 12:139–175, 1988.

[KS90] Deepak Kulkarni and Herbert A. Simon. Experimentation in machine discovery. In *Computational Models of Scientific Discovery and Theory Formation*, chapter 9, pages 255–273. Morgan Kaufmann Publishers, San Mateo, CA, 1990.

[Kuh73] Thomas S. Kuhn. *Die Struktur wissenschaftlicher Revolutionen*. Number 25 in Suhrkamp Taschenbuch Wissenschaft. Suhrkamp, Frankfurt am Main, 2nd edition, 1973. German translation of the english original "The Structure of Scientific Revolutions".

[Kui93a] Benjamin J. Kuipers. Qualitative simulation: Then and now. *Artificial Intelligence*, 59(1/2):133–140, february 1993. (Special Volume: AI in Perspective).

[Kui93b] Benjamin J. Kuipers. Reasoning with qualitative models. *Artificial Intelligence*, 59(1/2):125–132, february 1993. (Special Volume: AI in Perspective).

[KW93] Janet L. Kolodner and Linda M. Wills. Case-based creative design. *AISB Quarterly*, 85:50–57, Autumn 1993. Reprint from the AAAI Spring Symposium on AI and Creativity, Stanford, CA.

[Lan69] Erika Landau. *Psychologie der Kreativität*. Ernst Reinhardt Verlag, München, 2nd edition, 1969.

[LB84] Douglas B. Lenat and John Seely Brown. Why AM and EURISKO appear to work. *Artificial Intelligence*, 23:269–294, 1984.

[Leb87] Michael Lebowitz. Experiments with incremental concept formation: Unimem. *Machine Learning*, 2:103–138, 1987.

[Len82a] Douglas B. Lenat. AM: Discovery in mathematics as heuristic search. In *Knowledge-Based Systems in AI*, Advanced Computer Science Series. McGraw-Hill, 1982.

[Len82b] Douglas B. Lenat. The nature of heuristics. *Artificial Intelligence*, 19:189–249, 1982.

[Len83a] Douglas B. Lenat. Eurisko: A program that learns new heuristics and domain concepts, the nature of heuristics III: Program design and results. *Artificial Intelligence*, 21:61–98, 1983.

[Len83b] Douglas B. Lenat. The role of heuristics in learning by discovery: Three case studies. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, *Machine Learning*, chapter 9, pages 243–306. Tioga Publishing Co., 1983.

[Len83c] Douglas B. Lenat. Theory formation by heuristic search, the nature of heuristics II: Background and examples. *Artificial Intelligence*, 21:31–59, 1983.

[LG90] Douglas B. Lenat and Ramanathan V. Guha. Cyc: A midterm report. *AI Magazine*, 11(3):32–59, 1990.

[LGP+90] Douglas B. Lenat, Ramanathan V. Guha, Karen Pittman, Dexter Pratt, and Mary Shepherd. Cyc, towards programs with common sense. *Communications of the ACM*, 33(8):30–49, August 1990.

[LISG91] John M. Long, Erach A. Irani, James R. Slagle, and The POSCH Group. Automating the discovery of causal relationships in a medical records database: The POSCH AI project. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, chapter 27, pages 465–476. AAAI Press/ The MIT Press, Menlo Park, CA, 1991.

[LS92] Brian Logan and Tim Smithers. Creativity and design as exploration. In John S. Gero and Mary Lou Maher, editors, *Modeling creativity and knowledge-based creative design*, chapter 7, pages 139–176. Lawrence Erlbaum Associates; Hillsdale, New Jersey, 1992.

[LSB87] Pat Langley, Herbert A. Simon, and Gary L. Bradshaw. Heuristics for empirical discovery. In L. Bolc, editor, *Computational Models of Learning*. Springer, 1987.

[McL92] Sally McLaughlin. Emergent value in creative products: some implications for creative processes. In John S. Gero and Mary Lou Maher, editors, *Modeling creativity and knowledge-based creative design*, chapter 4, pages 43–89. Lawrence Erlbaum Associates; Hillsdale, New Jersey, 1992.

[Mic90] Ryszard S. Michalski. Learning flexible concepts: Fundamental ideas and a method based on two-tiered representation. In Y. Kodratoff and R. S. Michalski, editors, *Machine Learning: An Artificial Intelligence Approach (Volume III)*, pages 63–111. Morgan Kaufmann Publishers, San Mateo, CA, 1990.

[ML91] Kathleen B. McKusick and Pat Langley. Constraints on tree structure in concept formation. In John Mylopoulos and Ray Reiter, editors, *IJCAI-91, Proceedings of the 12th International Joint Conference on AI, August 24–30, 1991, Darling Harbour, Sydney, Australia*, volume 2, pages 810–816. Morgan Kaufmann Publishers, 1991.

[Moo91] Raymond J. Mooney. Explanation-based learning as concept formation. In Douglas H. Fisher, Michael J. Pazzani, and Pat Langley, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, chapter 7, pages 179–205. Morgan Kaufmann Publishers, 1991.

[Moo93] Raymond J. Mooney. Induction over the unexplained: Using overly-general domain theories to aid concept learning. *Machine Learning*, 10(1):79–110, 1993.

[Mor94] Katharina Morik. Maschinelles Lernen. In Günther Görz, editor, *Einführung in die KI*, chapter 3, pages 247–301. Addison-Wesley, 1994.

[MS88] Shaul Markovitch and Paul D. Scott. The role of forgetting in learning. In John Laird, editor, *Proceedings of the 5th International Conference on Machine Learning, Ann Arbor, MI*, pages 459–465. Morgan Kaufmann Publishers, 1988.

[MS93] Shaul Markovitch and Paul D. Scott. Information filtering: Selection mechanisms in learning systems. *Machine Learning*, 10:113–151, 1993.

[Nav91] D. Navinchandra. *Exploration and Innovation in Design: Towards a Computational Model*. Symbolic Computation. Springer, New York, 1991.

[NL93] Bernd Nordhausen and Pat Langley. An integrated framework for empirical discovery. *Machine Learning*, 12(1/2/3):17–47, August 1993. (Special Issue on Machine Discovery).

[Paz93] Michael Pazzani. Learning causal pattern: Making a transition from data-driven to theory-driven learning. *Machine Learning*, 11:173–194, 1993.

[Ped91] Edwin P. D. Pednault. Minimal-length encoding and inductive inference. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, chapter 4, pages 71–92. AAAI Press/ The MIT Press, Menlo Park, CA, 1991.

[PK86] Bruce W. Porter and Dennis F. Kibler. Experimental goal regression: A method for learning problem-solving methods. *Machine Learning*, 1:249–286, 1986.

[PK92] Michael J. Pazzani and Dennis Kibler. The utility of knowledge in inductive learning. *Machine Learning*, 9:57–94, 1992.

[Pri93] A. E. Prieditis. Machine discovery of effective admissable heuristics. *Machine Learning*, 12(1/2/3):117–141, 1993.

[QS90] Y. Qin and Herbert A. Simon. Laboratory replication of scientific discovery processes. *Cognitive Science*, 14:281–312, 1990.

[Raj93] Shankar A. Rajamoney. The design of discrimination experiments. *Machine Learning*, 12(1/2/3):185–203, august 1993. (Special Issue on Machine Discovery).

[RG92] Michael S. Rosenman and John S. Gero. Creativity in design using a design prototype approach. In John S. Gero and Mary Lou Maher, editors, *Modeling creativity and knowledge-based creative design*, chapter 6, pages 111–138. Lawrence Erlbaum Associates; Hillsdale, New Jersey, 1992.

[ROD80] Erwin Roth, Wolf D. Oswald, and Konrad Daumenlang. *Intelligenz: Aspekte, Probleme, Perspektiven*. W. Kohlhammer, Stuttgart, 4th edition, 1980.

[RP93] Jon Rowe and Derek Patridge. Creativity: A survey of AI approaches. *Artificial Intelligence Review*, 7(1):43–70, 1993.

[RS90] Larry Rendell and Raj Seshu. Learning hard concepts through constructive induction: Framework and rationale. *Computational Intelligence*, 6:247–270, 1990.

[RW91] Michael M. Richter and Oliver Wendel. Lernende Systeme, Teil I: Symbolische Methoden. Technical report, Fachbereich Informatik, Universität Kaiserslautern, 1991. revised edition 1994.

[Sch93] Jan Marten Schraagen. How experts solve a novel problem in experimental design. *Cognitive Science*, 17(2):285–309, 1993.

[Sha91] Cullen Shaffer. On evaluation of domain-independent scientific function-finding systems. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, chapter 5, pages 93–106. AAAI Press/ The MIT Press, Menlo Park, CA, 1991.

[She93] Wei Min Shen. Discovery as autonomous learning from the environment. *Machine Learning*, 12(1/2/3):143–165, august 1993. (Special Issue on Machine Discovery).

[SHJ+94] Colleen M. Seifert, Kristian J. Hammond, Hollyn M. Johnson, Timothy M. Converse, Thomas F. McDougal, and Scott W. Vanderstoep. Case-based learning: Predictive features in indexing. *Machine Learning*, 16(1/2):37–56, 1994. (Special Issue on Computational Models of Human Learning).

[SHM91] Barry G. Silverman, Michael R. Hieb, and Toufic M. Mezher. Unsupervised discovery in an operational control setting. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, chapter 25, pages 431–448. AAAI Press/ The MIT Press, Menlo Park, CA, 1991.

[SK74] Inge Seiffge-Krenke. *Probleme und Ergebnisse der Kreativitätsforschung.* Verlag Hans Huber, Bern, 1974.

[SL89] Roger C. Schank and David B. Leake. Creativity and learning in a case-based explainer. *Artificial Intelligence*, 40:353–385, 1989.

[SL90] Jeff Shrager and Pat Langley. Computational approaches to scientific discovery. In *Computational Models of Scientific Discovery and Theory Formation*, chapter 1, pages 1–26. Morgan Kaufmann Publishers, San Mateo, CA, 1990.

[SM93] Paul D. Scott and Shaul Markovitch. Experience selection and problem choice in an exploratory learning system. *Machine Learning*, 12(1/2/3):49–67, august 1993. (Special Issue on Machine Discovery).

[SNK91] Tony Simon, Allen Newell, and David Klahr. A computational account of children's learning about number conservation. In Douglas H. Fisher, Michael J. Pazzani, and Pat Langley, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, chapter 15, pages 423–462. Morgan Kaufmann Publishers, 1991.

[Som94] Léa Sombé. A glance at revision and updating in knowledge bases. *International Journal of Intelligent Systems*, 9:1–27, 1994.

[TCH89] Paul Thagard, Dawn M. Cohen, and Keith J. Holyoak. Chemical analogies: Two kinds of explanation. In N. S. Sridharan, editor, *IJCAI-89, Proceedings of the 11th International Joint Conference on AI, August 20–25, 1989, Detroit Michigan, USA*, volume 1, pages 819–824. Morgan Kaufmann Publishers, 1989.

[TL91] Kevin Thompson and Pat Langley. Concept formation in structured domains. In Douglas H. Fisher, Michael J. Pazzani, and Pat Langley, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, chapter 5, pages 127–161. Morgan Kaufmann Publishers, 1991.

[Ulm68] Gisela Ulman. *Kreativität.* Pädagogisches Zentrum, Veröffentlichungen Reihe C: Berichte, Band 11. Verlag Julius Beltz, Weinheim, Berlin, Basel, 2nd edition, 1968.

[VC91] Manuela M. Veloso and Jaime G. Carbonell. Learning by analogical replay in prodigy: First results. In *Machine Learning — EWSL-91, European Working Session on Learning, Porto, Purtugal*, pages 375–390. Springer Verlag, March 1991.

[Wei86] Robert W. Weisberg. *Creativity: Genius and Other Myths.* A Series of Books in Psychology. W.H. Freeman and Company, NY, 1st edition, 1986.

[Wei89] Robert W. Weisberg. *Kreativität und Begabung.* Spektrum der Wissenschaft Verlagsgesellschaft, Heidelberg, 1st edition, 1989. German edition of [Wei86].

[WM91] Edward J. Wisniewski and Douglas L. Medin. Harpoons and long sticks: The interaction of theory and similarity in rule induction. In Douglas H. Fisher, Michael J. Pazzani, and Pat Langley, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, chapter 9, pages 237–278. Morgan Kaufmann Publishers, 1991.

[WM94] Janusz Wnek and Ryszard S. Michalski. Hypothesis-driven constructive induction in AQ17-HCI: A method and experiments. *Machine Learning*, 14(2):139–168, february 1994.

[Wro89] Stefan Wrobel. Demand-driven concept formation. In Katharina Morik, editor, *Knowledge Representation and Organization in Machine Learning*, volume 347 of *LNCS*. Springer, 1989.

[Wro94] Stefan Wrobel. Concept formation during interactive theory revision. *Machine Learning*, 14(2):169–191, February 1994. (Special Issue on Evaluating and Changing Representation).

[WW91] Yi-Hua Wu and Shulin Wang. Discovering functional relationships from observational data. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, chapter 3, pages 55–70. AAAI Press/ The MIT Press, Menlo Park, CA, 1991.

[Yaz89] Masoud Yazdani. A computational model of creativity. In Richard Forsyth, editor, *Machine Learning: Principles and Techniques*, chapter 9, pages 171–184. Chapman and Hall, 1989.

[YF91] Jungson Yoo and Douglas H. Fisher. Concept formation over problem-solving experience. In Douglas H. Fisher, Michael J. Pazzani, and Pat Langley, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, chapter 10, pages 279–303. Morgan Kaufmann Publishers, 1991.