Autonomous Agents and the Concept of Concepts

Autonomous Agents and the Concept of Concepts

Paul Davidsson

Doctoral dissertation submitted in partial fulfillment of the requirements
for the degree of PhD in Computer Science.

Department of Computer Science, Lund University

First edition

© 1996 by Paul Davidsson

Department of Computer Science
Lund University
Box 118
S–221 00 Lund
Sweden

paul.davidsson@dna.lth.se

http://www.dna.lth.se

ISBN 91–628–2035–4

CODEN LUTEDX/(TECS–1006)/1–221/(1996)

Printed in Sweden by KFS i Lund AB

Preface

Although this thesis is written in the form of a monograph, some of the results presented have already been published in various articles and reports. These are listed below and will in the text be referenced using the associated Roman numerals as follows:

I. E. Astor, P. Davidsson, B. Ekdahl, and R. Gustavsson. "Anticipatory Planning", In *Advance Proceedings of the First European Workshop on Planning*, 1991.

II. P. Davidsson, "Concept Acquisition by Autonomous Agents: Cognitive Modeling versus the Engineering Approach", Lund University Cognitive Studies 12, ISSN 1101–8453, Lund University, Sweden, 1992.

III. P. Davidsson, "A Framework for Organization and Representation of Concept Knowledge in Autonomous Agents", In *Scandinavian Conference of Artificial Intelligence – 93*, pages 183–192, IOS Press, 1993.

IV. P. Davidsson, "Toward a General Solution to the Symbol Grounding Problem: Combining Machine Learning and Computer Vision", In *AAAI Fall Symposium Series, Machine Learning in Computer Vision*, pages 157–161, 1993.

V. P. Davidsson, E. Astor, and B. Ekdahl, "A Framework for Autonomous Agents Based on the Concept of Anticipatory Systems", In *Cybernetics and Systems '94*, pages 1427–1434, World Scientific, 1994.

VI. B. Ekdahl, E. Astor, and P. Davidsson, "Towards Anticipatory Agents", In *Intelligent Agents — Theories, Architectures, and Languages*, M. Wooldridge and N.R. Jennings (ed.), pages 191–202, Springer Verlag, 1995.

VII. P. Davidsson, "On the Concept of Concept in the Context of Autonomous Agents", In *Second World Conference on the Fundamentals of Artificial Intelligence*, pages 85–96, 1995.

VIII. P. Davidsson, "Learning Characteristic Decision Trees", In *Eighth Australian Joint Conference on Artificial Intelligence (AI'95)*, World Scientific, 1995.

IX. P. Davidsson, “A Linearly Quasi-Anticipatory Autonomous Agent Architecture: Some preliminary experiments”, In *Distributed Artificial Intelligence (preliminary title)*, C. Zhang and D. Lukose (ed.), To be published in the Lecture Notes in Artificial Intelligence series, Springer Verlag, 1996.

X. P. Davidsson, “Coin Classification Using a Novel Technique for Learning Characteristic Decision Trees by Controlling the Degree of Generalization”, Accepted as a long paper to the *Ninth International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE-96)*, 1996.

Acknowledgements

First of all, I would like to thank my advisor Eric Astor for his indispensable help and encouragement while writing this thesis, not to mention the time he has spent making it readable. I also thank all my colleagues at the Department of Computer Science for providing a pleasant research environment. In particular, Bertil Ekdahl (the third member of our small research group) for always sharing his often original ideas, Robert Pallbo (former group member) for similar reasons, Arne Andersson for stimulating discussions, Stefan Nilsson and Kurt Swanson for being good friends and for that they, like Roger Henriksson, Jesper Larsson, Hans Olsson and many others, have helped me solve numerous problems related to Unix, LaTeX, C++, Simula and other esoteric topics.

I am grateful to Rune Gustavsson at the Department of Economics and Computer Science, University of Karlskrona/Ronneby who among other things introduced me to the field of autonomous agents more than five years ago and to Staffan Hägg and Fredrik Ygge at the same department for their useful comments on my work and for making my visits to Ronneby memorable.

I also want to thank everybody at the Department of Cognitive Science, led by Peter Gärdenfors, for always providing open-minded discussions at their weekly coffee-seminars which have forced me to view many scientific problems from new perspectives. Special thanks to Christian Balkenius and Måns Holgersson for many valuable comments on earlier drafts of this thesis.

Furthermore, I would like to thank Olof Tilly at the Department of Philosophy for his comments on the style and content of a large part of this thesis and Anders Holtsberg at the Department of Mathematical Statistics for kindly answering my naïve questions about multivariate statistics.

Finally, I want to thank Tanja for love and understanding, and my mother and father for their encouragement and support.

Lund, April 1996

Paul Davidsson

Contents

Chapter 1

Introduction

This thesis actually has two main topics. The first regards *autonomous agents* which can be seen as systems capable of interacting independently and effectively with their environment via their own sensors and effectors in order to accomplish some task. Although the development of such systems has always been regarded as one of the ultimate goals of Artificial Intelligence (AI), it is only in the last decade that research on autonomous agents has begun to receive significant attention. The main reason for this delay is the inherent complexity of the task. In order to reduce its complexity, AI researchers have traditionally studied the different aspects of intelligence separately, e.g., planning, learning, vision, or knowledge representation.

Moreover, with the fast progress in the field of computer communications and the emergence of highly complex heterogeneous software systems, the amount of research on software agents has increased rapidly in the last couple of years (and is, in fact, becoming mainstream computer science).[1] Software agents differ from autonomous agents mainly in that they are not embodied, that is, they are not physical entities acting in the physical world but rather software entities acting in software environments and communicating with other software agents. However, there are also many similarities between software agents and (embodied) autonomous agents, for instance, both are situated in an environment that they are supposed to reason about and interact with. As a consequence, many aspects of their internal structure, i.e., architectural issues, are common to both types of agents.

Apart from being a main topic in it own right, autonomous agents will also provide the framework in which we will study the second topic. A topic that concerns the entities which "...seem to be the very stuff of which cognitions are made" [223] and "...are assumed to be the basic constituents of thought and belief" [250], namely, *concepts*.

[1] Unfortunately, there is a tendency to call all kinds of different software programs agents, devaluing the term and turning it into a "buzz-word".

Why study concepts in the first place? Is it not a well investigated topic within AI? According to some leading researchers, it is not. Kirsh, for instance, writes [142]: "Evidence that the notion of concept is understudied in AI is easy to find." As the present thesis examines concept formation in the context of autonomous agents acting in real-world environments, this is even more pertinent.

In the early stages of the work documented here, the main objects of interest were the learning and the formation, and to some extent the representation, of concepts by autonomous agents. But as the research proceeded it became apparent that these topics could not, or at least should not, be studied without taking some more fundamental aspects of concepts into account. Examples of such aspects are the functions of concepts and the nature of the categories that the concepts represent. In short, we will investigate the very concept of concepts and in contrast to most other studies of concept learning, these topics will here be given a detailed treatment.

1.1 On the Scientific Method

There are in principle two approaches to the studying of computational intelligence, or cognition:

- *Cognitive modeling*, which strives to develop theories of the actual cognitive processes in humans (or animals).

- *The engineering approach*, which attempts to explore all possible cognitive mechanisms, irrespective of their occurrence in living organisms.

Traditionally, cognitive modeling of the different aspects of concepts has mainly been studied within the different cognitive sciences. Most notable are *cognitive psychology*, which studies how humans deal with concepts in memory, perception, and reasoning, and *philosophy (of mind)*, where ontological and epistemological questions regarding the nature of concepts are studied. Some interesting work has also been carried out within the fields of *developmental psychology*, which deals with questions regarding how we learn and form concepts during childhood, *linguistics*, where the relation between concepts and language is studied, and *neurology*, which investigates the low-level processing of concepts in the brain. The engineering approach, on the other hand, has mainly been studied within the field of artificial intelligence. As concept learning and representation are indeed central parts of the general problem of computational intelligence, this distinction applies to these topics as well.

While the engineering approach has sometimes been successful in the learning and formation of artificial concepts in restricted domains, its success has been limited in more realistic scenarios. One such scenario, probably the most natural, general and realistic, concerns a concept-learning autonomous agent acting in a real-world environment, which is exactly the scenario in which the different aspects of concepts will be studied in this thesis.

In what follows we will basically follow the engineering approach. However, since humans are clearly autonomous agents capable of acquiring and using concepts while interacting with the real world in a far more successful way than current AI systems, it may be a good idea to become inspired by the research on cognitive modeling. When adopting such a mixed approach, there are some things to keep in mind. The most important is perhaps that the task of creating an autonomous agent is not equivalent with cognitive modeling. For example, if there are no advantages in mimicking a particular feature of human concept acquisition then we have no reason to do so, i.e., there are no reasons for assuming that humans are optimal agents. On the other hand, a cognitive model does not have to be a true model of human cognition to be useful in AI. As a consequence, we will not argue about the biological and psychological plausibility of the cognitive models presented. Moreover, it is important to remember that we only have very limited insight into the actual workings of human cognition. Another problem with adopting this approach is that in experimental psychology, the theories of cognitive processes are often not described in enough detail to be implemented in a computer.

One of the main goals of this thesis is to pull together different lines of argumentation that have emerged from the cognitive sciences and the field of artificial intelligence in order to establish a solid foundation for further research into representation, learning, and formation of concepts by autonomous agents. The aim has been to do this in an unbiased fashion without any preconceptions concerning the actual implementation, i.e., without assuming a symbolic, connectionist, or hybrid system. In addition to being a survey of the research within these fields, this will also result in some original hypotheses concerning different aspects of concepts in the context of autonomous agents. Thus, the results and conclusions, of this survey will sometimes, but not always, be stated in terms of new insights and ideas rather than resulting in new algorithms or methods. Moreover, since no adequate formalism exists, it is not possible to provide formal proofs of any of these conclusions. Indeed, it is the author's opinion that at the present stage of research any attempt at formalization would be premature.[2] This will hopefully become apparent to the reader as well when the formalizations that do exist concerning concept learning are discussed. It will, for instance, be argued that

[2] In fact, Herbert Simon, in an invited talk at AAAI-93, pointed out that the apparent elegance of formalisms and theorems makes the AI field oversimplify and ignore complex problems and structures that need to be studied [49]. He argued that AI belongs to the *sciences of qualitative descriptions*, such as biology and chemistry, rather than being a science that is adequately describable by mathematical formulae, such as physics (and algorithm complexity).

these formalizations make too strong assumptions about both the representation and the learning situation to be of any practical interest in the context of autonomous agents. In fact, many of the hypotheses suggested here will concern this kind of assumptions, and are thus also of interest for the development of novel, more interesting formalizations. Consequently the arguments presented will be informal, and moreover they will often be of a qualitative nature rather than quantitative.

1.2 Outline

In Chapter 2 and Chapter 4 the issues of autonomous agents and world modeling are discussed. We argue against both purely reactive agents, based on stimulus-response behavior, and purely deliberative agents, based on a sense-model-plan-act cycle. Instead, support is presented for hybrid approaches that make use of both explicit world models for deliberative reasoning and stimulus-response behavior for low-level control. A novel approach of this kind referred to as *anticipatory agents* and based on the concept of *anticipatory systems* is presented in Chapter 3. Some experimental results on a special kind of such agents based on a linearly quasi-anticipatory agent architecture are also presented.

Chapter 5 and Chapter 6 address some fundamental questions regarding concepts, such as: What does it mean to have a concept? and: What functions do, or should, concepts serve? By analyzing the research in cognitive science and philosophy, these questions are first answered in the context of human cognition. This analysis then provides a basis for a discussion of the questions in the context of artificial agents. This approach, beginning with an analysis of the cognitive modeling research on a particular topic and then letting this analysis serve as a basis for the discussion of the corresponding topic using the engineering approach, is repeated in most of the remaining chapters of this part of the thesis.

Chapter 7 is devoted to a discussion on what can be said about *categories* in general without taking into account how they are actually represented internally by an agent. This includes analyses of some fundamental concepts such as similarity, property, and taxonomy. Moreover, an attempt to identify different kinds of categories is made.

Chapter 8, on the other hand, treats the issue of how an agent should represent categories internally. Different suggestions from different fields are presented and evaluated according to the desirable functions identified in Chapter 6. It is concluded that none of the existing approaches is able to serve all these functions and that it is unrealistic to expect that any singular kind of (i.e., a monolithic) representation would be adequate. Based on this insight, a new approach for representing categories in autonomous agents is presented. It is a composite representation scheme where each component is motivated by the functions a concept should serve.

Chapter 9 then, concerns the acquisition of concepts. A number of theories from various fields are presented. Some of the requirements that an autonomous concept learning system must meet are identified and provide the basis for the evaluation of the existing theories. A method for making any learning algorithm satisfy one such requirement, namely that of representing concepts by characteristic descriptions, is presented together with some promising experimental results. In contrast to previous methods for learning characteristic descriptions, it is possible with this method to control the degree of generalization. In addition, a model for integrating learning by being told, learning from examples and learning by observation is outlined. This chapter also includes a brief discussion concerning the question of what actually can be learned. Finally, Chapter 10 provides a summary of the conclusions of each chapter together with some pointers for further research.

To sum up, the main contributions of this thesis, besides the surveys of autonomous agents and of different aspects of concepts, are (i) the framework for anticipatory agents, in particular the linearly quasi-anticipatory agent architecture, (ii) the composite concept representation scheme, and (iii) the general method for learning characteristic concept descriptions.

Chapter 2

Autonomous Agents

An autonomous agent can be seen as a system capable of interacting independently and effectively with its environment via its own sensors and effectors in order to accomplish some given or self-generated task(s). Thus, humans and most animals may in this sense also be regarded as autonomous agents. In the following, however, we will by autonomous agents refer to artificial ones. One of the ultimate goals for AI is to construct artificial intelligent autonomous agents capable of human level performance (or better). However, a look at the state of current AI-research reveals that we are quite far from achieving this goal.

This chapter provides an introduction to autonomous agents in general, but with emphasis on the two major approaches for designing autonomous agents: the deliberative and the reactive approach. It is concluded that a combination of these probably will yield the best solution. A novel suggestion for such a solution will be provided in the next chapter.

2.1 Introduction

All autonomous agents have, more or less, the basic architecture shown in Figure 2.1. The sensors (e.g., visual, range and touch) receive input from the environment and provide data for the cognitive component. The cognitive component then decides which actions to perform and commands the effectors (e.g., different kinds of motors) to actually carry out these actions.

2.1.1 Different Kinds of Agents

The term “autonomous agent” is, as most terms in AI, ambiguously used. What one researcher would consider an autonomous agent, another refers to as a simulation pro-

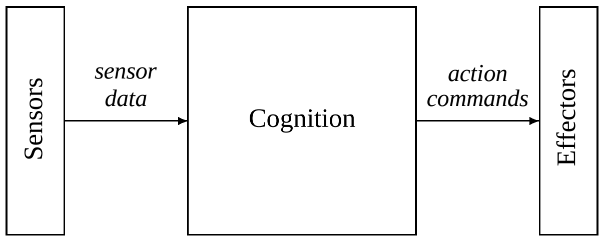


Figure 2.1: The basic architecture of an autonomous agent.

gram. However, it is possible to divide the (most general) class of autonomous agents into categories on the basis of how, and to what degree, they actually interact with the real world. Two important features are, then, whether they are *situated* or not, and whether they are *embodied* or not. According to Brooks [42], situated agents are situated in the world in the sense that they do not only deal with abstract descriptions of it. The "here" and "now" of the environment directly influence the behavior of the agent. Embodied agents, on the other hand, "...have bodies and experience the world directly — their actions are part of a dynamic with the world, and the actions have immediate feedback on the robots' own sensations" (p. 1227). To make the distinction between situatedness and embodiment clearer, let us discuss some different kinds of autonomous agents in these terms.

Agents that are neither embodied nor situated are those that have least interaction with the real world; they are basically pure computer simulations of actual agents. A class of embodied agents which are not situated is, for instance, traditional industrial robots. They have physical bodies but do not use information about the current state of the environment to guide their behavior; they just execute a pre-programmed series of actions. A ticket reservation system, on the other hand, is situated, as the events in the environment (requests, database changes and so on) directly affect the system's behavior. However, since the system is not physical (in some sense) and since the interaction with the environment only consists of sending and receiving messages, it cannot be regarded as embodied. Other kinds of agents belonging to this category are *software agents*, or *softbots* (*soft*ware ro*bots*), that is, intelligent agents in real-world software environments such as operating systems or databases. For instance, Etzioni [80] have implemented a UNIX softbot that accepts high-level user goals and dynamically synthesizes appropriate sequences of commands. In this case, the effectors are UNIX

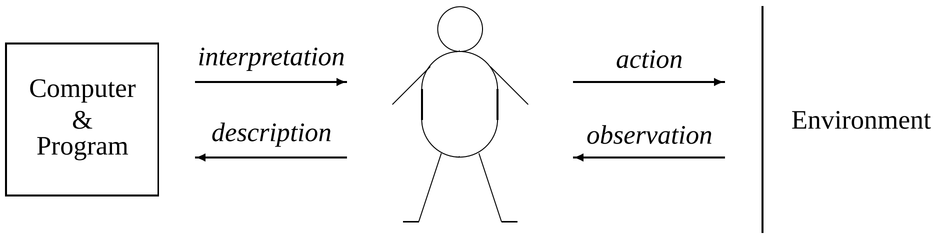


Figure 2.2: Traditional AI-system.

shell commands transmitted to the environment in order to change its state (e.g., `mv` or `compress`), whereas the sensors are commands that provide information to the agent (e.g., `pwd` or `ls`). Finally, we have agents that are both embodied and situated such as autonomous mobile robots (vehicles) and other physical robots that perceive the environment and use this information to guide their behavior. Table 2.1 summarizes the four possible cases.

	situated	not situated
embodied	mobile robots	traditional industrial robots
not embodied	software agents	computer simulations

Table 2.1: Categorization of autonomous agents.

It is common to regard only the members of the last category described above as real autonomous agents (as I did earlier in this chapter). We will here follow this usage in the sense that we will concentrate on agents that are both embodied and situated. It is, however, not unusual to regard also agents that are not embodied (e.g., softbots) as autonomous agents.

Finally, let us make explicit the relation between traditional AI systems and autonomous agents and how they interact with the environment. In traditional AI-systems (see Figure 2.2) there is a human operator present who observes the environment (i.e., the problem) and describes it to the computer. The results of the computer's computations are interpreted by the operator who then performs the required actions. An autonomous agent (see Figure 2.3), on the other hand, must observe the environment by itself and

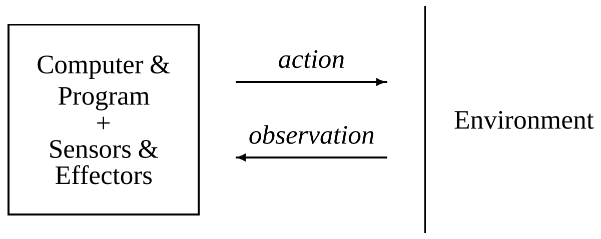


Figure 2.3: Autonomous agent.

turn these observations into descriptions for further computations. Moreover, it must interpret the results of its computations and then perform the appropriate actions.

2.1.2 Requirements

What general requirements should we make on an autonomous agent? According to Brooks [41] and Hayes-Roth [120], for instance, an autonomous agent should be:

- Adaptive; it must cope appropriately and in a timely fashion with changes in the environment.
- Robust; minor changes in the properties of the environment should not lead to total collapse of the agent's behavior.
- Tactical; it should be able to maintain multiple goals and, depending on the circumstances it finds itself in, change which particular goals it is actively pursuing.
- Versatile; it must be able to perform a large variety of tasks (in contrast to being single-purposive).

At the moment, there certainly do not exist agents with all these features, the features should rather be seen as guidelines. Moreover, these features are neither independent nor exhaustive.

2.1.3 Single versus Multi-Agent Scenarios

From the viewpoint adopted here, there are in principle two scenarios possible for an autonomous agent. Either the agent is alone in its environment, or there are other agents in the environment. A prototypical example of an agent being alone in its environment is an agent on an exploration mission on the surface of Mars. In a multi-agent scenario the agents can be either humans or machines (or both), as on a factory floor. However, in present research on multi-agent systems it is typically assumed that all agents are artificial.

Multi-agent systems are often assumed to be heterogeneous, i.e., the agents are of different kinds and are not specified by a single designer. Examples of multi-agent frameworks are, for instance, Societies of Computation suggested by Gustavsson and his colleagues [112] and the BEABLE system by Belo and Neves [25]. The agents are either *cooperating* in order to achieve a common goal or *competing* to achieve conflicting goals.

In multi-agent scenarios it is often supposed that the agents are able to communicate with each other. One approach for inter-agent communication, originating from the DARPA knowledge sharing effort [208], is ACL (Agent Communication Language) which use the Knowledge Query and Manipulation Language (KQML) as a high-level language and protocol, the Knowledge Interchange Format (KIF) [100] to specify content, together with a vocabulary. KQML supports a set of *performatives* which defines the permissible communicative operations that agents may attempt on each other such as: informing, asking questions, and commanding actions. A KQML expression can be seen as of consisting of three layers: (i) a content expression encapsulated in (ii) a message wrapper that specifies the performative, which in turn is encapsulated in (iii) a communication wrapper that specifies communication parameters such as, sender and receiver. The content can be expressed in any representation language that follows some general syntactic constraints. One candidate is KIF (used in ACL), which is an extended version of first order predicate logic formulated in a LISP-like syntax.

When implementing a multi-agent system there is a trade-off between having many simple agents, which to a great extent rely on agent communication, and having a few complex agents, which rely on the cognitive abilities of singular agents. We will in the following concentrate on scenarios with one or a few complex agents.

2.1.4 Possible Applications

There are numerous application areas for autonomous agents. Some of the already investigated are:

- maintenance activities in radiation-prone or toxic environments [270]
- means for the disabled [163, 263]
- deep-sea exploration and exploitation [32]
- servicing, management, and assembly tasks in space [195, 130]

- planetary exploration [21].

The objectives for developing these kinds of agents are mainly humane such as, replacing humans in hazardous, strenuous, or repetitive tasks. However, there may also be economic objectives, such as enhanced productivity, profitability, or quality. In addition, there is an enormous potential of unexplored applications which certainly will be investigated in the future, for example, personal household robots.[1]

2.1.5 Three Perspectives on Agents

Wooldridge and Jennings [288] identify three perspectives from which it is possible to study agents:

- *Agent theories:* What exactly are agents? What properties should they have, and how are we to formally represent and reason about these properties?
- *Agent architectures:* How are we to construct agents that satisfy the properties we expect from them? What software and/or hardware structures are appropriate?
- *Agent languages:* How are we to program agents? What are the right primitives for this task? How are we to effectively compile or execute agent programs?

Most agent theories regard agents as *intentional systems* (cf. Dennett [65]). An intentional system is an entity whose behavior can be predicted by ascribing beliefs and desires (and perhaps other kinds of mental states). If the system has beliefs and desires but no beliefs or desires about beliefs and desires, it is a first-order intentional system. A second-order intentional system, on the other hand, has also beliefs and desires about both its own beliefs and desires and those of other agents. Several logic-based formalisms have been suggested as agent theories. An overview of agent theories is provided by Wooldridge [287].

An agent language is a language which allows one to program a computer system in terms of beliefs, desires and other concepts developed by agent theorists. Most existing agent languages are inspired by the paradigm of *agent-oriented programming* (AOP) suggested by Shoham [244]. He argues that from the engineering point of view, AOP can be seen as a specialization of the object-oriented programming (OOP) paradigm:

> " ...whereas OOP proposes viewing a computational system as made up of modules that are able to communicate with one another and that have individual ways of handling incoming messages, AOP specializes the framework by fixing the state (now called *mental state*) of the modules (now

[1] In fact, this has already, at least partially, been subject for intensive studies; one symposium of the AAAI fall symposium series in 1993 [1] was devoted to autonomous vacuuming robots.

> called *agents*) to consist of components such as beliefs (including beliefs about the world, about themselves, and about one another), capabilities, and decisions, each of which enjoys a precisely defined syntax. Various constraints are placed on the mental state of an agent, which roughly correspond to constraints on their common sense counterparts. A computation consists of these agents informing, requesting, offering, accepting, rejecting, competing, and assisting one another." (p. 56)

Shoham presents in the same article a primitive agent language called AGENT-0. Examples of other agent languages are APRIL by McCabe and Clarke [172], and DAAL by Hägg and Ygge [113].

However, in remaining part of this chapter and in the next we will concentrate on architectures. Maes [164] gives the following picture of what is meant by an agent architecture:

> An *architecture* proposes a particular methodology for building such an autonomous agent. It specifies how the overall problem can be decomposed into subproblems, i.e. how the construction of the agent can be decomposed into the construction of a set of component modules and how these modules should be made to interact. The total set of modules and their interactions has to provide an answer to the question of how the sensor data and the current internal state of the agent determine the actions (effector outputs) and future internal state of the agent. An architecture encompasses techniques and algorithms that support this methodology. (p.115)

There are two major approaches for designing autonomous agents: the traditional top-down approach and the recently emerged bottom-up approach. Agents constructed according to these approaches are often called *deliberative* and *reactive* respectively. Characteristic for the traditional approach is that the cognitive abilities (perception, world modeling, planning, and so on) are modularized. Thus, the cognitive component is functionally decomposed. In this way it is possible to begin with the design of the overall architecture of the agent and then develop the different components separately. According to the bottom-up approach on the other hand, one should start with implementing simple behavior, covering the complete range from perception to action, and then incrementally adding more sophisticated behaviors. This distinction between behavioral and functional modularization is further elaborated by Brooks [41].

2.2 The Deliberative Approach

The deliberative approach has a long tradition in several areas of research such as: AI, Cognitive Science, Philosophy and Robotics (cf. Albus [6]). For instance, in 1943 the cognitive psychologist Craik [58] described the process of cognition in such terms that it closely resembles that of deliberative agents. The first step, he suggests, consists of translation of stimulus into an internal representation. This representation is then manipulated by cognitive processes to form new internal representations, which in turn are translated back into actions.

2.2.1 Deliberative Agents

Wooldridge and Jennings [289] define a deliberative agent "...to be one that contains an explicitly represented, symbolic model of the world, and in which decisions (for example about what actions to perform) are made via logical (or at least pseudo-logical) reasoning, based on pattern matching and symbolic manipulation." According to the most common deliberative approach the cognitive component consists of essentially two parts; a planner and a world model. Figure 2.4 illustrates the basic architecture.

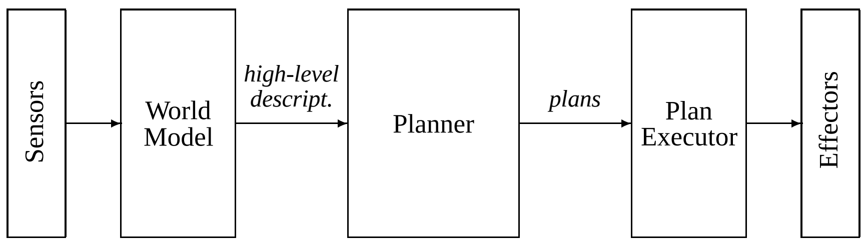


Figure 2.4: The basic architecture of a deliberative agent.

The world model is an internal description of the agent's external environment and sometimes also including a description of the agent itself. (Different aspects of world models will be discussed in Chapter 4.) The planner uses this description to make a plan of how to accomplish the agent's goal in the following way: given the atomic actions (operators) the agent is able to perform, their pre-conditions and their effects in the world (post-conditions), and the initial and goal situation, it searches through the space of operator sequences until one is found that will transform the initial state into the goal state. The resulting plan is a list of actions that is given to a plan executer, which will perform these actions by calling various low-level routines of the effectors. The most well-known (and one of the earliest) planning system is STRIPS developed by Fikes and Nilsson [84], which uses a simple means-ends analysis that matches the post-conditions of actions against the desired goal. Later, more sophisticated planners such as hierarchical and non-linear planners (cf. Sacerdoti [234]) were developed. Two examples of deliberative agent architectures are IPEM [8] and IRMA [36].

The working of a deliberative agent can be described in terms of a sense-model-plan-act cycle. The sensors sense the environment and produce sensor-data that is used to update the world model. The world model is then used by the planner to decide which actions to take. These decisions serve as input to the plan executor which commands the effectors to actually carry out the actions.

There is an underlying assumption in this approach that it is possible to modularize cognition functionally, i.e., that it is possible to study the different cognitive functions (e.g., perception, learning, planning, and action) separately and then put these components together to form an intelligent autonomous agent. Moreover, this assumption seems to have influenced the division of the field of AI into sub-fields. For instance, perception (vision) is studied within the field of *computer vision*, learning within *machine learning*, planning within *planning*, and action within *robotics*. According to Minton [187] there are clearly good reasons for functional modularization from the engineering perspective as modularity reduces the apparent complexity of a system. Additional support for functional modularization comes from brain-research. For instance, Gazzaniga [99] writes: "An emerging view is that the brain is structurally and functionally organized into discrete units or "modules" and that these components interact to produce mental activities." Also within cognitive psychology, similar ideas concerning modularization has been suggested, for example, by Fodor [93].

2.2.2 Limitations of the Deliberative Approach

The main part of the research on deliberative agents have studied the cognitive component in isolation (i.e., as a disembodied agent). When actually embodying this kind of agents (e.g., the mobile robot "Shakey" developed at the Stanford Research Institute [202]), it has been noticed that although the embodied agents are able to do some "sophisticated" cognitive tasks such as planning and problem solving, they have problems with "simpler" tasks such as routine reaction that require fast action but no extensive deliberation [42]. In fact, Chapman [51] showed that planning is undecidable in the general case, and computationally intractable due to NP-completeness even in its simpler forms. In other words, traditional planning is very time-consuming, requiring exponential search through potentially enormous problem spaces.

In addition, Brooks [41] argues that human-level intelligence is too complex and not well enough understood to be decomposed into the right components. Moreover, even if the decomposition should be right he thinks that we still do not know the right interfaces between the components.

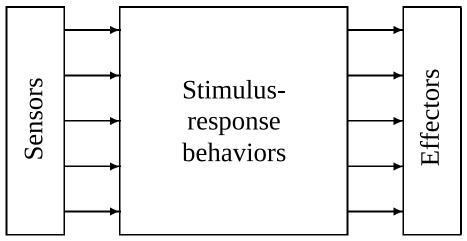


Figure 2.5: The basic architecture of a reactive agent.

2.3 The Reactive Approach

The first reactive agents emerged in the mid-eighties and were inspired by the idea that most of our every-day activities consist of routine action rather than abstract reasoning. So instead of doing world-modeling and planning, the agents should just have a collection of simple behavioral schemes which react to changes in the environment in a stimulus-response fashion. This results in the simple architecture shown in Figure 2.5 where cognition is reduced to a mapping of perceptual stimuli onto primitive actions.

Some of the most influential agents of this kind are Brooks' robots based on the *subsumption architecture* [39], *Pengi* [3], and those based on *situated automata* [230]. In the following, however, we will concentrate on Brooks' work since it is the most influential and extreme, and since it relies on a few principles which can be explicitly stated. More comprehensive overviews of reactive agents are provided by Davidsson [59] and Lyons and Hendriks [161].

2.3.1 The Work of Brooks

Some of the principles that guides Brooks' work on reactive agents are:

- behavioral modularization
- incremental construction
- no explicit representation of the world
- embodied agents
- purposive vision.

The reason he gives for choosing behavioral decomposition instead of functional is mainly, as described above, that we do not know which the functional components are or, for that matter, the appropriate interfaces between them. Brooks adopts a pure engineering approach when he constructs his reactive agents and proceeds with the construction in an incremental fashion. He starts with the simplest behavior and makes it work before more advanced behaviors are added.

The most controversial of Brooks' principles is the one concerning representation. He argues that explicit representations of the world are not only unnecessary but do also get in the way when implementing actual agents. Instead the agent should use "...the world as its own model — continuously referring to its sensors rather than to an internal world model" [41]. According to him, this is possible only by having situated agents that act in direct contact with the real world, not by using abstract descriptions of it only.

Another principle is that the agents should be embodied. In this way one cannot "cheat"; one has to face all the real problems that typically are disregarded otherwise. In addition, Brooks argues that "...only through physical grounding can any internal symbolic or other system find a place to bottom out, and give "meaning" to the processing going on within the system." In Chapter 4 we will further discuss the issue of symbol grounding.

The last principle concerns perception. In AI and Computer Vision there is (or, at least, has been) an assumption that the goal of vision is to make a three-dimensional model of the real world. Brooks argues that this task is too difficult to perform when acting in the real world. Instead, his agents use *purposive vision*. (This issue also will be treated in Chapter 4.)

2.3.2 Limitations of the Reactive Approach

Reactive agents have, at least in some experiments, been proved to be superior to traditional ones at doing a limited number of simple tasks in real-world domains. However, besides of not being particular versatile, they have problems to handle tasks that require knowledge about the world that must be obtained by reasoning or from memory, rather than perception. According to Kirsh [143] some possible candidates for such tasks are activities which require: response to events beyond the agent's current sensory limits, some amount of problem solving, understanding a situation from an objective perspective, prediction of other agents' behavior, or creativity (i.e., stimulus free activities).

Moreover, reactive agents are often hard-wired (sometimes by hand) and do often not have any learning abilities. This and the fact that each behavior must be separately encoded in the agent, leads to complexity problems both at design time and at execution time (cf. Ginsberg [103]). Related to this is the scaling problem addressed by Tsotsos [271]. (For more criticism of the reactive approach, see Kirsh [142, 143].)

There is also an evolution-based critique of the reactive paradigm that seldom is expressed. It seems reasonable to compare reactive agents to, for example, reptiles in the sense that their cognition is based on stimulus-response behavior. For instance, Sjölander [245] writes: “There is thus no true intermodality in the snake, just a number of separate systems, involving specific behavioral patterns connected to a specific sensory input.” Humans and other more developed species, on the other hand, are equipped with central representations and Sjölander suggests that this could be an explanation of why reptiles were superseded by mammals and birds. He concludes: “To go from monosensorially governed constructions of several internal representations to a centralized intermodal one must surely be one of the most important breakthroughs in the evolution of mind.” This suggests that agents based on the reactive paradigm will never reach human-level performance.

However, as Kirsh points out [143], there is absolutely nothing wrong with, for instance, Brooks’ approach from a scientific point of view. That is, if you interpret his work as an attempt to see how far you can go using only this kind of simple architecture. He should also be acknowledged for having pointed out several weaknesses of the traditional paradigm. The problem is that he seems to jump to conclusions based on some initial success (much in the same way as early AI-scientists did).

2.4 Combining the Approaches

More recently, several researchers have acknowledged that an intelligent agent should have both high-level reasoning and low-level reactive capabilities. In this way it is possible to utilize the reaction ability of reactive agents, which is necessary for routine tasks, and still having the power of deliberation, which is necessary for more advanced or long term tasks. In fact, the weaknesses of reactive architectures correspond closely to the strengths of deliberative architectures. Moreover, a combined, or *hybrid*, approach seems to model human functioning closer than the purely reactive approach, which resembles that of more primitive animals.

As Hanks and Firby [114] point out, two categories of hybrid agent architectures can be distinguished. *Uniform* architectures employ a single representation and control scheme for both reaction and deliberation, whereas *layered* architectures use different representations and algorithms (implemented in separate layers) to perform these functions. However, most uniform architectures, for example the Procedural Reasoning System (PRS) developed by Georgeff and his colleagues [101, 130], do not make any specific commitments on how reaction and deliberation should be interleaved. This, in addition to Ferguson’s [82] remark that “There are a number of other reasons for advocating a layered control approach, including increased behavioral robustness and operational concurrency, as well as improved program comprehensibility and system

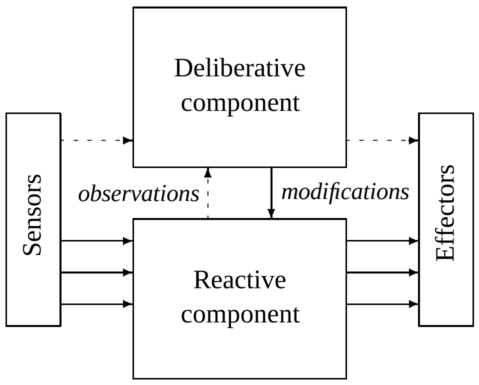


Figure 2.6: The basic architecture of a layered hybrid agent.

testability and analysability." (p. 48), is the reason that we in what follows will concentrate on layered approaches.

Figure 2.6 shows the basic principles of a layered hybrid agent architecture. The reactive component (i.e., layer) correspond closely to a reactive agent as described earlier, mapping perceptual stimuli onto primitive actions. The deliberative component (i.e., layer) performs symbolic reasoning in order to guide the behavior of the reactive component (sometimes by observing, or monitoring this behavior), for instance, by suggesting actions and plans, or by changing the reactive component's set of situation-action rules. In some architectures the deliberative component is directly connected to the agent sensors and/or effectors, whereas it is not in others.

As illustrated in Figure 2.7 we can, in fact, also describe a deliberative agent in terms of a hybrid agent: the planner together with the world model corresponding to the deliberative component and the plan executor to the reactive component. However, the reactive component is in this case totally dependent of the deliberative component; it will not perform any actions if there is no plan. This, in contrast to real reactive components that are able to select actions from sensor data independently of the deliberative component. In this case, the existence of an advice (i.e., action, plan or modification) will simply increase the goal-achieving ability of the reactive component. Thus, it is possible to define a continuum of agents from purely deliberative agents to layered hybrid agents with highly independent reactive and deliberative components.

Traditionally, and in the context of deliberative agents, planning has been defined as the "activity" of computing a sequence of operations that when applied will transform

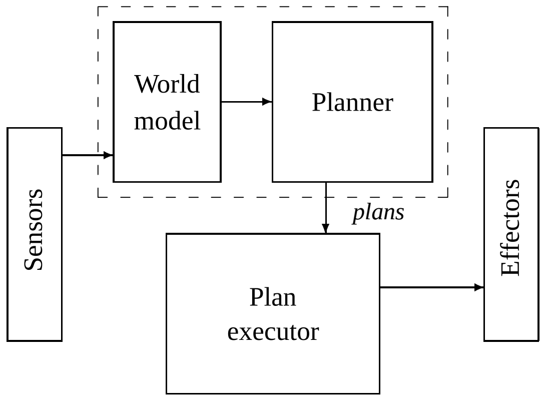


Figure 2.7: The basic architecture of a deliberative agent in terms of a layered hybrid agent.

an initial state into a goal state. However, as this definition often is not appropriate when dealing with hybrid agents, Lyons et al. [162] suggest, inspired by Agre and Chapman [3], that in this context: "...a planner is a system that continually modifies a reactive system, the reactor, so that its behavior becomes more goal directed." (As we shall see below, this definition does not suit all kinds of hybrid agents either.)

When specifying a layered hybrid agent architecture there are several critical design decisions that must be made. Of particular interest are:

(i) Is one reactive and one deliberative layer enough, or should additional layers be introduced? How should the cognitive workload be divided between the layers?

(ii) How should the components corresponding to the different layers interact?

(iii) When should the agent react and when should it deliberate, i.e., how should the scheduling algorithm be specified?

In the remaining part of this section we will take a closer look upon some of the proposals for hybrid architectures and compare them in terms of (i)–(iii) above. Although several hybrid architectures have been suggested in the last five years, very few, if any, comparisons between them can be found in the literature. When a new hybrid architecture is suggested, they are typically not compared with earlier suggestions in any depth. Thus, many ideas that have been put forward as novel are very similar to old ones.

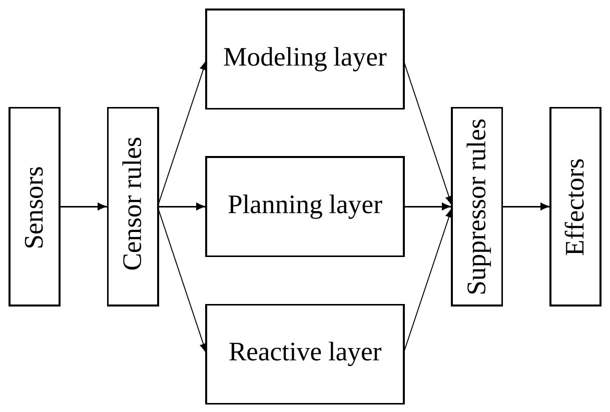


Figure 2.8: The basic architecture of a TouringMachine agent.

2.4.1 TouringMachines

(i) The TouringMachine architecture suggested by Ferguson [83, 82] consists of three layers: the *reactive* layer which is composed of a set of situation-action rules, the *planning* layer whose main component is a hierarchical, partial planner, and the *modeling* layer. These layers models the agent's world at different levels of abstraction and have different task-oriented capabilities. The task of the reactive layer is to provide fast actions for coping with immediate or short-term events which the higher levels are either unaware of, or not able to response to, because of lack of time or computational resources. The main task of the planning level is to generate and execute plans for achieving the agent's long-term goals. Finally, the task of the modeling layer is to detect and foresee potential goal conflict situations between agents and then propose suitable actions for avoiding such conflicts. This is done "...via execution monitoring (observation), abductive inference (explanation), and temporal and counterfactual reasoning (prediction)..." Figure 2.8 gives an overview of the TouringMachine architecture.

(ii) Each layer is independently connected to the sensors and effectors and acts as if it controls the agent by itself. As a result of this, the different layer's proposed actions will frequently conflict with each other. These conflicts are solved by suppressor rules which prevent particular action commands from being fed to the effectors. Likewise, on the perception side there are censor rules that filter sensor data so that each level gets the appropriate parts of the sensor data. However, there are no strictly hierarchical flow of control between the different layers. The different layers can send messages to each

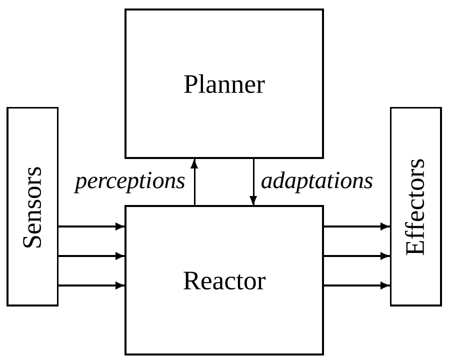


Figure 2.9: The basic architecture of the planner-reactor system.

other. Messages can be of two kinds: passive passing of information (e.g., the reactive layer offers suggestions to the modeling layer which entities in the world the latter may focus on) and actively alter another layer's control decisions (e.g., the modeling layer instructs the planning layer to generate a plan for a new task).

(iii) The layers are operating concurrently, but synchronously (controlled by an internal agent clock).

2.4.2 The Planner-Reactor Architecture

(i) The main idea behind the Planner-Reactor agent suggested by Lyons, Hendriks and Mehta [162] is that the Planner is a system which interacts with the Reactor much in the same way that the Reactor interacts with the world. (See Figure 2.9.) Input to the Planner consists of: a world model, a description of the reactor, and advice from the user regarding the goals of the Reactor and the constraints the Reactor should obey in its behavior. The Reactor is specified using the $\mathcal{RS}$ model, an extension of the *Robot Schemas* model [160] able to represent highly conditional plans.

(ii) The Planner cycles continuously, perceiving the Reactor's behavior using perception processes, and improving the Reactor according to the advice given by the user. It has no access to the complete internals of the Reactor, e.g., raw sensor data; its observations are restricted to the output of the perception processes. These processes are used to determine when the Reactor is in danger of failing, to reason about the current resource usage, and to determine when a goal has been met.

(iii) Reactor and Planner are regarded as completely separated concurrent systems.

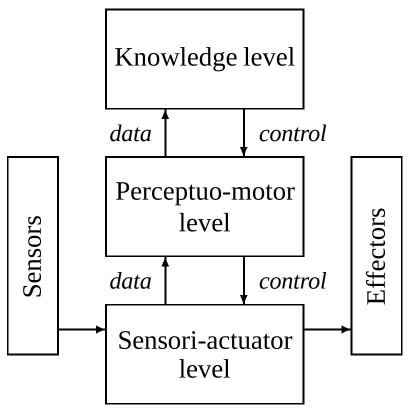


Figure 2.10: The basic architecture of a GLAIR agent.

2.4.3 The GLAIR Architecture

The Grounded Layered Architecture with Integrated Reasoning (GLAIR) suggested by Hexmoor, Lammens and Shapiro [150, 122] stresses symbol grounding, a topic which we will discuss in some detail in Section 8.5.1.

(i) GLAIR specifies three levels: the *knowledge level*, which contains a traditional planning system that uses a relatively course-grained representation of objects, events, and states of affairs; the *perceptuo-motor level*, which uses more fine-grained representations that are agent-centered and agent specific; and the *sensori-actuator level*, which has no explicit representations of any kind, only procedural representations at the effector side and sensor data on the sensory side. Moreover, a distinction is made between deliberative, reactive, and reflexive behaviors. Reflexive behavior does not require any deliberation or detailed sensor data processing and is carried out at the sensori-actuator level, whereas reactive behavior require some data processing but no extensive deliberation and is carried out at the perceptuo-motor level. In our terms, the perceptuo-motor level together with the sensori-actuator level can be said to constitute the reactive component. Finally, all deliberation takes place at the knowledge level.

(ii) The components interact as illustrated in Figure 2.10. Sensor data is abstracted from lower levels whereas decisions on higher levels are propagated through the levels below (and translated into more specific descriptions).

(iii) The three levels are semi-autonomous and processed in parallel. In particular, the lower level mechanisms are able to preempt higher level mechanisms.

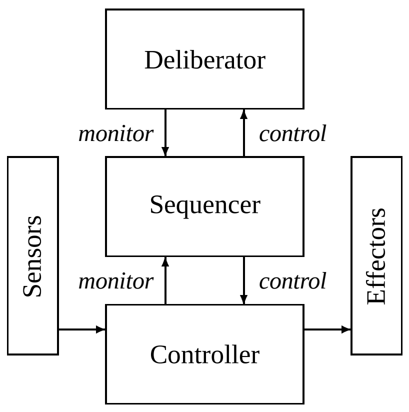


Figure 2.11: The basic architecture of an ATLANTIS agent.

2.4.4 ATLANTIS

ATLANTIS (A Three-Layer Architecture for Navigating Through Intricate Situations) was developed by Gat [97, 98] primarily for controlling mobile robots.

(i) It consists, as illustrated in Figure 2.11, of the following three layers: the *controlling* layer, which is a pure reactive component programmed in ALFA (A Language for Action) and responsible for routine behavior and low-level control of sensors and effectors, the *deliberative* layer, which is responsible for constructing plans and maintaining world models, and the intermediate *sequencing* layer, responsible for constructing controlling sequences of primitive activities and deliberative computations.

(ii-iii) The Sequencer monitors the activities in the Controller and the Deliberator and all activities in both these components are initiated and terminated by the Sequencer.

An important concept underlying the design of the Sequencer is the notion of *cognizant failure*, which are those failures the system is able to detect by itself. Gat suggests that, instead of trying to design algorithms that never fails, we should develop algorithms that “...(almost) never fail to detect a failure.” ([97] p.811) He continues: “There are two reasons for doing this. First, it is much easier to design navigation algorithms which fails cognizantly than ones which never fails. Second, if a failure is detected then corrective action can be taken to recover from that failure. Thus, algorithms with high failure rates can be combined into an algorithm whose overall failure rate is quite low provided that the failures are cognizant failures [127].”

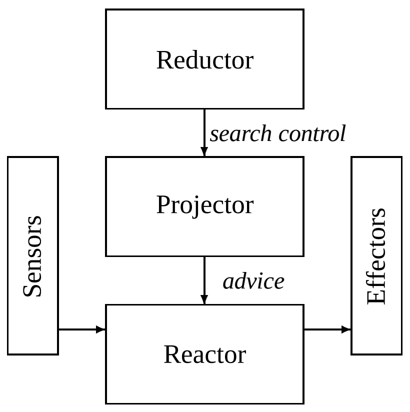


Figure 2.12: The basic architecture of an ERE agent.

2.4.5 The ERE Architecture

(i) The Entropy Reduction Engine (ERE) architecture presented by Bresina and Drummond [37, 71] has the following main components: the *Reactor*, which produces reactive behavior in the environment, the *Projector*, which explores possible futures and provides advice about appropriate behaviors to the reactor, and the *Reductor*, which reasons about behavioral constraints and provides search control advice to the projector. (See Figure 2.12.)

(ii) The Reactor interacts with the Projector and the Projector with the Reductor. The Reactor is able to act independently as a reactive agent, but its behavior will improve by advice from the Projector in the form of Situated Control Rules (SCR's). Similarly the Projector is able to project independently, but advice from the Reductor will reduce its search space and consequently improve its behavior. The Projector can be compared to a traditional planning system in that it through search considers possible future states that are consequences of various possible actions. However, it makes use of a causal theory of its environment which includes: ”...a set of operators which defines both the *actions* that the system can take and the exogenous *events* that can occur in the application environment” [71] (p. 139) In other words, while projecting it takes into account both actions, which are performed by the agent itself, and external events, which are determined by something or someone in its environment. Moreover, the Projector is able to deal with goals of maintenance and prevention (which are temporally extended), in addition to the traditional goals of achievement.

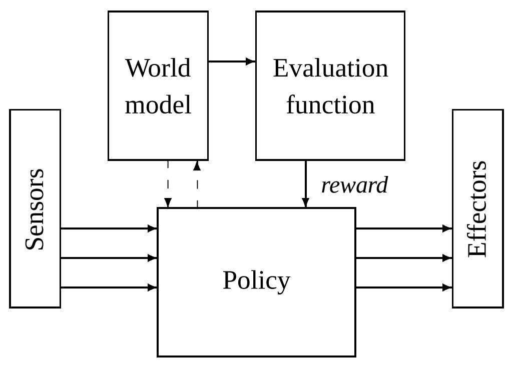


Figure 2.13: The basic architecture of an DYNA agent.

(iii) The Projector is an *anytime algorithm* in the sense that as soon as an advice is found it is given to the Reactor. The Reductor has similar anytime characteristics. However, it is not clear whether the three components are parallel and/or separate processes.

2.4.6 The DYNA Architecture

Also the DYNA architecture suggested by Sutton [262] can be seen as a hybrid, but not entirely layered, architecture integrating reacting and planning. Being very closely related to *reinforcement learning*,[2] it also incorporates learning as an integral part.

(i) DYNA has three main components: the *policy* which is a reactive system, the world model, and the *evaluation function* which maps states to values (see Figure 2.13). It should be noted that due to DYNA's close relation to reinforcement learning, it is required that the world (and the world model) produces a reward after each performed action.

(ii) The planning process, referred to as *relaxation planning* by Sutton, consists of a series of shallow searches (typically of depth one) using the world model. The selection of which actions to perform in the world model during these searches is performed in a random fashion. The purpose of relaxation planning is to continuously adjust the evaluation function in such a way that credit is propagated to the appropriate action steps within action sequences.

[2] DYNA performs, in principle, reinforcement learning, but makes use of internal simulations of actions performed in a world model in addition to the actions performed in the external world.

(iii) Planning is interleaved with acting in a not completely specified way (in Sutton's example they were alternated). However, they cannot be performed in parallel, because when planning DYNA shuts off its real world sensing and acting. Thus, during the planning phase, it is not able react to events in the real world making it very vulnerable.

2.5 Conclusions

There is at least one general conclusion to be drawn from the survey of autonomous agent architectures in this chapter: in order to be able to meet the requirements stated in the beginning of the chapter (i.e., that the agent should be adaptive, robust, tactical, and versatile), it seems necessary that the agent should be able to perform both reactive and high-level reasoning. However, as been pointed out by Wooldridge and Jennings [289] there is a weakness in the existing suggestions for hybrid architectures. They argue that:

> One potential difficulty with such architectures, however, is that they tend to be *ad hoc* in that while their structures are well-motivated from a design point of view, it is not clear that they are motivated by any deep theory. (p. 26)

In order to improve this situation, a layered hybrid approach based on the theory of anticipatory systems will be presented in the next chapter.

Chapter 3

Anticipatory Autonomous Agents

Anticipation is a mental activity that humans, but also some other living organisms, are capable of and one which is frequently practiced. A tennis player, for instance, has to anticipate the trajectory of the ball in order to make a good hit. A stockbroker makes forecasts of stock prices in order to make a good deal. In short, they use knowledge of future states to guide their current behavior. To do this they make use of an internal model of the particular phenomenon. The experienced tennis player's use of his model is probably on an unconscious level and has been learned through tedious sensorimotor training. A novice, on the other hand, has to use his model of the ball's trajectory in a more conscious manner. Similarly, the stockbroker's model is probably on a conscious level, learned through theoretical studies and experience of previous stock prices.

A special kind of anticipatory behavior is when an anticipated undesired situation makes us adapt our behavior in order to avoid that this situation will ever occur. For example, assume that you are going out for a walk and that the sky is full of dark clouds. Using your internal weather model and your knowledge about the current weather situation, you anticipate that it will probably begin to rain during the walk. This makes you foresee that your clothes will get wet which, in turn, might cause you to catch a cold, something you consider a highly undesirable state. So, in order to avoid catching a cold you will adapt your behavior and bring an umbrella when going for the walk.

It is the author's opinion that this kind of anticipatory reasoning, in particular of the latter kind, has not been sufficiently studied and, as a consequence, is not well-understood within the field of intelligent autonomous agents. Moreover, it seems probable that autonomous agents with the ability to anticipate as described above would exhibit novel, interesting and possibly unexpected properties that might enhance their capacity.

In this chapter, a novel way of combining the reactive and the deliberate approaches will be presented. This approach is based on the idea of *anticipatory planning* (first described in paper I) which, in turn, is based on the concept of *anticipatory systems* as described by Rosen [229]. We will begin with a brief description of anticipatory systems followed by a presentation of a general framework for *anticipatory autonomous agents*. (The presentation is mainly adopted from paper V while a somewhat different view of the approach is described in paper VI.) I will then suggest *a linearly quasi-anticipatory autonomous agent architecture* as one reasonable instantiation of this general framework. To support this claim, some promising empirical results from simulations of such agents in both single- and multi-agent contexts are presented (taken from paper IX). A comparison to related approaches and a discussion concludes the chapter.

3.1 Anticipatory Systems

According to Rosen [229], an anticipatory system is "...a system containing a predictive model of itself and/or of its environment, which allows it to change state at an instant in accord with the model's predictions pertaining to a latter instant." (p. 339) Thus, such a system uses the knowledge concerning future states to decide which actions to take in the present.

In more formal terms, the next state of an ideal anticipatory system would be a function of past and future states:

$$s_{n+1} = f(s_1, s_2, ..., s_n, s_{n+1}, ..., s_k), \qquad k > n$$

whereas a causal system depends only on past states:

$$s_{n+1} = f(s_1, s_2, ..., s_n)$$

However, since an agent acting in the real world cannot normally have true knowledge of future states,[1] it is, of course, not possible to implement an anticipatory system in this strict sense. The best we can do is to approximate such a system by using *predictions* of future states. Thus, we have:

$$s_{n+1} = f(s_1, s_2, ..., s_n, \hat{s}_{n,1}, ..., \hat{s}_{n,k-n}), \qquad k > n$$

where $\hat{s}_{n,i}$ is the predicted value of s_{n+i}.

Let us describe a simple class of anticipatory systems suggested by Rosen [228]. It contains an ordinary causal (i.e., non-anticipatory) dynamic system, S. With S he associates another dynamical system, M, which is a model of S. It is required that the sequence of states of M are parameterized by a time variable that goes faster than real time. That is, if M and S are started out at some time t_0, then after an arbitrary time interval Δt, M's sequence of states will have proceeded $t_0 + \Delta t$. In this way, the

[1] An agent acting in a closed world and having a perfect world model, on the other hand, would be able to have true knowledge of future states.

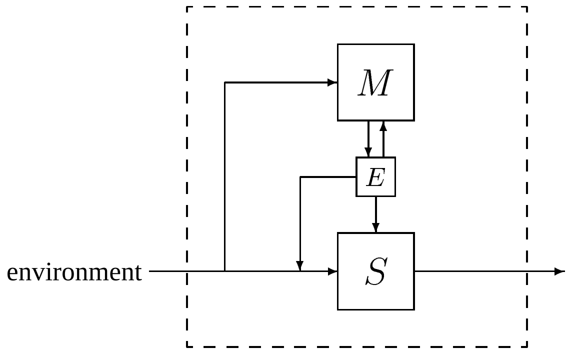


Figure 3.1: The basic architecture of a class of anticipatory systems.

behavior of M predicts the behavior of S: by looking at the state of M at time t, we get information about the state that S will be in at some later time than t. In addition, M is equipped with a set E of effectors which allows it to operate either on S itself, or on the environmental inputs to S, in such a way as to change the dynamical properties of S. If S is modified the effectors must also update M. This class of anticipatory systems is illustrated in Figure 3.1. Rosen argues that such a system would be an anticipatory system in the strict sense if M were a perfect model of S and if the environment were constant or periodic. However, as M in general is not a perfect model of S, he calls the behavior of such system *quasi-anticipatory*.

Then, how should the predictions about the behavior of S be used to modify S's properties? Rosen [228] argues that this could be done in many ways, but suggests that the following is the simplest:

> Let us imagine the state space of S (and hence of M) to be partitioned into regions corresponding to "desirable" and "undesirable" states. As long as the trajectory in M remains in a "desirable" region, no action is taken by M through the effectors E. As soon as the M-trajectory moves into an "undesirable" region (and hence, by inference, we may expect the S-trajectory to move into the corresponding region at some later time, calculable from a knowledge of how the M- and S-trajectories are parameterized) the effector system is activated to change the dynamics of S in such a way as to keep the S-trajectory out of the "undesirable" region. (p. 248)

3.2 Anticipatory Agents

How could we transfer this type of systems into an agent framework? There are some additions and changes to the simple system suggested by Rosen that we have found necessary. First, we need a meta-level component that runs and monitors the model, and evaluates the predictions and decides how to change S or the input to S.[2] Thus, we also include the effectors, E, in this meta-level component that we here will call the *Anticipator*. Second, in order to predict future environmental inputs to S we need to extend the model M to include also the environment. This inclusion is in line with later work of Rosen (cf. [229]).

Thus, in the suggested framework, an *anticipatory agent* consists mainly of three entities: an object system (S), a world model (M), and a meta-level component (Anticipator). The object system is an ordinary (i.e., non-anticipatory) dynamic system. M is a description of the environment *including* S, but excluding the Anticipator.[3] The Anticipator should be able to make predictions using M and to use these predictions to change the dynamic properties of S. Although the different parts of an anticipatory agent certainly are causal systems, the agent taken as a whole will nevertheless behave in an anticipatory fashion.

When implementing an anticipatory agent, what should the three different components correspond to, and what demands should be made upon these components? To begin with, it seems natural that S should correspond to some kind of reactive system similar to the ones mentioned above. We will therefore refer to this component as the *Reactor*. It must be a fast system in the sense that it should be able to handle routine tasks on a reactive basis and, moreover, it should have an architecture that is both easy to model and to change. The Anticipator would then correspond to a more deliberative meta-level component that is able to "run" the world model faster than real time. When doing this it must be able to reason about the current situation compared to the predicted situations and its goals in order to decide whether (and how) to change the Reactor. There is a large body of work concerning different aspects of meta-level reasoning, but none of this work seems readily applicable to the outlined approach. The closest is perhaps the studies on reflective architectures [165] and some works on meta-reasoning architectures in the context of autonomous agents [148]. However, different kinds of meta-level reasoning, such as reflection in anticipatory agents, will be examined in detail by Ekdahl in his (forthcoming) PhD thesis [77].

The resulting architecture is illustrated in Figure 3.2. To summarize: The sensors receive input from the environment. This data is then used in two different ways: (1) to update the World Model and (2) to serve as stimuli for the Reactor. The Reactor reacts

[2]We will in the following regard both these types of changes as changes to S.

[3]The importance of having an internal model that includes both the agent as a part of the environment and (a large portion of) its abilities has been stressed by, for instance, Zeigler [293] and Kohout [145].

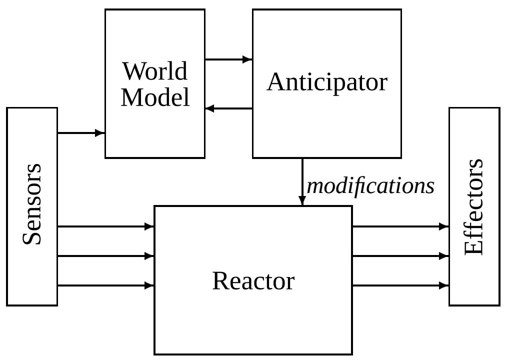


Figure 3.2: The basic architecture of an anticipatory agent.

to these stimuli and provides a response that is forwarded to the effectors, which then carry out the desired action(s) in the environment.[4] Moreover, the Anticipator uses the World Model to make predictions and on the basis of these predictions the Anticipator decides if, and what, changes of the dynamical properties of the Reactor are necessary. Every time the Reactor is modified, the Anticipator should, of course, also update the part of the World Model describing the agent accordingly. Thus, the working of an anticipatory agent can be viewed as two concurrent processes, one reactive at the object-level and one more deliberative at the meta-level.

3.3 A Basic Class of Anticipatory Agents

I have developed A Linearly Quasi-Anticipatory Agent Architecture (ALQAAA) which is a specialization of the general architecture described in the last section. In particular, it adopts Rosen's suggestion of the simplest way of deciding when (and how) to change the Reactor, i.e., by dividing the state space into desired and undesired regions. In this section some initial results from experiments with ALQAAA will be presented.

Some simple ALQAAA-agents and a testbed has been implemented.[5] The problem domain has deliberately been made as simple as possible in order to make the principles of anticipatory behavior as explicit as possible.

[4]Do not confuse these effectors with those discussed above that modifies the object system S.

[5]They have been implemented in the object-oriented language Simula on a Sun SparcStation running Solaris 2.3. Local class packages were used to achieve concurrency (Simlib IOProcesses) and graphical interface to X (WindowPackage).

3.3.1 Agent Implementation

The Reactor and the Anticipator are run (asynchronously) as two separate processes. The Reactor process is given a high priority whereas the Anticipator is a low priority process that runs whenever the Reactor is "waiting" (e.g., for an action to be performed). Since the Reactor is able to preempt the Anticipator at any time, reactivity is always guaranteed. Thus, the Anticipator has to be a kind of *anytime algorithm* [63], or rather anytime process, in that it should always be able to return a result when it is interrupted.[6] The appropriateness of using anytime algorithms in autonomous agent contexts where real-time requirements are common has been pointed out by, for example, Zilberstein and Russell [294] and Bresina and Drummond [37, 71].

The Reactor carries out a never ending cycle of: perception of the environment, action selection by situation-action rules, and performance of action. Rather than having explicit goals, the goals of the Reactor are only implicitly represented in its collection of situation-action rules. The basic algorithm of the Reactor is given in Figure 3.3.

```
procedure REACTOR;
while true do
    Percepts ← Percieve;
    Action ← SelectAction(Percepts);
    Perform(Action);
```

Figure 3.3: The basic algorithm of the Reactor.

The Anticipator, on the other hand, carries out a never ending cycle of anticipation sequences. Each such sequence begins with making a copy of the agent's world model, which as mentioned earlier is a description of the environment containing the agent as a physical entity in the environment, and a copy of the agent's current set of situation-action rules. These are then used to make a sequence of one-step predictions. After each prediction step, it is checked whether the simulated agent has reached an undesired state, or whether it has achieved the goal. If it has reached an undesired state, the Reactor will be manipulated (in some way or another) in order to avoid reaching this state. Thus, this functioning corresponds to that of the simplest kind of anticipatory system suggested by Rosen. The basic algorithm of the Anticipator is given in Figure 3.4.

[6] According to Dean and Boddy [63], the main characteristics of anytime algorithms are that "... (i) they lend themselves to preemptive scheduling techniques (i.e., they can be suspended and resumed with negligible overhead), (ii) they can be terminated at any time and will return some answer, and (iii) the answers returned improve in some well-behaved manner as a function of time." (p. 52)

```
procedure ANTICIPATOR;
while true do
    WorldModelCopy ← WorldModel;
    ReactorCopy ← Reactor;
    UndesiredState ← false;
    while not UndesiredState and not GoalAchieved(WorldModelCopy) do
        Percepts ← WorldModelCopy.Percieve;
        Action ← ReactorCopy.SelectAction(Percepts);
        WorldModelCopy.Perform(Action);
        UndesiredState ← Evaluate(WorldModelCopy);
    if UndesiredState then
        Manipulate(Reactor);
```

Figure 3.4: The basic algorithm of the Anticipator.

It is important to note that since the behavior of the Reactor in each situation is determined by situation-action rules, the Anticipator always "knows" which action the Reactor would have performed in that situation. Also the environment, including all other agents present in the environment, is treated as being purely reactive. Thus, since everything is governed by situation-action rules, the anticipation mechanism requires no search, or in other words, the anticipation is *linear*. It should also be noted that the agent is not limited to have only a singular goal. In a multi-goal scenario, some of the changes (manipulations) of the Reactor should only hold for a limited interval of time (e.g., until the current goal has been achieved). Otherwise, there is a danger that these changes might prevent the agent to achieve other goals. Furthermore, it seems straightforward to generalize the Anticipator algorithm to handling goals of maintenance rather than of achievement (e.g., by removing the "GoalAchieved" condition).

In more formal terms a linearly quasi-anticipatory agent can be specified as a tuple:

$$\langle \mathcal{R}, \mathcal{W}, \mathcal{U}, \mathcal{M} \rangle$$

where

$\mathcal{R}$	is the set of situation-action rules defining the Reactor.
$\mathcal{W}$	is the description of the environment (the world model).
$\mathcal{U}$	is the set of undesired states.
$\mathcal{M}$	is the set of rules describing how to modify $\mathcal{R}$.

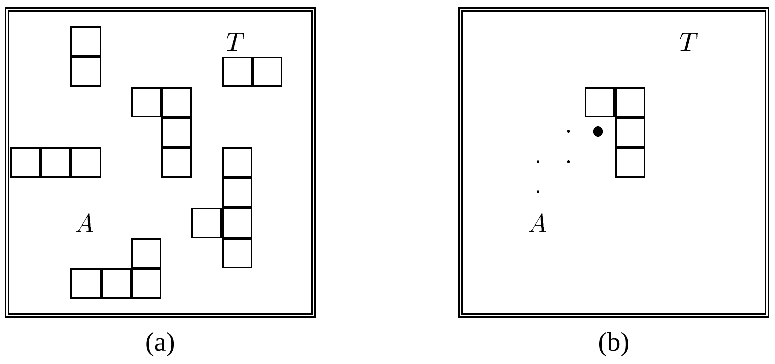


Figure 3.5: A simple maze containing 20 obstacles is shown in (a). A indicates the agent's initial position and T the position of the target. (b) illustrates the behavior of Reactive agent 0.

The Anticipator is defined by $\mathcal{U}$ and $\mathcal{M}$. For each element in $\mathcal{U}$ there should be a corresponding rule in $\mathcal{M}$, which should be applied when an undesired state of this kind is anticipated. Thus, we need in fact also a function, $f : U \rightarrow M$, that determines which rule for modifying the Reactor that should be applied given a particular type of undesired state. However, as this function typically is obvious from the specification of $\mathcal{U}$ and $\mathcal{M}$, it will not be described explicitly. Moreover, in all simulations described below, $\mathcal{W}$ will consist of the positions of all obstacles, targets, and agents present in the environment.

Using these terms, the function Evaluate can be described as checking whether the current anticipated state belongs to $\mathcal{U}$, and Manipulate as first applying f on the anticipated undesired state and then using the resulting rule from $\mathcal{M}$ to modify $\mathcal{R}$.

3.3.2 The Testbed

The agent's environment is a two-dimensional grid (10×10) in which a number of unit-sized square obstacles forms a maze. In addition to these static objects, there are two kinds of dynamic objects: agents, which can move about in the maze, and targets, which can be removed by an agent. Figure 3.5 (a) shows a very simple maze with 20 obstacles (the black squares). The goal of an agent is to pick up the target(s) (marked T in the figure). To be able to pick up a target, the agent must be in the same position as the target. A indicates the agent's initial position. The agent is able to move in four directions (north, south, east, and west), unless there is an obstacle that blocks the way. The agent is always able to perceive the direction to the target, and whether there are any obstacles immediately north, south, east, or west of the agent. (Think of the obstacles as being made of glass: it is possible to see the targets through the obstacles but not the obstacles themselves, which only can be perceived by tactile sensors.)

Almost identical environments have been used in other experiments, for instance by Sutton [262]. It is also similar to the Tileworld [212] (if we interpret the targets as holes) except for that: (i) following Kinny and Georgeff [140] our testbed has no tiles as they only make the simulations unnecessarily complex and (ii) some randomness have been excluded (e.g., the pattern-less appearance and disappearance of holes) since it makes much of the deliberation pointless (i.e., prediction becomes impossible, cf. Hanks [115]). In Section 3.3.4 the environment is generalized into a multi-agent scenario (as suggested by, for instance, Zlotkin and Rosenschein [295]). The advantages and disadvantages of this kind of testbeds have been discussed at length by Hanks, Pollack and Cohen [115].

3.3.3 Single Agent Simulations

In this section some experiments on single agents will be presented. We will compare the performance of reactive agents with that of ALQAAA agents containing such reactive agents as their Reactor. Several different Reactors are compared, beginning with the least powerful.

ALQAAA Agent 0

To begin with, let us consider a simple reactive agent. It has only one simple stimuli-response rule: *reduce the distance to the target if possible.* If there are two possible directions, it chooses the one yielding the greatest reduction in distance to the target. If there is an obstacle that blocks the way in this direction, it tries the other direction that reduces the distance to the target, and if this direction is also blocked the agent gets stuck.[7] A pure reactive agent of this kind will in the situation depicted in Figure 3.5 (b) follow the marked path, and eventually get stuck in the position marked "•".

Let us now consider an ALQAAA agent that has this reactive agent as its Reactor. That is, $\mathcal{R} = \{$*reduce the distance to the target if possible*$\}$. We define the undesired states as those in which the agent is stuck, and if such a state (i.e., position) is detected by the Anticipator, the Reactor is manipulated in such a way that it will not enter this position from now on. That is, $\mathcal{U} = \{$*being stuck*$\}$ and $\mathcal{M} = \{$*avoid this position*$\}$. As a result, the ALQAAA agent will follow the marked path in Figure 3.6 (a). Thus, it will by anticipation detect the undesired position (marked by an dashed box), adapt to the situation by avoiding this position in the future, and eventually reach the target.

[7]Although this seems as a very dumb strategy, many primitive animals behave in this manner.

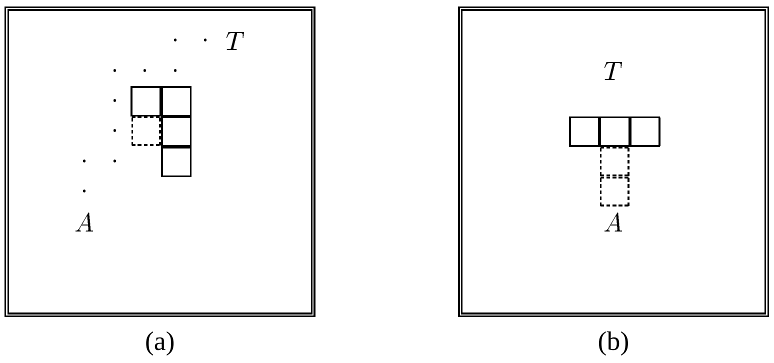


Figure 3.6: The behavior of ALQAAA agent 0 in two different mazes.

Although an ALQAAA agent of this kind is able to find its way in many different types of mazes, it will not be successful in the surprisingly simple one in Figure 3.6 (b). If we to begin with regard only the reactive agent, it will move two steps in the direction of the target (north) and then get stuck in this position (since the only way of reducing the distance to the target is to take another step to the north were there is an obstacle blocking the way). The ALQAAA agent, on the other hand, will detect this position (the upper dashed box) and avoid it. However, when the Anticipator begins a new anticipation sequence it will find that the Reactor this time gets stuck in the position immediately north of the starting position (the lower dashed box). Thus, this position should also be avoided and as a consequence the agent will get stuck in the starting position. The reason for this inferior behavior is that the Reactor prevents the agent to move away from the target.

ALQAAA Agent I

Let us now construct a reactive agent that is only a little more advanced. In contrary to the earlier reactive agent, this one chooses the direction that increases the distance the least if there is not any direction that reduces the distance. Thus, we have that: $\mathcal{R}$ = {*reduce the distance to the target if possible, else increase it as little as possible*}. A reactive agent of this kind will behave as illustrated in Figure 3.7 (a). It will move two positions towards the target and then "loop" between two positions as marked in the figure. If we let this reactive agent be the Reactor in the ALQAAA agent above the behavior will not improve. The reason is that the Anticipator does not regard loops in

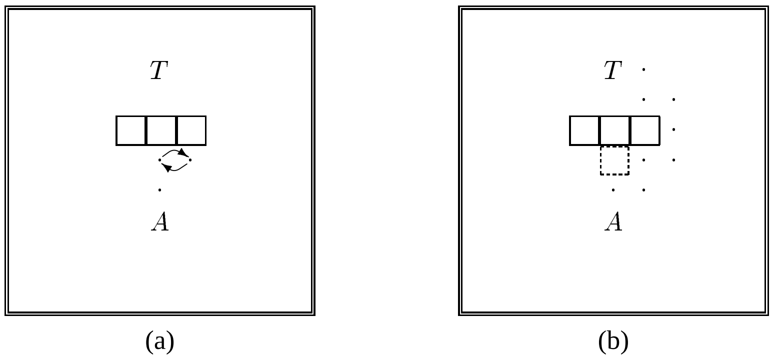


Figure 3.7: The behavior of (a) Reactive agent I and (b) ALQAAA agent I.

general as undesired states. (However, to get stuck in one position can be seen as a special case of being in a loop.)

In order to solve these kind of tasks, we must make a small modification to the Anticipator. Rather than only define the undesired states as those in which the agent is stuck, also states in which the agent is trapped in a loop should be regarded as undesired. That is, $\mathcal{U} = \{$*being in a loop*$\}$ and $\mathcal{M} = \{$*avoid the position in the loop closest to the target*$\}$.[8] An ALQAAA agent of this kind will then behave as showed in Figure 3.7 (b). The Anticipator will detect the loop and make the Reactor avoid the position closest to the target. This is sufficient for the Reactor to find its way to the target.

We have compared the performance of a reactive and an ALQAAA agent of this kind in a series of experiments. There was one target in the environment and the number of obstacles varied between 0 and 35. In each experiment the positions of the agent, the target and the obstacles were randomly chosen. In order to avoid too many trivial scenarios there was also a requirement that the distance between the agent and the target should be at least five unit-lengths. Moreover, only scenarios in which it is possible for the agent to reach the target were selected. From the result in Table 3.1, we see that the more complex the environment gets, the more useful is the anticipatory behavior. If there are no obstacles at all, even the Reactor will, of course, always find the target.

This ALQAAA agent is able to reach the target (when this is possible) in almost all kinds of mazes. However, as Table 3.1 shows, there are some in which it will not succeed and they are typically of the kind depicted in Figure 3.8 (a). The reactive agent will in this case be trapped in a loop whereas the ALQAAA agent will detect this loop

[8] Which position in the loop to avoid can, of course, be selected according to other principles.

No. of obstacles	0	5	10	15	20	25	30	35
Reactive	100%	98%	96%	92%	84%	80%	76%	74%
ALQAAA	100%	100%	100%	99%	99%	98%	97%	97%

Table 3.1: Comparison between Reactive and ALQAAA agents of the number of successful experiments (averages over 200 runs). An experiment is considered successful if the agent picks up the target.

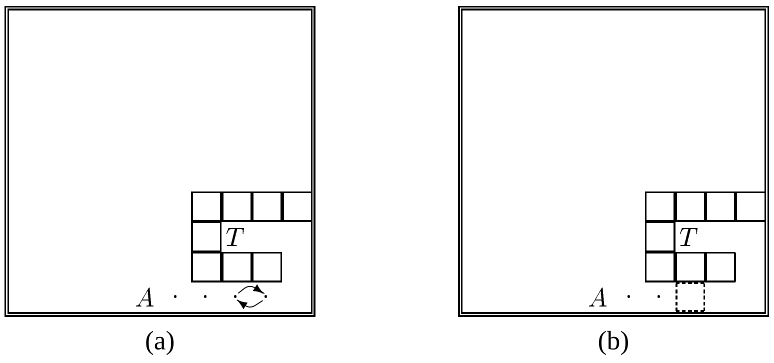


Figure 3.8: The behavior of (a) Reactive agent I and (b) ALQAAA agent I.

beforehand and block the position as shown in Figure 3.8 (b). Thus, the only possible way to the target is blocked and the agent will never reach the target.

The problem with this Anticipator is that it is too eager to block a position. The reason for this is that the Reactor is inclined to "turn around" as soon as it is not decreasing the distance to the target. If we just augment the Reactor's situation-action rule with the condition that it should only change its direction 180° when there are no other alternatives (i.e., if there are obstacles in the three other directions), we will get a reactive agent that solves this problem. This rule together with the $\mathcal{U}$ and $\mathcal{M}$ used in the last example will result in an ALQAAA agent which seems always to reach the target (if the Anticipator is given enough time to anticipate, that is). This Reactor (i.e., $\mathcal{R}$ = {*reduce the distance to the target if possible, else increase it as little as possible, but do not turn around if not forced to*}) will be used in all experiments below.

3.3.4 Multi-Agent Simulations

In this section some experiments in multi-agent domains are presented. We will point out the advantages of being anticipatory, both when competing and when cooperating with other agents. Although the experiments have been carried out with two competing or cooperating agents in order to make things as clear as possible, it would be trivial to extend the experiments to permit a larger number of agents. Moreover, the environments in the figures below do not contain any obstacles. This was not the case in the actual simulations made.

Competing Agents

The main idea here is that an ALQAAA agent should use its knowledge about the behavior of other agents in order to detect future situations in which the other agents interfere with the agent's own intentions (i.e., goals). When such a situation is detected, the Anticipator should manipulate the Reactor in order to minimize the probability that this situation will ever occur.

We will evaluate this approach in the same testbed as above but with two agents and more than one target in the environment. Agent A should be regarded as "our" agent, whereas agent B represents the agent with which it competes. The goal of both the agents is to pick up as many targets as possible. In addition to the basic algorithm described in Figure 3.4, the Anticipator needs a model of the agent B which it uses to predict B's actions in the same manner as it predicts its own (i.e., A's) actions. These predictions can be used in the following way: When the Anticipator realizes that B will reach a target before A, it notifies the Reactor that it should ignore this target. Thus, we have that: $\mathcal{U}$ = $\{$*being in a loop, pursuing targets that presumably will be picked up by another agent*$\}$ and $\mathcal{M}$ = $\{$*avoid the position in the loop closest to the target, avoid the target that presumably will be picked up by another agent*$\}$.

However, let us start with two reactive agents of the kind described above. An environment containing three targets is described in Figure 3.9 (a). If the agents start at the same time the following will happen. Both agents perceive that T_1 is their closest target and they both head towards it. As B is somewhat closer to T_1 than A, it will reach it first and pick it up (see Figure 3.9 (b)). B will then head for T_2 which now is the closest target. A will also head for T_2 following B. B then reaches T_2, picks it up, and heads for the last target T_3 with A still behind. Eventually B will pick up also T_3. Thus, B gets all the targets and A gets none.

If we, on the other hand, let A be an ALQAAA agent and start in the same position as above, it will soon detect that B will be the first to reach T_1. So, A will avoid this target and instead head towards T_3 (which is the next closest target to A). It will reach T_3 at the same time as B reaches T_1. When the agents have picked up their targets,

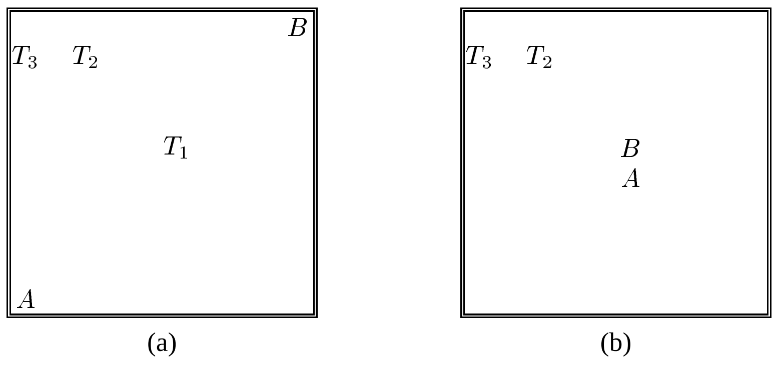


Figure 3.9: The behavior of two competing reactive agents: (a) the initial state (b) the situation after 8 time steps, agent B has picked up target T_1.

there is only one target left (T_2). Since A is closest to T_2, it will reach it first. Thus, by anticipating the behavior of both itself and the other agent, A will in this case pick up two targets whereas B only picks up one.

There are also some quantitative results on the superiority of ALQAAA agents when competing with reactive agents. In the experiments there were 30 obstacles, 5 targets and two agents. In the first session both the agents were reactive and in the second there was one ALQAAA agent competing with a reactive one. The results shown in Figure 3.10 tell us that the performance indeed is improved (being almost optimal) when the agent behaves in an anticipatory fashion.

Cooperating Agents

We shall now see how ALQAAA agents can be used for cooperation. The task for the two agents is to pick up all the targets in shortest possible time. It does not matter which agent that picks up a particular target.

To begin with, we apply the agents in the last example (i.e., A is an ALQAAA agent and B a reactive agent) to the situation described in Figure 3.11 (a). As these agents are not cooperating, their global behavior will (as one might expect) not be optimal. What will happen is that both agents initially head towards the same target (T_1). When agent A reaches T_1 we have the situation depicted in Figure 3.11 (a). The other targets will then be approached in the same fashion, with one agent following the other. As a result, it will take these non-cooperating agents 15 time-steps to pick up all the targets.

Cooperating agents, on the other hand, should be able to make use of the fact that

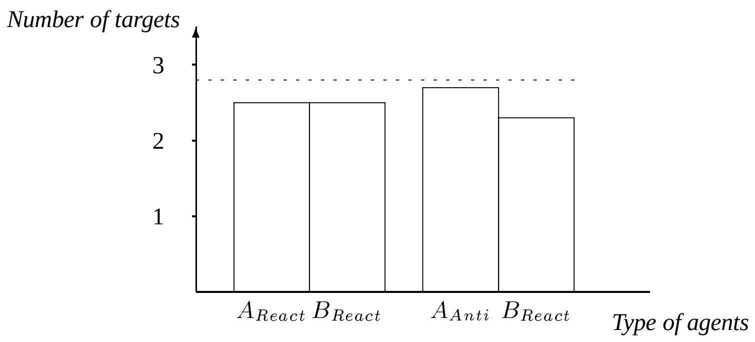


Figure 3.10: Comparison between two sets of competing agents. To the left are both A and B reactive agents and to the right is A an ALQAAA agent and B a reactive agent. The vertical axis indicates the number of targets picked up by an agent (averages over 1000 runs). The optimal number of targets that A is able to pick up in this situation (i.e., given the behavior of B) is illustrated by the dashed line.

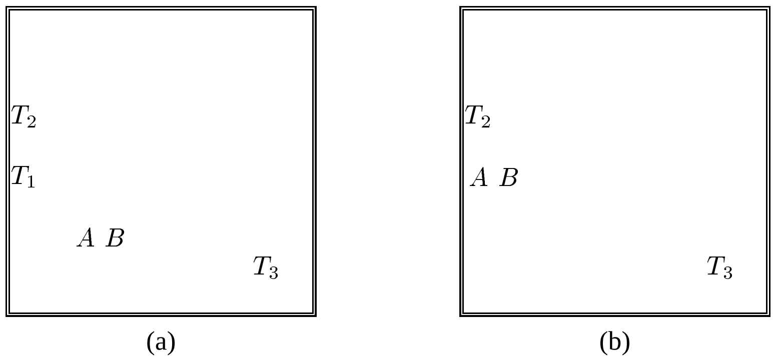


Figure 3.11: The behavior of two non-cooperating agents one ALQAAA (A) and one reactive (B). (a) the initial state (b) the situation after 4 time steps, agent A has picked up target T_1.

the ALQAAA agent knows that it will pick up the two closest targets. One way of doing this is to let agent A send a message to agent B (which still is a reactive agent) when it believes that it will pick up a particular target. This message contains the information that agent B should avoid this target. Thus, we add "*other agent pursuing target that presumably will be picked up by me*" to $\mathcal{U}$ and "*send message to other agent that it should avoid the target*" to $\mathcal{M}$.

When this method is applied to the previous example, agent A will detect that it will pick up targets T_1 and T_2 and sends therefore messages to B that these should be avoided. B then directly heads towards T_3, which is the only remaining target that A will not reach before B. As a result, this system of cooperating agents will use only 6 time-steps to pick up all the targets.

In the scenario described here, it is only agent A that is an ALQAAA agent whereas B is an ordinary reactive agent. Even if we also let B be an ALQAAA agent with the same model of the world, we would not increase the performance. The reason for this is that both agents would have made the same predictions and therefore send messages to each other about things that both agents have concluded by themselves.[9] However, in a more realistic setting where the agents do not have exactly the same information about the world, such an approach would probably be fruitful. In such a case things get quite complicated if one ALQAAA agent simulates the anticipatory behavior of another ALQAAA agent which in turn simulates first agent's anticipatory behavior. The solution I suggest is to simulate only the reactive component of other agents and when an agent modifies its reactive component, it should communicate (e.g., broadcast) information about this modification to the other agents. In this way we are still able to make linear anticipations. This approach can be contrasted with the Recursive Modeling Method (RMM) suggested by Gmytrasiewicz and Durfee [106] in which an agent modeling another agent includes that agent's models of other agents and so on, resulting in a recursive nesting of models.

The quantitative results (using 30 obstacles, 5 targets and two agents) are summarized in Figure 3.12. We see that when the two agents are cooperating they come close to optimal behavior.

Non-Optimal Behavior

As we have seen in Figure 3.10 and Figure 3.12, the behavior of the ALQAAA agents implemented has not been optimal. In Figure 3.13 two examples of non-optimal behavior are illustrated: (a) shows a situation were the agent is enticed to make two un-

[9] As described here, there is, of course, the possibility that B (as it is reactive) gets caught in a loop in the first scenario. Something which could not happen in the second scenario. However, it is possible to modify agent A so that it will also detect when B gets trapped in a loop and send a message to B which position it should avoid.

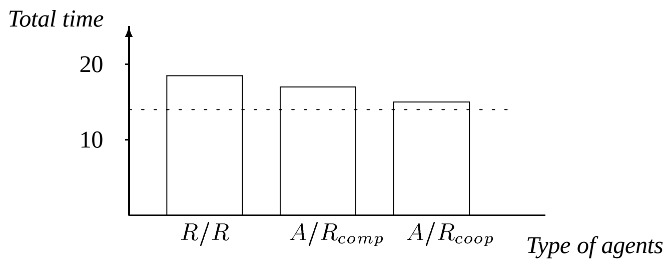


Figure 3.12: Comparison between three sets of agents in terms of the total time it takes to collect all the five targets (averages over 1000 runs). R/R denotes two reactive agents, A/R_{comp}, one reactive and one ALQAAA agent competing, and A/R_{coop}, one reactive and one ALQAAA agent that are cooperating. The optimal time for two agents to collect all targets in this situation is illustrated by the dashed line.

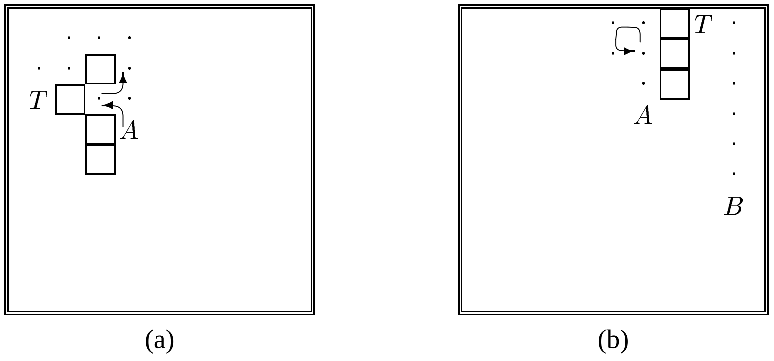


Figure 3.13: Two examples of sub-optimal behavior. (a) A enters a dead-end and takes two unnecessary steps. (b) A draws to fast the conclusion that B will be the first agent to reach T (the figure shows A's world model as it anticipates, B has reached the target and A is just entering a loop but has not yet realized this).

necessary steps, and in (b) when anticipating A jumps to the conclusion that agent B will reach T first (A has not yet discovered that it will be trapped in a loop if it moves to the north) and marks T as a target to avoid. However, we have deliberately chosen very simple Reactors and Anticipators for the purpose of illustrating how easily the performance can be improved by embedding a Reactor in an ALQAAA agent. It should be clear that it is possible to develop more elaborate $\mathcal{R}$, $\mathcal{U}$, and $\mathcal{M}$ components that produce behavior closer to the optimal.

3.3.5 Discussion of the Experiments

We have shown the viability of an approach for designing autonomous agents based on the concept of anticipatory systems called ALQAAA in a simple navigation task. Adaptation to the environment is performed by letting the Anticipator component of the agent manipulate the Reactor component according to anticipated future states.

Compared to traditional planning, anticipation as described here[10] is a more passive way of reasoning. The ALQAAA agent just tries to predict what will happen if nothing unexpected occurs, whereas a planning agent actively evaluates what will happen when a number of different actions are performed. The result is that planning agents will rely heavily on search, whereas ALQAAA agents will not. The main reason for this is that all agents in the environment (also the ALQAAA agent itself) are treated as being reactive.

In addition, it is interesting to note the small amount of heuristic domain knowledge that is given to the Reactor and the Anticipator (i.e., $\mathcal{R}, \mathcal{U}$, and $\mathcal{M}$). Thus, this approach drastically reduces the amount of search needed while at the same time requiring only a small amount of heuristic domain knowledge. Instead, it relies on a linear anticipation mechanism to achieve a more complex behavior.

Anticipation Failures

In more realistic scenarios than the one described above, the anticipation will probably fail now and then. There are two possible reasons for such failures: either the world model, $\mathcal{W}$, is faulty, or the Anticipator, $\mathcal{U}$ or $\mathcal{M}$, is faulty. Failures of the first kind can be detected by comparing the world state at time t+Δ to the predicted world state Δ time-steps ahead at time t. There are two ways of dealing with faulty world models: try to update $\mathcal{W}$, and if this is not possible, adapt (i.e., shorten) the anticipation length to keep the discrepancy between predicted and actual future states sufficiently low to make reliable predictions.

A faulty Anticipator, on the other hand, can only be detected by noting that the agent does not achieve its goal. To do this we must introduce a higher level behavioral component that monitors the behavior of the Anticipator/agent. The simple solution is to let

[10]There may be other ways to anticipate.

a human operator do the monitoring and redesign $\mathcal{U}$ or/and $\mathcal{M}$ when necessary. A more sophisticated, but also much more complex, solution would be to let a new component adapt the Anticipator in a similar way as the Anticipator adapts the Reactor.

Related Work

The main task of the Anticipator is to avoid undesired states whereas the main task of the Reactor is to reach the desired state(s). In other words, the Anticipator's goals are goals of maintenance and prevention rather than of achievement. Compare this to Minsky's suppressor-agents, discussed within his Society of Mind framework [185], which waits until a "bad idea" is suggested and then prevents the execution of the corresponding action. However, there is a big difference, suppressor-agents are not predictive. The Anticipator takes actions beforehand so that the bad idea never will be suggested! Thus, an Anticipator can be regarded as predictive suppressor-agent.

In conformity with the Sequencer component in Gat's ATLANTIS architecture [97, 98], the Anticipator can be viewed as being based on the notion of *cognizant failures* (i.e., a failure that can be detected by the agent itself). However, the Anticipator detects these failures in a simulated reality (i.e., a model of the world), whereas the Sequencer has to deal with real failures.

The notions of Reactor and Anticipator have some relations to the Reactor and Projector components in the ERE architecture suggested by Bresina and Drummond [37, 71]. In particular, the Reactor in ERE is able to produce reactive behavior in the environment independently, but also takes advise from the Projector based on the Projector's explorations of possible futures. However, the Projector is similar to a traditional planner in that it is based on search through a space of possible Reactor actions (a third component, the Reductor, is introduced to constrain this search), whereas the Anticipator simulates the behavior of the Reactor in its environment linearly (i.e., without search). Moreover, the Anticipator's main task is to avoid undesired states, whereas the Projector in the ERE tries to achieve desired states.

There are also some similarities to Sutton's DYNA architecture [262] if we let the Reactor correspond to DYNA's Policy component and the Anticipator to its Evaluation function.[11] In DYNA two kinds of rewards can be identified: *external* rewards, which are those that the Evaluation function gets from the environment (this kind of rewards is not required by an ALQAAA agent), and *internal* rewards, which are those that the Policy gets from the Evaluation function (these can be compared with the manipulations that the Anticipator performs on the Reactor). However, there are many disadvantages with the DYNA architecture compared to ALQAAA agents: (i) The planning process in DYNA requires search (in fact, random search) when it internally tests the

[11] Contrary to DYNA, the Evaluation function (i.e., $\mathcal{U}$) is static (at least in the current implementations).

outcome of different actions. Moreover, several trials are typically necessary whereas ALQAAA agents only need one. (ii) When DYNA plans, it cannot act in the real world and thus cannot guarantee reactivity. The main reason for this is that the planning process shuts of sensors and effectors in order to make use of the Policy component to act in the world model. In an ALQAAA agent, on the other hand, planning and reaction can be performed in parallel since it instead uses a *model* of its Reactor/Policy. (iii) Even if the goal is changed only slightly (e.g., the target is moved one position), the learning in DYNA must start again from scratch. (iv) It is not clear how to implement the Evaluation function. In the initial experiments (cf. [262]), it was implemented using tables where each possible state is represented. This approach is clearly not viable in realistic environments. A more appropriate approach is the one used in ALQAAA agents where only *categories* of undesired states have to be defined, which is often much easier to define than to define complex reward functions.

Limitations of Experiments

The problems that the ALQAAA agents solved above can certainly be solved by other methods, but the point to be made is that we can qualitatively enhance the abilities of a reactive agent by embedding it in an ALQAAA agent. However, there are several obvious limitations to the application presented in this section: (i) the environment is quite static (the only events that take place not caused by the agent itself are those caused by other agents), (ii) the agents have perfect models of the world, (iii) the agents have perfect sensors and the outcome of an action is deterministic, and (iv) this is just a simulation (the agent is neither embodied nor situated) and thus escaping the hard problems of perception and uncertainty.

Moreover, only a single domain has been investigated. Future work includes evaluation of the approach in other domains to see in which types of applications it performs well and whether there are any in which it is not appropriate.

3.4 Related Research

The only experiment on computer-based anticipatory systems known to the author has been carried out by Tsoukalas [272]. He applied an anticipatory approach to the diagnosis and control of a nuclear reactor. His work is, however, not directly comparable to our framework, mainly because it is not addressing autonomous agents. In addition, he makes several simplifications. For instance, the system has no potential for updating its world model or for learning in general. Moreover, his system has no explicit meta-level, or monitoring component (cf. the Anticipator) which we regard as a highly important feature for achieving a more powerful and flexible system.

However, some systems in control theory can, although in a limited sense, be said to show anticipatory behavior. More closely related to our notion of anticipation is some work done within the field of biology.

3.4.1 Anticipation in Control Systems

Some control systems use a method called *feed-forward* to eliminate disturbances that can be measured. According to Åström and Wittenmark [14], ”the basic idea is to use the measured disturbance to anticipate the influence of the disturbance on the process variables and to introduce suitable compensating control actions.” (p. 166) To do this, the feed-forward method requires an adequate model of the process to be controlled.

Thus, the feed-forward method resembles the suggested approach in that a model is used to predict future states and that these predictions are used to guide the behavior of the system. However, the anticipation is limited to the prediction of just one future state (i.e., the error in output that the disturbance would have caused) whereas anticipatory systems are able to make predictions of states arbitrary “time-steps” into the future. Moreover, the process to be controlled is characterized by just a single numerical value, whereas in autonomous agent contexts there is a need for qualitatively more powerful models (i.e., symbolic descriptions).

3.4.2 Anticipation in Biological Systems

It has been suggested that anticipation, in addition to causal relationships, should be taken into account when analyzing biological phenomena (cf. Burgers [46]). Sjölander [245] provides an example of such an analysis: a dog hunting a rabbit uses its internal world model to predict the rabbit’s future positions. These predictions can then be used to focus the attention on the relevant aspects of the situation. This anticipatory behavior can also help in recognition tasks. For example, the dog need not see the whole rabbit all of the time. Since it can predict the rabbits current position, it needs only glimpses of parts of the rabbit to confirm its predictions.

3.5 Conclusions

While initial experiments, where only very simple reactive components (i.e., Reactors) have been used, have shown some promising results, I need to test more sophisticated reactive components in order to evaluate the approach. As another consequence, the manipulations (modifications) of the Reactor made by the Anticipator have been very simple in nature. However, there are some previous work that have dealt with translation of high-level descriptions of behavior into low-level executable form, which can be used for automatic modification and generation of reactive components. For instance, the Projector in the Entropy Reduction Engine (ERE) is able to compile (partial) plans into Situated Control Rules which are sent to the Reactor. Similarly, the GAPPS system of Kaelbling [136] translates goal reduction expressions into directly executable circuits.

Moreover, I have in the experiments used a very limited Anticipator whose manipulation of the Reactor actually can be seen as being governed by a collection of situation-action rules. However, to reason about the dynamical properties of S and the environment in a more powerful way demands an ability to make sense of (i.e, interpret) the symbols used to represent the knowledge about its environment and of its problem solving capabilities (cf. paper VI). This is a very hard problem that will be discussed to some extent in later chapters (and in detail by Ekdahl [77]).

Since an autonomous agent should be able to act in environments that are changing continually, it is important that it can adapt its behavior according to these changes. In the approach suggested there are two ways of making such adaptations: (1) by changing the Reactor, and (2) by changing the world model. As implemented in the experiments on ALQAAA agents, one way of changing the Reactor is to let the Anticipator modify the Reactor's situation-action rules directly based on its predictions. It is, however, also desirable that the Reactor itself has a learning capacity for acquiring low-level stimuli-response schemas. What is more, by letting the Anticipator control the training, the suggested framework seems to have a potential for performing this kind of learning in a more systematic manner than in existing systems.

As the Anticipator will rely heavily on the world model, it is important that it is continually updated as the environment changes. (This topic was not addressed in the experiments described above.) While a considerable amount of work has been done on building and maintaining environment models (the part of the world model that describes the current state of the agent's surroundings) [13, 231], less research has been carried out on updating the more general parts of a world model. However, the representation and updating of these general parts will be one of the main topic in the remaining chapters of this thesis.

Chapter 4

World Modeling

An autonomous agent based on the concept of anticipatory systems, as well as most other hybrid and deliberative agents, will rely heavily on its world model. It is important that the model mirrors the environment as close as possible so that the predictions of future states are as correct as possible. However, since the agent is supposed to act in a highly dynamic environment, it is often not possible for the programmer to provide a complete world model to the agent at design time. Consequently, the agent must be able to update the world model autonomously as the environment changes.

In this chapter some fundamental questions concerning world models will be considered, including: what a world model is, if it is necessary to have one, and whether it is actually possible to construct adequate world models. The chapter also contains brief characterizations of the two AI-fields where the task of world modeling mainly has been studied, namely, *machine learning* and *computer vision*, together with an attempt to clarify the relation between these fields.

4.1 What is a World Model?

In general, a world model is an agent's internal representation of the external world. However, it is important to make a distinction between two kinds of world models: (1) those that only describe the current state of the agent's surroundings, and (2) those that include more general knowledge about other possible states and ways of achieving these states. We will here follow Roth-Tabak and Jain [231] and call models of the first kind *environment models*, and models of the second kind *world models*.

An environment model is typically some kind of spatial 3-D description of the physical objects in the environment. It contains dynamic and situation-dependent knowledge useful, for instance, in navigation tasks. Work on building environment models is described by, for example, Roth-Tabak and Jain [231] and Asada [13]. A world model, on the other hand, typically includes more stable and general knowledge about objects, properties of objects, relationships between objects, events, processes, and so on.

4.1.1 Terminology

One should note that the word "model" has a different meaning in logic and related disciplines than in AI and the Cognitive Sciences. A model in logic is an *interpretation* of a sentence or theory (i.e., an interpretation of some kind of description) that is stated in some formal language. In AI, on the other hand, the word "model" refers to a description, typically represented in some formal language. For reasons of convenience I will here follow the usage in AI and treat "model" as synonymous with "description".

4.1.2 Explicit and Implicit Knowledge

The knowledge of an agent may be represented either *implicitly* or *explicitly*. Implicit knowledge is mainly embedded in the agent's control and sensory processing algorithms. Knowledge is explicit when it is separated from the algorithm that uses the knowledge. This distinction is very similar to the one between *procedural* and *declarative* knowledge. Thus, we can say that reactive agents typically only have implicit, or procedural knowledge, whereas deliberative agents mainly rely on explicit, or declarative knowledge. Implicit knowledge has the advantages of simplicity and efficiency, but at the expense of flexibility. On the other hand, explicit knowledge is complex and inefficient, but flexible and general. Explicit knowledge is easily modified and generalizations can take place over entire classes of entities. In what follows, the term "knowledge" will refer only to explicit knowledge unless otherwise stated.

4.2 Is Explicit Knowledge about the World Necessary?

Does an agent need explicit knowledge about the world at all? Ten years ago this question would have been superfluous in the sense that everybody in the field believed that the answer was positive. However, as described in Chapter 2 some researchers (Brooks and others) have begun to question this belief. To make the discussion more lucid let us formulate the following hypothesis: An autonomous agent needs explicit knowledge about the world to be able to act intelligible in it. By "acting intelligibly" we here mean ability to fulfill the requirements stated in Section 2.1.2 (adaptiveness, robustness, versatility and so on). Followers of the deliberative view believe that this hypothesis is correct, whereas Brooks et al. believe that it is incorrect. The hypothesis can be falsified if somebody actually constructs an agent without explicit world knowledge that acts intelligible. This has not been done (yet).

In the previous chapter some support for the hypothesis was presented. We argued against purely reactive agents, mainly because of their lack of explicit knowledge about the world. One argument was that without a world model (i.e., environment model), it seems difficult to carry out tasks that demand knowledge about objects and other entities not currently perceivable. Moreover, problem solving activities are much harder to perform without explicit knowledge (i.e., world model).

Additional support for the hypothesis comes from biology. It seems reasonable to compare reactive agents to reptiles (and other lower animals) in the sense that their cognition is based on stimulus-response behavior. For instance, Sjölander writes [245]: "There is thus no true intermodality in the snake, just a number of separate systems, involving specific behavioral patterns connected to a specific sensory input." Humans (and other higher animals), on the other hand, are equipped with central representations and Sjölander suggests that this could be an explanation of why reptiles were superseded by mammals and birds. As mentioned in Chapter 2 he argues that this must be one of the most important breakthroughs in the evolution of mind.

4.2.1 On the Proper Level of Abstraction

A world or environment model, or any kind of representation, cannot perfectly describe a given subset of the real world. The only completely accurate representation of an entity is the entity itself, all other representations are only approximations, or abstractions (cf. Davis et al. [60]). Consequently, every representation must be on some specific level of abstraction.

On what level of abstraction should knowledge about the world (especially the environmental model) be represented? Gat [98] argues that the debate concerning deliberative agents versus reactive agents is really an argument about the proper use of world knowledge (or, as he calls it, internal state information). He argues that "...internal state should be maintained at a high level of abstraction and that it should be used to guide a robot's action but not to control these actions directly." (p. 65) Local sensor information, on the other hand, seems necessary for the immediate control.

He provides an example to show that this is also the way humans probably work. We are able to find our house and belongings because we have an environment model at a high level of abstraction. We do not know the exact location of our house nor of our belongings, but we use sensor data to fill in the details that the model does not provide.

4.2.2 Is It Possible to Acquire Explicit Knowledge about the World?

As was pointed out in the beginning of this chapter, it seems unrealistic to assume that complete world and environment models can be pre-programmed into an agent. Thus, we have a situation where an agent must construct adequate models by itself. Is this really possible? The general opinion of AI researchers is that it probably is possible (but difficult). It is certainly possible in extremely simple worlds where light, reflection and other critical conditions can be precisely controlled (cf. the experiments with the mobile robot called "Shakey" at Stanford Research Institute in the late sixties [202]). However, no existing vision system is able to produce environment models of adequate quality in more realistic settings. (The task of learning more general knowledge to be incorporated in the world model is for some reason typically not addressed by the computer vision community.) There are two reactions to this: the optimistic, the most wide-held in AI, which says that it eventually will be possible produce the desired models, and there is the pessimistic, advocated by, for example, Brooks [40] who believes that "complete objective models of reality are unrealistic." The reasons for his pessimism are mainly due to problems with sensors, such as noise, and the inherent complexity of the task [43].

However, here we will adopt a rather optimistic view partly based on the fact that humans indeed are able to produce sufficiently good models of the world (i.e., it is not impossible), and partly on the assumption made earlier that complete detailed models are not needed (i.e., descriptions on a rather high level of abstraction will suffice).

4.3 World Modeling within AI

The task of world modeling is mainly studied within two AI-fields: Computer Vision and Machine Learning. In this section we will concentrate on the assumptions made and the tasks studied, rather than the methods (algorithms) used in these two fields.

4.3.1 Computer Vision

As humans (and most animals) rely to a large extent on vision when learning about the world, it seems reasonable to believe that visual sensors would be an important source for acquiring knowledge to autonomous agents as well. The ultimate purpose of a vision system is, according to Fischler and Firschein [85]: "...to provide the information that allows an organism to interact with its surrounding environment in order to achieve some set of goals."

In analogy with the research on autonomous agents, there exist within computer vision two competing paradigms: one traditional and one new. The traditional approach treats vision as a *reconstruction problem*; the goal is to construct a detailed symbolic representation of the world (i.e., an environment model) independent of the tasks under consideration. The new approach, on the other hand, studies vision from a *purposive* point of view; the goal is to solve particular visual tasks. Most deliberative agents use reconstructionist vision, whereas some reactive agents, Brooks' for instance, use purposive vision. Related to purposive vision is the concept of *visual routines* [274] as used in Pengi [3].

Reconstructionist Vision

It is generally believed that one cannot proceed in a single step from raw camera images directly to a symbolic description. Almost all current reconstructionist vision systems successively transform the scene information through a series of representations.

The initial representation is often an intensity image produced by, for instance, a video-camera. This image is typically processed numerically to produce a 2-D image description (or primal sketch, cf. Marr [166]) that makes explicit important information such as intensity changes. The 2-D image description is then used to produce a 3-D scene description that describes the spatial organization of the scene. The final stage is then to interpret this description in terms of classes of objects in order to form a symbolic description of the scene.

There are, of course, many vision systems that do not use this set of representations. In general, one can say that at the lower levels of representation numerical computation is used, whereas symbolic computation is often used at the higher levels.

Purposive Vision

In contrast to the traditional vision paradigm where the goal is to make a complete description of the environment, the more recent purposive paradigm suggests that you should only describe the parts of the environment that are relevant for the tasks at hand. For instance, if an agent is looking for an object that can be used for a certain purpose, it may only be necessary to recognize some of its qualitative features, not the exact shape. Similarly, if the agent needs to find a path from its current position to a desired position, it does not need to know the exact shapes of all the objects in the environment. From these examples it is clear that within this paradigm, vision is very task dependent (in contrast to the traditional paradigm where vision is task independent). In other words, the goal of purposive vision is to develop different "vision-guided behavior" that can be used to solve different tasks. Moreover, purposive vision mainly makes use of implicit knowledge in contrast to reconstructionist vision which mainly relies on explicit knowledge.

Since this paradigm is rather new, at least within computer-based vision, the terminology is somewhat confusing. The name *purposive vision* is adopted from Aloimonos and Rosenfeld [7], but *active vision* has also been used to label this paradigm. However, also approaches to vision that make use of camera movements to facilitate reconstruction problems (often low- and middle-level vision) (cf. Pahlavan [206]) are sometimes labeled active vision. Yet another related concept is *animate vision* [20] that seems to cover both purposive vision and the latter interpretation of active vision.

An example of the use of purposive vision in the context of autonomous agents, is Herbert's [55] (one of the robots constructed at Brooks' lab) mechanism for the locating of soda cans.[1] Herbert does this by sweeping a plane of laser light up and down. By letting a camera whose scan lines are oriented vertically observe this, the depth can be computed by measuring the distance between the laser line and the bottom of the picture. Herbert considers every object that has "a flat top, and two fairly straight sides of equal length" as a soda can.

Discussion

It is an undisputed fact that the reconstructionist approach to vision has not been very successful. The reason is simply that the problem is inherently complex; any single object can be projected into an infinite number of 2-D images. The orientation of the object in relation to the viewer can vary continuously, giving rise to different 2-D projections, and the object can, in addition, be occluded by other objects. However, it seems that these problems can be solved, at least partially, by using stereo vision and active vision (i.e., making camera movements).

Purposive vision, on the other hand, also has some complexity problems, such as: (i) a very large selection of "vision-guided behavior", possibly one type of behavior for every task, seems to be needed; (ii) it probably will be hard to make decisions between competing behaviors. Additional problems with (and advantages of) the two paradigms were presented at a panel discussion at a recent major AI conference [31]. The conclusion of this discussion was that extreme purposive and reconstructive views are both untenable and that a more pragmatic stance probably would be more fruitful.

In this section, we have only discussed visual sensors. However, since the task studied here is not 2-D image analysis, but actual perception of real objects, there are other kinds of sensors that can also be useful, for instance: range, proximity, force and touch sensors. For a study on multisensor integration refer to, for instance, Lou and Kay [159].

4.3.2 Machine Learning

Machine Learning (ML) is the sub-field of AI in which the automated acquisition of (domain-specific) knowledge is studied. The aim is to construct systems that are able to learn, i.e., systems that improve their performance as the result of experience [238].

[1] Herbert's task is to wander around in office areas, going into peoples' offices and stealing empty soda cans from their desks.

Learning in many different contexts has been investigated, two of the most predominant being classification and problem solving.

The most relevant kind of learning for world modeling is probably *concept learning,* which belongs to the classification domain. The goal in concept learning is typically to learn descriptions of categories that can be used for classifying instances. As input the learning system is typically given a set of *descriptions of instances*. We will discuss concept learning in greater detail in Chapter 9.

4.3.3 Computer Vision – Machine Learning Relation

In reconstructionist computer vision, learning is often restricted to the creation of environment models and is often done by using only pre-programmed object models.[2] Thus, any learning of new knowledge about categories is not performed, resulting in static systems that do not increase their performance in the face of experience. However, some examples of vision systems that actually learn this kind of knowledge will be presented in Section 9.2.2.

Within ML and within concept learning in particular, learning problems where the input is in symbolic form are almost exclusively studied. There is, however, an emerging understanding among researchers in the field that a change is necessary. For instance, Michalski and Kodratoff write [180]: "So far, this input information (examples, facts, descriptions, etc.) has been typically typed in by a human instructor. Future machine learning programs will undoubtly be able to receive inputs directly from the environment through a variety of sensory devices."

One might conclude that there is a natural relation between computer vision and machine learning; the vision system produces symbolic descriptions that the learning system can use as input. However, since these fields are often studied in isolation we cannot be sure that the vision system produces output that can be used by the learning system (regarding to both form and content). Moreover, it is not clear at which level learning should take place. It may be the case that the learning of different kinds of knowledge must be accomplished at different levels. Thus, to integrate ML and computer vision we must in some way smoothen the transition from signals (i.e., analog, subsymbolic representations) to symbols.

From Signals to Symbols

Since most of an agent's knowledge is about its environment, the agent must somehow extract this information from observations of the environment. Since we are dealing with autonomous agents that receive information directly from the environment and

[2]That is, internal representations of the objects, or categories, to be recognized.

process it on different levels by different systems (for instance, vision and learning systems), we seem to need several notions of observation.

It might be useful to follow Gärdenfors [96], who distinguishes three levels of observation descriptions.[3] The highest level is the *linguistic* level, where observations are described in some language (cf. ML). The intermediate level is the *conceptual* level where observations are not defined in relation to some language. Rather, they are characterized in terms of some underlying *conceptual space*, which consists of a number of *quality dimensions*. Some of these dimensions, like temperature, color and size, are closely related to what is produced by our sensory receptors while others, like time, are more abstract. At the conceptual level an observation can be defined as "an assignment of a location in a conceptual space". The lowest level is the *subconceptual* level where observations are characterized in terms of the "raw" (not processed in any way) inputs from sensory receptors (cf. computer vision). This input, however, is too rich and unstructured to be useful in any conceptual task. It must be transformed to suit the conceptual or the linguistic level. To do this the subconceptual information must be organized and its complexity reduced.

Whereas a situated and embodied agent probably has to deal with observations on the subconceptual level, disembodied agents such as softbots receive their information at the linguistic level. Harnad [117] makes a distinction between learning based on perceptual observation which he calls *learning by acquaintance*, and *learning by description* that bases the learning on observations at the linguistic level.[4] Thus, most ML systems perform learning by description, whereas computer vision-based systems, such as situated and embodied agents, are forced to learn by acquaintance.

This three-level perspective also raises several more or less philosophical questions. For instance, is it convenient or even possible, to perform any useful cognitive tasks (e.g., reasoning) on a sub-linguistic level, or do we need some kind of language? If we assume that some kind of language is necessary for reasoning, then observations must be described on some linguistic level in the reasoner. On the other hand, it seems clear that at some (early) stage in the sensors, observations must be described on the subconceptual level.

We can conclude that there are at least two levels of representation (observation) involved here: the linguistic and the subconceptual. Gärdenfors' conceptual level is only one suggestion of an intermediate level. As pointed out earlier, a series of representations are used in reconstructionist vision: the raw camera image (corresponding to the subconceptual level), 2-D image description, 3-D scene description (intermediate levels), and symbolic description (linguistic level). A problem with Gärdenfors' conceptual level is that, while the intermediate representations of computer vision sys-

[3] A similar distinction is made by Harnad [117].

[4] This distinction is probably inspired by Russell's [233], but is not equivalent to his.

tems are designed for representing structural information, it is not clear if and how the conceptual spaces can handle this type of information.

The Symbol Grounding Problem

The symbol grounding problem as described by Harnad [118] concerns the meaning of the symbols in symbol systems. Traditional AI systems manipulate symbols that are systematically interpretable as meaning something. The problem is that the interpretation is made by the mind of an external interpreter. The system itself has no idea of what the symbols stand for. Since this is a problem that arises in all symbol systems, it concerns also the world model of an autonomous agent. However, as described in Chapter 8, this problem will be resolved if we are able to integrate vision and learning in an adequate way and to use appropriate representation schemes.

4.4 Conclusions

As pointed out earlier, it is often assumed within AI that it is possible to modularize the cognition process without much thought; we solve the planning problem, you solve computer vision, and a third part solves the learning problem, and then we meet and build an autonomous agent. As we have noted, many of the underlying assumptions of ML and computer vision are not compatible. Most important is the difference in representation schemes. Whereas ML-systems typically use symbolic descriptions, often in the form of attribute-value pairs, to describe objects, computer vision systems typically use non-symbolical, geometrical 3-D models containing also structural information. On the other hand, learning is essential for the development of vision systems, since the construction of object models by hand is very tedious and often does not produce the desired results. Thus, it is easy to agree with Moscatelli and Kodratoff [191] when they write "...that adaption through the use of machine learning is the key to autonomous outdoor vision systems." The general conclusion is that closer integration between machine learning and computer vision is necessary. The central problem is not only how to make the transition from the subsymbolic data that the sensors output to a symbolic representation, but to decide on which levels different kinds of learning should take place.

Another fundamental problem is that in traditional computer vision there is a strong emphasis on building environment models and thereby ignoring the problem of building world models. It is typically assumed that there is a pre-programmed world model.

A problem that has not been mentioned earlier in this thesis, is how to make the agent focus its attention on the relevant aspects of the environment. It is not computationally tractable to process all the information available through the sensors. The system needs to know what input information is relevant; some kind of mechanism that controls the focus of attention. This gives rise to a further question: on which level(s) is it most appropriate, or even possible, to have such a mechanism? Is the “filter” between the subconceptual and the conceptual level sufficient or do we need further “filters” at higher levels? One suggestion, put forward by Sjölander [245], is that the focus of attention could be controlled by an anticipating higher level. For instance, a dog hunting a rabbit uses his internal world model to predict the rabbit’s future positions. These predictions can then be used to focus the attention on the relevant aspects of the situation. This kind of anticipatory behavior can be of help in recognition tasks as well. For example, the dog needs not to see the whole rabbit all of the time. Since it can predict the rabbit’s current position, it needs only glimpses of parts of the rabbit to confirm its predictions. Actually, these ideas are in line with the anticipatory agents approach described in the last chapter. Notice also that in these cases, the focus of attention is closely coupled to the task at hand.

What if the hypothesis that an agent needs explicit knowledge about the world turns out to be false? Even if world and environment models, contrary to the author’s belief, will turn out to be obsolete, it seems that an agent cannot manage without *concepts* (which actually may constitute a considerable part of a world model). Thus, we can formulate the weaker hypothesis that an autonomous agent needs concepts to act intelligibly. This hypothesis is supported by, for instance, Kirsh [143] and Epstein [78]. The concepts must not be explicitly represented though. In Brooks’ robots for instance, the concepts are present only implicitly in the circuits and the mechanical devices (cf. Herbert’s representation of “soda cans”). However, Epstein argues that there are several advantages with having explicit representations of concepts. For example, it facilitates the organization of knowledge and the focus of attention.

In the remaining chapters of this thesis we will concentrate on the notions of concept and category. Starting (in the next chapter) with some fundamental issues, such as what it means to have a concept, and what possible purposes of concepts there are.

Chapter 5

Concepts and Categories

In the remaining chapters of this thesis we will concentrate on concepts and categories. Since this topic is rather complex, especially in an autonomous agent context, it is useful to divide it into the following sub-topics:

- the functions of concepts
- the nature of categories
- the representation of categories
- the acquisition of concepts.

Given the task of constructing an autonomous agent able to have and to acquire concepts, the only issues of direct interest would be the representation and the acquisition of concepts. However, as will become apparent, the representation is dependent on what functions the concepts should serve. Moreover, the choice of representation is constrained by the nature of the actual categories that they represent. As a consequence of this, and since representation and acquisition are clearly dependent on each other, it is obvious that we should not study either representation or acquisition of concepts without also examining these more fundamental sub-topics.

Each of the next four chapters will be devoted to one of the sub-topics listed above. In this chapter, we will after a terminological discussion concerning the words "concept" and "category", analyse the basic question of what it actually means to have a concept. This analysis is mainly adopted from paper VII.

5.1 Terminology

As with most other terms that are shared between several research fields, the term "concept" has been used in many different ways. In everyday language "concept" often refers just to the *name*, or designator, of a category. However, the main concern here is not the linguistic task of learning the names of categories (although we will also discuss briefly this topic). Rather, the assumption is made that concepts are independent of language, or at least that they can be studied independently of language.[1] The rejection of any assumption of language as a prerequisite for concepts has been made by a number of scientists. Edelman [76], for instance, cites chimpanzees as an example of animals which lack linguistic abilities, but can have, and are able to acquire, concepts.

Instead, we will in what follows use the term in a different way, more in line with uses within cognitive science and AI. Smith [250] provides a typical cognitive science definition: "...a concept is a mental representation of a class or individual..." However, we will here not deal with concepts representing individuals. A typical AI, or rather ML, definition is Rendell's [220]: "The term concept is usually meant as a description of a class (its intension)."[2] What in these definitions are referred to as classes we will here call *categories*, and by category we will mean a set of entities united by some principle(s). Such a principle may be rather concrete, like having similar perceptual characteristics, or more abstract, like having the same role in a theory or having similar functions.[3]

In contrast to these mainstream definitions we have, for instance, more general ones such as Epstein's [78]: "Concept is defined here as some recognized set of regularities detected in some observed world." There are also some rather odd ones like Matlin's [170]: "A concept is a way of categorizing items and demonstrating which items are related to one another." In this definition the term concept refers to a process rather than a representation.

In the light of these definitions, and several others not mentioned, it is not difficult to agree with Heath [121] who suggests that: "the term 'concept' is thus essentially a dummy expression or variable, whose meaning is assignable only in the context of a theory, and cannot be independently ascertained." Thus, before we continue we should make explicit the intended interpretation of the term "concept". In this thesis we will use, and have used, the following definitions:

Definition 1 *A* category *is a class of entities in the world that are united by some principle(s).*

Definition 2 *A* concept *is an agent's internal representation of a category.*

As mentioned above, a principle for uniting entities may be rather concrete, like having similar perceptual characteristics, or more abstract, like having the same role in a theory

[1] By language we mean here an external natural language, not an internal language thought.

[2] Thus, the commonly used term "concept description" is tautologous as a concept is a description.

[3] Unfortunately the terms "category" and "concept" are often confused in the literature. For instance, Michalski writes [177]: "...concepts, that is, classes of entities united by some principle."

or having similar functions. Prototypical examples of entities are objects, events, and situations.

Although these definitions are consistent with the most common uses in cognitive science and AI other interesting suggestions exist. One is provided by Matheus [168] who suggests that: "a concept is a purposeful description of a possibly fuzzy class of objects." This definition is more specific than Definition 2 in the sense that it contains a constraint, i.e., that the concept should be purposeful. As will become apparent, the author completely agrees with Matheus that the purpose, or function, of a concept is important. However, since it is possible to think of a concept without an explicit purpose, we will hold on to the purer definition. The other addition made, as compared to our definition, "possibly fuzzy", seems redundant (i.e., the class of objects may be fuzzy, not fuzzy, or whatever) and is thus unnecessary. On the other hand, in our definition we emphasize that the representation is internal to an agent. This is due to the approach of this thesis: studying concepts in the context of autonomous agents.

5.2 What does it mean to have a concept?

Although Kirsh [142] has pointed out that in AI "...the worry about what it is to have a concept is seldom articulated", there have been some attempts in related fields to state explicitly what it means to have a concept. For instance, in philosophy the following suggestions have been put forward (cf. Heath [121]). To have a concept "x" is:

- to know the meaning of the word 'x'
- to be able to pick out or recognize a presented x, or to be able to think of x (or x's) when they are not present
- to know the nature of x, to have grasped or apprehended the properties which characterize x's and make them what they are.

As we can notice, these conditions are rather vague, especially if we try to apply them to artificial agents. Several questions remain open: What is it for an auotonomous agent, i.e., a computer system, to "know" the meaning of (this will be discussed in some detail below), to "think of", and to "apprehend" something? What is the nature of a concept? Actually, it is only the first part of the second condition, to be able to recognize instances of the class "x", that seems reasonably straightforward.

A proposal that is easier to comprehend, comes from Smith [249], a cognitive psychologist, who suggests that: "To have a concept of *X* is to know something about the properties of *X's* instances." However, it seems that this condition is too weak and underspecified; it seems not to capture the full meaning of "having a concept". Kirsh [142], on the other hand, gives the following view (in AI terms) on the problem:

> We cannot just assume that a machine which has a structure in memory that corresponds in name to a structure in the designer's conceptualization is sufficient for grasping the concept. The structure must play a role in a network of abilities; it must confer on the agent certain causal powers [30]. Some of these powers involve reasoning: being able to use the structure *appropriately* in deduction, induction and perhaps abduction. But other powers involve perception and action – hooking up the structure via causal mechanisms to the outside world. (p. 10)

From this, and the above discussion, it seems appropriate to draw two conclusions.

1. It is perhaps not adequate to treat "having a concept" as a two-valued predicate, i.e., either you do have the concept or you do not. Instead, we should think in terms of a continuum of degrees of having the concept.

2. Rather than trying to state a number of conditions that should be satisfied for having the concept, it seems that a more fruitful approach would be to ask which *functions*, or purposes, (cf. causal powers) a concept should have.

Thus, the more functions a particular agent's representation of a particular concept can serve and the better it can serve these functions, the higher is the degree to which the agent has the concept. The functions of concepts will be discussed in Chapter 6.

We will in the following concentrate on concepts associated with classes of concrete objects, e.g., "chair" and "dog". Thus, we will not discuss concepts of singular objects or more abstract concepts, e.g., concepts associated with events and situations, and Kant's a priori categories. Nor will we discuss concepts referring to *properties* of objects, e.g., red and large. In my opinion, a considerable source of confusion is the fact that all these kinds of concepts are discussed simultaneously (in contrast to treating them separately) in the philosophical literature.

5.2.1 Entity Theories versus Dispositional Theories

An examination of the philosophical literature on concepts reveals that it is possible to distinguish two perspectives on concepts: (1) that they should be seen as entities and (2) that they are just dispositions or capacities. Entity theories identify concepts with individual entities of one kind or another, for instance, "subsistent" word meanings, abstract ideas in the mind, and external unitary forms (cf. Plato). Dispositional theories, on the other hand, suggest that concepts are essentially habits or capacities for: the right use of words, the production of suitable conditioned responses, recognition, or image formation (cf. Hume, Kant).

It is the author's opinion that these views are complementary in the sense that both are necessary to get a complete picture of concepts in the context of autonomous agents.

We need an entity theory since a concept, in some way or another, has to be represented in the memory of an agent, i.e., of a computer. A dispositional theory, on the other hand, is needed because in order to decide how to implement concepts in an autonomous agent, we must know what they should be used for.

Closely associated with entity theories is the old philosophical problem of *universals*, which will be discussed in the next section. In the next chapter we will sketch a dispositional theory of concepts in terms of which functions they should serve.

5.3 The Problem of Universals

When discussing entity theories of concepts it is important to clarify which fundamental view one has on universals. There are basically two camps: realists and non-realists. Realists, such as Plato, believe that universals are non-mental, mind-independent entities that exist in themselves. Non-realists, on the other hand, argue that universals are mental, mind-dependent entities that would not exist if there were no minds. The most common non-realist theory is *conceptualism*, which was suggested by the classical British empiricists, e.g., Locke and Berkeley. According to conceptualism, there is some general mental representation (i.e., concept) that mediates between a word and what that word stands for. A more radical non-realistic approach is *nominalism*, suggested by Hobbes among others, which argues that not even concepts (general concepts) are necessary, i.e., only words are general. A more detailed discussion of these theories is provided by Woozley [290].

As may have been understood by earlier statements, we are here adopting a non-realist stance. Thus, rather than being a priori entities, it is supposed that categories are the inventions, or constructions, of an agent or a collective of agents, used to structure the environment in order to facilitate cognition. Examples of collectively formed categories are “chair” and “ostrich”. More personal categories invented by a particular individual are, for example, “the-things-that-are-mine” and “articles-relevant-for-my-thesis”. Moreover, since we also assume that the agent has a structure (i.e., a concept) in its mind that represents a category, we can classify our view as conceptualistic.

In short, we suggest (as an entity theory) that a concept is an internal representation of a class, or category, of external objects. This category is not an objective, a priori entity, but a construction of some agent(s) in the domain.

5.4 The Meaning of (Symbols Designating) Concepts

One suggested definition of “having a concept” was to know the meaning of the word, or symbol that designates the concept. Although this ability to interpret symbols will not be explicitly discussed in the next chapter’s survey of the functions of concepts, it seems as a fundamental ability that deserves some discussion. Moreover, in paper VI (and to some extent in paper I and V) it is argued that the ability to interpret and reason *about* its own descriptions is desirable, perhaps necessary, if we want to create a more powerful kind of agents than those presently available.

Since the interpretation of descriptions in formal languages has been studied within the field of logical semantics, it seems as a good idea to closer examine the work in this field (especially if we believe that the representations of an agent are best expressed in a language similar to predicate logic). In particular, we will investigate whether it is possible to program an agent to autonomously interpret symbols according to the principles of logical semantics (a task for which these theories were not originally intended).

5.4.1 Logical Semantics

For logicians, semantics is *truth conditional*, i.e., to know the meaning of a logical description, or formula, is to know what the world have to be like for it to be true. Thus, to give the meaning of a description is to specify its truth-conditions in terms of necessary and sufficient conditions. For instance,

‘$\forall x P x$’ is true iff every object in the domain has the property denoted by P.

As Tarski [266] has pointed out, the truth-conditions of a description must be specified in a *meta-language* to the language in which the description is formulated, i.e., the *object language*. In the example above, the object language is first-order predicate logic whereas the meta-language is ordinary English.

However, this seems not to get us very far. As Baker and Hacker [17] suggest, we may ask ourselves: “...the question whether metalinguistic equivalences on Tarski’s model actually connect expressions with reality or whether, on the contrary, they merely constitute a translation manual for rendering expressions in the object language into other expressions in the metalanguage.” (p. 126) In other words, traditional truth conditional semantics attempts to *describe* the meaning of the symbols. However, this only leads to another set of symbols, which would likewise need to be interpreted, and in the end to an infinite regression.

Above we have mainly discussed interpretation of logical *sentences*. But, since the meaning of a sentence is typically defined as a function of the meanings of its constituents, the problem of interpreting constants and predicates has to be solved. To begin with, the semantic value of a constant is an individual, i.e., the *entity* it designates, not another symbol. Consequently, the interpretation of a constant is its assignment to some member in the domain of interpretation. However, as the semantic value of a predicate is a set of individuals, it is predicates that are most interesting to us (since a concept represents a class of objects, i.e., a set of individuals). For instance, it is possible to denote the concept “chair” by the predicate *Chair*.[4] The interpretation of a predicate is then its assignment to a set of members in the domain. Another way to express the difference between the interpretation of sentences and primitive symbols (such as constants and predicates) is that, whereas the meanings of sentences are systematically determined by their composition, the meanings of primitive symbols are arbitrary (cf. Haugeland [119]).

How then, are these assignments actually carried out? This is often explained in terms of an abstract mathematical *model* of those entities in the world making up the semantic values of symbols in the description language. Formally, a model is an ordered pair: $\langle A, F \rangle$ where A is a set of individuals, and F is a function which assigns semantic values of the appropriate kind to basic expressions. However, it seems that there is a tacit assumption that it is the logician himself who actually makes these assignments, much in the same way as the operator interprets the symbols (e.g., the output) of a traditional AI-system. Wittgenstein [285] suggested that the process of analysis of sentences must ultimately terminate in truth-functions of elementary sentences, each of which is composed of symbols incapable of further analysis. Meaning is assigned to these “indefinables” by their direct association with simple objects through “elucidations”. He concluded that these methods of projection cannot be described in language. Thus, the interpretation of symbols designating concepts seems to go beyond the scope of traditional truth conditional semantics. However, if we want to realize an autonomous agent capable of interpreting (some of) its own descriptions we have to implement these methods (at least partly) in some language.

As described earlier, to know the meaning of a logical formula according to truth-conditional semantics is to know the conditions under which the formula is true. But exactly what does it mean to know these conditions? We argued above that just the ability to explicitly *state* the conditions does not get us very far. Instead, we argued that one, in some way or another, must be able to *recognize* the situations when the conditions are true. In a similar vein, Dummett [73, 74] has suggested a *verificationist semantics* where the meaning of a formula consists of those conditions that would *verify* the formula, i.e., specifications of the procedures that would verify the formula. In this case, the meaning of a formula becomes an epistemological concept rather than a metaphysical since this approach requires a set of procedures that when applied recognize whether the formula is true or not.[5] Related to this approach is the notion of *procedural*

[4] In the framework of composite concepts outlined above, *Chair* would correspond to the internal designator of the concept “chair”.

[5] Related to this is ones conception of truth. Dummett [75] makes a distinction between *realism*, which treats truth as an objective property of descriptions independent of human cognition, and *anti-realism*, which suggests that, at least, some fundamental propositions are made true by conditions associated to human recognition mechanisms. Whereas traditional truth conditional semantics assumes the former, we are here suggesting the latter as being more appropriate.

semantics put forward by Johnson-Laird [134, 135] which suggests that the meaning of a symbol consists of a set of procedures that operate on it, for instance, computing its truth value.

5.4.2 The Meaning of Symbols in Autonomous Agents

In present autonomous agents the programmer has typically tried to foresee all possible situations that the agent might find itself in, and to program his own grounding based on his own experiences (which probably are meaningless for the agent as it does not have the same kinds of experiences as humans, e.g., artificial agents do not have the same kind of sensors as humans). However, in order to cope with situations not anticipated by the programmer, the agent must by itself be able to make sense of the symbols used to represent its knowledge about its environment and of its problem solving capabilities (cf. paper VI). Thus, as pointed out earlier, it must be able to interpret and reason about these symbols. In the last section we stressed that the agent must ground the primitive symbols by itself, most notably by constructing its own concepts. As a consequence, the agent will develop subjective and individual concepts.[6]

Moreover, and in line with the view that meaning should be regarded as an epistemological rather than metaphysical concept, Dorffner and Prem [70] have pointed out that: “No objective world has to be assumed, except some causal dependencies between sensory signals and internal states and except for what is expressed in the meta-level representations that have to be part of the system.” How much pre-programmed knowledge of this kind (cf. meta-level representations) an agent must be equipped with is, however, an open question.

Dorffner and Prem also note an important insight that follows from this, namely, that one problem with many of the proposed theories of semantics is that they try to explain meaning as not being tied to individual agents. From the above discussion it seems clear that the agent itself cannot be ignored in a complete theory of meaning. There are, however, newer approaches (other than those presented above) that may not be affected by this criticism. Devlin [66], for instance, has suggested that the root of most of the problems with classical logic is that it is based on the concept of truth. Instead, he proposes a logic based on the concept of information.

5.4.3 Theories of Meaning and Theories of Universals

We can now see that there is a deeper connection between theories of meaning and theories of universals. However, let us first of all make a distinction between the *exten-*

[6] However, it might be able to associate them with the symbol. or word that other agents use to refer to the category of objects that the concepts represent (e.g., by supervised learning), which in the end will make the agent able to communicate with other agents.

sion and the *intension* of a symbol. Whereas the extension of a symbol is those objects which it designates, the intension is often defined as those properties possessed by all and only the objects that the symbol designates.[7]

Traditional truth-conditional semantics is basically an extensional theory of meaning. Intensional theories of meaning such as verificationist and procedural semantics, on the other hand, can be divided into objective intensionalism, which holds that meanings are objective intensions, and subjective intensionalism, which holds that meanings are subjective intensions. Thus, the position that has been sketched here can be characterized as subjective intensionalism; concepts are regarded as subjective intensions of predicates, i.e., individually formed descriptions of classes of objects. Moreover, I agree with Barwise and Etchemendy [24] (cf. their theory of situation semantics) in that predicates should be treated as primitive symbols, and not defined in terms of the set of objects they designate.

To sum up, we can express the connections between theories of meaning and theories of universals as follows: extensional theories of meaning are consistent with a nominalistic view on universals, objective intensionalism with realism, and subjective intensionalism with conceptualism.

5.5 Conclusions

In this chapter we have dealt with some fundamental questions regarding the concept of concept in the context of autonomous agents. The most basic of these was that of defining what it actually means for someone to have a concept. Rather than trying to state a number of conditions that should be satisfied in order to have the concept, it was concluded that having a concept is a matter of degree, which can be defined in terms of the functions the concept can serve. The more functions it can serve and the better it can serve these functions, the higher is the degree to which one has the concept.

The distinction between entity and dispositional theories of concepts was discussed, and it was concluded that they are complementary in that both perspectives are necessary to get a full picture of the concept of concepts. A conceptualistic entity theory and a dispositional theory based on which functions the concept should be able to serve was then put forward.

We also discussed the meaning of concepts, i.e., the problem of interpreting the symbols used to designate concepts, and presented some arguments of why an autonomous agent should have the ability to interpret (some of) its own descriptions. When examining the work carried out within the field of logical semantics, we concluded that since traditional truth conditional semantics requires a human who grounds the mean-

[7] This definition seems to assume a classical view of concepts (see Chapter 8). However, we will here assume a more general notion of intension, i.e., one that is not tied to a particular view of concept.

ing of elementary symbols, i.e., one who assigns objects and sets of objects to constants and predicates, this approach was not appropriate. Instead, a subjective intensionalistic approach based on the grounding of symbols was suggested, which is more in line with the verificationist and procedural approaches to semantics. Finally, we showed that theories of meaning are closely linked with views on universals.

Chapter 6

The Functions of Concepts

This chapter is devoted to the functions of concepts. Here, and in most of the following chapters, we will start with a survey of the research in the cognitive sciences on the current sub-topic of human concepts. This is then followed by a survey of the corresponding work in AI and a discussion that compares the research and tries to draw some conclusions concerning this aspect of concepts in an autonomous agent context.

6.1 Introduction

The importance of investigating the functions of concepts and how they affect the representation and acquisition, becomes even more apparent if we study the following example. It is taken from a paper written by Michalski and some of his colleagues [27] that describes their two-tiered approach for the learning of concepts from examples. The example in the paper concerns the category "chair". One part of the representation suggested by the authors is shown in Figure 6.1. A two-tiered representation consists of two parts. The Basic Concept Representation (BCR) describes the most relevant properties of the category whereas the Inferential Concept Interpretation (ICI) handles, for instance, special cases. Since the ICI is not very relevant to the point I wish to make, it is omitted here. (We will, however, discuss the two-tiered approach later.) The BCR part of the representation says that: a chair is a kind of furniture, its function is to seat one person, it consists of a seat that is supported by legs and a backrest, it often has four legs and is often made of wood and so on.

This seems to be a rather powerful representation that probably can be used for several purposes. It is probably something like this that we want our agent to have. In any case, it seems more appropriate than most other approaches to the representation of categories suggested in the ML literature. But if we study the text in the article ([27]) closer, we find that this description is *not* something that their system has learned. They

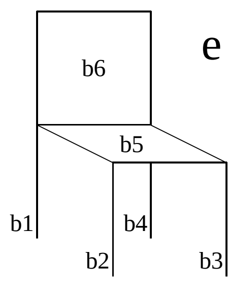
Superclass: A piece of furniture.
Function: To seat one person.
Structure: A seat supported by legs and a backrest attached from the side.
Physical properties: The number of legs is usually four. Often made of wood.
The height of the seat is usually about 14–18 inches from the end of the legs, etc.

(The BCR may also include a picture or a 3D model of typical chairs)

Figure 6.1: The BCR part of a two-tiered representation of the category “chair”.

write: “...(for an example of a two-tiered chair description actually learned, see [26])”.

Before we take a look at the learned description, let me describe the learning situation. The system was given a number of positive and negative examples of chairs. To the left in Figure 6.2 a positive example is depicted. However, the input to the system was symbolic descriptions as illustrated to the right in Figure 6.2. We should notice at this point that the transition from real objects to symbolic descriptions is made in two steps (using two different kinds of representation). To the left in Figure 6.2 we have a 2-D line drawing (a sub-symbolic representation of the actual object) and to the right we have a symbolic representation of this drawing. Both these transitions, which in this case are made by hand, are of course very difficult to accomplish in the form of a computer program.



contains(e,b1,b2,b3,b4,b5,b6),
type(b1) = line, type(b2) = line,
type(b3) = line, type(b4) = line,
ontop(b1 & b2 & b3 & b4, floor),
type(b5) = rectangle,
person_can_sit_on(b5),
ontop(b5, b1 & b2 & b3 & b4),
type(b6) = rectangle,
attached_from_above(b6,b5)

Figure 6.2: A 2-D line drawing and a symbolic description of a chair.

$$\exists\, x\ \exists\, z\ \exists\,(y \geq 3)\ \ [\ \text{person\_can\_sit\_on}(x)\ ]\ \&\ [\ \text{type}(y) = \text{leg}\ ]\ \&$$
$$[\ \text{ontop}(x,y)\ ]\ \&\ [\ \text{attached\_from\_above}(z,x)\ ]$$

Figure 6.3: The actually learned BCR part of a two-tiered representation.

From a number of such symbolic descriptions, the representation in Figure 6.3 was learned.[1] It says that a chair is something that has something that a person can sit on that is on top of at least three legs. Moreover, there should be something that is attached to the "sit-thing" from above.

This seems to be a much less powerful description than the first one. In any case, it contains less information. Why does not the system learn something like the first description, which seems better? The reason is, in fact, rather obvious. The second representation is learned for a certain purpose; it is meant to serve a certain function. Namely, to discriminate between (symbolic descriptions of) chairs and non-chairs. The first description, on the other hand, is probably meant to be more general in the sense that it should be able to serve many functions.

It should be stressed that this does not cause any problems in most traditional AI settings where the learning system is used as a tool (used to improve human performance). Because in this case, discriminating between members and non-members of a category might be just what we want the system to do; it might be precisely the function we want the learned concept to serve. The other functions can be taken care of by the human operator. In an autonomous agent setting, on the other hand, there is no human operator available. Thus, a more powerful way of representing categories, able to support all the desired functions is needed.

Which are these functions that the concepts should serve? Unfortunately, this question is rarely discussed in the AI literature. So, to answer this question we have to turn to literature in cognitive science and philosophy to find out what functions human concepts serve. A survey of this work will then provide a basis for a discussion on what functions the concepts of artificial agents should serve. Much of the material in this chapter comes from paper II and III.

[1] It was also given some background knowledge such as the following rule: [type(x) = line] & [ontop(x,floor)] $\Rightarrow$ [type(x) = leg]. (It seems somewhat inconsistent, however, to use a high-level predicate such as "person_can_sit_on(x)" directly in the description of the example. It would have been nicer if a similar rule had been used to infer it from more basic predicates.)

6.2 The Functions of Human Concepts

To begin with, we can restate some of the very first words of this thesis. Namely, that concepts seem to be the very stuff on which reasoning and other cognitive processes are based. Actually, it is difficult to think of a mental activity that does not make use of concepts in one way or another. However, it is possible to distinguish several functions of human concepts, some of them are:

- stability functions
- cognitive economical functions
- linguistic functions
- communicative functions
- metaphysical functions
- epistemological functions
- inferential functions.

This list is inspired by different work in cognitive science and philosophy, in particular by Rey [223] and Smith [249]. However, we will not always use the terms in exactly the same ways as in these articles.

Concepts give our world *stability* in the sense that we can compare the present situation with similar past experiences. For instance, when confronted with a chair, we can compare this situation with other situations where we have encountered chairs. Actually, there are two types of stability functions, *intrapersonal* and *interpersonal*. Intrapersonal stability is the basis for comparisons of cognitive states within an agent, whereas interpersonal stability is the basis for comparisons of cognitive states between agents.

By partitioning the set of objects in the world into categories, in contrast to always treating each individual entity separately, we decrease the amount of information we must perceive, learn, remember, communicate and reason about. In this sense we can say that categories, and thus concepts, promote *cognitive economy*. For instance, by having one representation of the category "chair" instead of having a representation for every chair we have ever experienced, we do not have to remember that the chair we saw in the furniture-shop yesterday can be used to rest on.

The *linguistic function* is mainly providing semantics for linguistic entities (i.e., words), so that they can be translated and synonymy relations be revealed. For instance, the fact that the English word "chair" and the Swedish word "stol" have the same meaning enables us to translate "chair" into "stol" and vice versa. Furthermore, it seems that it is the linguistic function together with the interpersonal stability function that makes it possible for us to communicate (by using a language). Thus, the *communicative function* is reducible to the linguistic and the interpersonal stability functions.

In philosophy, metaphysics deals with issues concerning how the world is, while epistemology deals with issues concerning *how we know* how the world is. Similarly, we might say that the *metaphysical functions* of a concept are those that determine what makes an entity an instance of a particular category. For example, we can say that something actually is a chair if it has been made with the purpose of seating one person (or something like that).[2] The *epistemological functions* then, are those that determine how we decide whether the entity is an instance of a particular category. For instance, we recognize a chair by size, material, form, and so on. A better example for illustrating this distinction is the category "gold". Something is actually a piece of gold if it has a particular atomic structure. However, when we recognize a piece of gold, we use other features such as: color, weight, and so on. We should note that both these functions are related to categorization: the metaphysical considers what actually makes an entity an instance of a particular category, whereas the epistemological considers how an agent decides whether the entity is of a particular category. However, we should emphasize that for human concepts this distinction is maybe not as clean-cut and unproblematic as described here (cf. Lakoff [149]) but nevertheless it suits our purposes very well.

Finally, concepts allow us to *infer* non-perceptual information from the perceptual information we get from perceiving an entity, and to make predictions concerning it. In this sense, we can say that concepts enable us to go beyond the information given. For instance, by perceptually recognizing a chair we can infer that it can be used to rest on, or by recognizing a scorpion we can infer that it is able to hurt us. This is maybe the most powerful function of concepts; it emphasizes the role of concepts as the central elements of cognition. As Smith [249] writes: "Concepts are our means of linking perceptual and non-perceptual information... they serve as entry points into our knowledge stores and provide us with expectations that we can use to guide our actions." In addition to prediction, concepts allow us to explain relationships, situations, and events.

6.3 Functions of Concepts in Artificial Autonomous Agents

As mentioned earlier, the functions of concepts have almost never really been subject to discussion in AI-literature. The only treatment of this topic known to the author is made by two AI-researchers, Matheus and Rendell, and two psychologists, Medin and Goldstone, [169]. They consider five functions: *classification*, *prediction*, *explanation*,

[2] We use the word "metaphysic" in a more pragmatic way than in philosophy. In our notion, that which makes an entity an instance of a particular category is decided by some kind of consensus amongst the agents in the domain.

communication, and *learning*. The classification function corresponds to our two categorization functions, the epistemological and the metaphysical. Prediction, on the other hand, can be seen as a special case of what we have called the inferential function. This can to some extent also be said about the explanation function.[3] The communication function can be divided into two kinds of functions. One kind corresponds approximately to a combination of what we have called the (external) linguistic function and the interpersonal stability, providing the basis for a shared understanding. The other kind of communication function is that of transmitting the description of the concept to some other representational system, e.g., a human. This function is more relevant for traditional ML-systems in which the learned concepts are intended for human comprehension than for autonomous agents where the concepts are mainly created for internal use. The last function on their list, learning, seems hard to regard as a *function* of concepts.[4] It should, of course, be possible to learn the concept, and the easier this is, the better. (To facilitate learning, on the other hand, could possibly be seen as a function of concepts.) These topics are of great importance and will be discussed in later chapters. However, let us now concentrate on the actual functions of concepts.

In ML there is often an implicit assumption made that the concepts acquired are to be used for some classification task (cf. the example in the beginning of this chapter). Thus, the function of the concepts learned by ML-systems is mainly of an epistemological (or metaphysical, depending on attitude) nature. To see if this is also sufficient for autonomous agents, we will now go through the functions in the previous section one by one, discussing whether they are desirable (or necessary) or not, for an artificial autonomous agent.

The function of intrapersonal stability is of course important, but it is trivial in the sense that it emerges more or less automatically for the agent just by having concepts, independently of the choice of representation. This can also be said about the function of cognitive economy, that is, cognitive economy will emerge as long as we do not memorize every instance of the category.

By analogy to the stability functions, we can say that an agent's concepts can serve both intrapersonal and interpersonal linguistic functions. However, the intrapersonal function is a rather weak one, implied only by the fact that the categories have names internal to the agent (that may be different to the external names). This function is, of course, also trivial in the same sense as above. But what about the interpersonal stability and linguistic functions? They are clearly not necessary in a one-agent scenario. However, if we are interested in a multi-agent scenario with communicating agents, the concepts also must be able to serve these functions.

[3]The explanation function was probably included as a separate function because of the then recently emerged and very influential learning paradigm explanation-based-learning (EBL) [189, 64].

[4]They mainly discuss whether the concept description is adequate for incremental learning. As we shall see later, however, this is a very important question in the context of autonomous agents.

It is, however, the remaining three functions, the metaphysical, the epistemological and the inferential, that are the most interesting, and the ones we will further concentrate on in the remaining part of this thesis. Since an autonomous agent should be able to classify objects in ordinary situations, the epistemological function is necessary.

The metaphysical functions can of course be useful for an agent to have, but in most cases it seems that it can manage without them. In fact, the relevance of this function even for human cognition is not fully understood, for contrasting opinions compare Smith et al. [252] (less relevant) and Rey [224] (more relevant).

Finally, if the agent is to be able to reason and plan about objects it is necessary for it to have at least some inferential functions. This is certainly the central function of concepts corresponding to both the prediction and the explanation function of Matheus et al. [169]. For an autonomous agent, however, the ability to predict future states seems more relevant than the ability to explain how the present state has arisen. For this reason, we will in the following mainly discuss the inferential function in terms of predictions.

6.4 Conclusions

The main insight provided in this chapter was that, when developing a new representation scheme, it is essential to take into account what functions the representation should serve. In the case of concepts for autonomous agents, we singled out five basic type of functions that seemed particularly important: stability functions, cognitive economical functions, linguistic functions, epistemological functions, and inferential functions.

Chapter 7

The Nature of Categories

What can be said about *categories* in general, without taking into account the issues of representation and acquisition of concepts? (Remember that by categories we mean classes of entities in the world and that concepts are an agent's internal representations of these.) As with the functions of concepts, this issue, which we will refer to as the nature of categories, is almost never discussed in the AI literature. In analogy with the last chapter, we will begin with a survey of the psychological and philosophical research on the nature of categories that humans use and then discuss the nature of the categories used by an artificial agent.

In this chapter we will try to identify the most important categories of categories. Moreover, one section will be devoted to a discussion on the nature of properties. In the definition of category we stated that it was a class of entities united by some principle(s). The most important of these principles, similarity, will be treated in detail. Finally, some important issues regarding taxonomies will be discussed.

7.1 The Nature of Human Categories

To begin with, we should make a distinction between categories that we normally use and *artificial* categories. Artificial categories are typically equivalence classes that are constructed for a particular psychological experiment [251]. Typical examples of artificial categories can be found in Bruner et al. [44]. The problem-domain is cards with different symbols. On each card there are one, two, or three symbols of some kind: circle, square or cross. These are colored red, green, or black. Moreover, the cards have one, two, or three borders. A category in this domain is then, for instance, "cards that have red crosses".[1]

[1] It should be noted that a distinction is often made between well-defined categories that can be described by a conjunctive description and ill-defined categories that needs a disjunctive description.

Natural categories, on the other hand, are those that have evolved in a natural way through everyday use. Artificial categories are constructed to be specified by a short and simple definition in terms of necessary and sufficient conditions, while this is often not possible with natural categories. Until quite recently cognitive psychologists have studied the different aspects of concepts using only artificial categories. We who investigate psychological theories in order to build machines able to learn concepts efficiently, find this state of affairs rather unfortunate for at least two reasons: first, machines are already much better than humans in learning definitions that specifies artificial concepts, and second, autonomous agents in real-world environments will have to deal with natural categories, not artificial. However, during the last decades it has become apparent that the study of artificial categories will not significantly increase our understanding of how humans really acquire natural concepts, i.e., representations of natural categories. Therefore, some researchers have begun to use natural categories for their experiments. This movement toward a more sound approach is further elaborated by Neisser [199].

Members of natural categories are either concrete, such as physical objects, or abstract, such as events or emotions. As mentioned in Chapter 5 we will here concentrate on concrete object categories. While dealing with autonomous agents trying to learn about their environment, this is a quite natural initial assumption; the environment consists of physical objects. However, at some point in the future it will certainly be necessary to introduce more abstract categories such as categories of events.

As we will see later, humans also use other types of categories that cannot be classified as natural, namely *derived* [248], or *ad-hoc* [23], categories. Moreover, natural categories can be divided into *natural kinds* and *artifacts* [248]. However, let us begin by examining a more fundamental topic: the properties of objects.

7.1.1 Properties

In both cognitive science and AI, it is generally assumed that the basis for the representation and categorization of an object should be a set of properties (i.e., features, attributes) that characterize the object. But what exactly is a property? Is it something we ascribe to an entity or does it really exist in the entity? So, in analogy with our treatment of universals, we have to decide whether we are realists or not with respect to properties. By arguments similar to those presented then and as a consequence our conceptualistic stance it seems reasonable to adopt a non-realists view. Thus, I argue that properties are constructions, or inventions of an agent or a set of agents, and that they are developed in order to simplify classification and characterization of categories and objects.

Where do our knowledge about properties of objects comes from? It is generally believed that most, if not all, of this knowledge originates from different kinds of per-

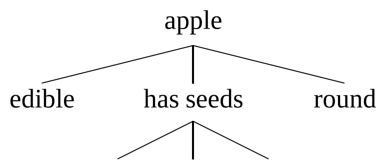


Figure 7.1: Part of the property-hierarchy of the category “apple”.

ceptions. Moreover, some high level features are constructed from lower level features delivered by the perceptual system (cf. Wisniewski and Medin [284]). A general opinion is, however, that there are fundamental differences between different kinds of properties. Several ways of dividing properties into classes have been suggested.

Most properties can be measured according to a *scale*. Although Shepard [241] has presented a typology of scale-types containing eight different types of scales, we will here only distinguish between three kinds of scales. In a *nominal* scale, a property is assigned to a value from an unordered set of mutually exclusive, labeled values. For instance, the weather can be described as sunny, cloudy, rainy and so on. A special case of nominal scale is the *binary* scale, which only has two values. In an *ordered* scale, on the other hand, there is a rank order of the different values. For instance, temperature can be measured in terms of hot, mild, and cold. Finally, a *numeric* scale is an ordered scale in which the values are numeric. For instance, temperature described in terms of $^\circ$C. Properties measured according to a particular type of scale are often referred to as properties of this kind (e.g., weather is a nominal property), although many properties can measured according to more than one scale. Another common distinction is made between *qualitative* properties (i.e., nominal or ordered) and *quantitative* properties (i.e., numeric) that sometimes are called dimensions. It is in principle possible to transform any numeric property into an ordered property and any ordered into a nominal. For example, the values for any dimension can be expressed as a set of nested features [15]. When we go in the opposite direction, however, we may lose some aspects that we usually ascribe to quantitative properties. For example, that a dimension should have the notion of betweenness (cf. Smith and Medin [251] p.13).

Moreover, some properties are called *perceptual*, in the sense that they (in some sense) are directly available from the perceptual system, while others are regarded as being more *abstract*, such as functional properties (cf. Smith and Medin [251] p.18). Furthermore, some features are *global* (e.g., a chair is made of wood) whereas others describe *structural relationships* such as parts (e.g., a chair has legs).

As illustrated in Figure 7.1, what is considered a feature is relative in the sense that

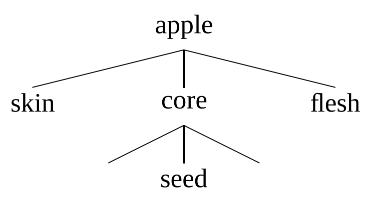


Figure 7.2: Part of the partonomy of the category "apple".

some features correspond to categories themselves, or, at least, can be regarded as principles that unite categories (cf. Definition 1). As a matter of fact, we have a kind of tree-hierarchy of categories. (Note that only a small part of the hierarchy is shown in the figure. Apples obviously have more than three properties and so on.) Smith and Medin [251] make a distinction between the *identification procedure* and the *core* of a concept. They argue that some of the leaves (i.e., the branch end-points, in this case, "edible" and "round") of the resulting tree structure are perceptual features ("round")[2] are those that we normally use to determine which category an object belongs to. In their terminology, they constitute a considerable part of the identification procedure. The core of the concept is the features on the second level (edible, has seeds, round). Thus, the identification procedure is closely related to what we have called the epistemological function whereas the core together with the rest of the features is more connected to the metaphysical and inferential functions. Or, like Smith [249] puts it: "When reasoning we use the cores, when categorizing we use the identification procedures." (p. 29)

The structure in Figure 7.1 is not entirely consistent, though. It is primarily the features that represent parts (i.e., has seeds) that can be thought of as categories.[3] The functional features (i.e., edible) and perceptual features are best thought of as just features. In Figure 7.2 we have a tree-structure, or *partonomy* [268], that describes the different parts of an instance of a category.[4]

7.1.2 Natural Kinds

It seems natural to assume that categories emerge as a consequence of the correlational structure of the environment, i.e., the perceived properties of the instances of a category

[2]Whether "round" is a perceptual feature or not is open for discussion, but let us suppose that it is.

[3]Actually, it is "seed" that is the category.

[4]Note that this hierarchy does not reveal how the different parts are fitted together. To do that we need a structural description. Moreover, do not confuse "core" (i.e., the part of the apple where the seeds are located) with the core of the concept.

make them stand out as a natural class, distinct from other categories. For instance, take the situation where you encounter an elephant for the first time (supposed that you have not read, or been taught, anything about elephants). Then, because of its distinct perceptual features, you create a new category. Moreover, if you see another elephant you decide that it belongs to the same category because of the features it shares with the first one. Quine [216] has termed this type of categories *natural kinds*. Rosch and her colleagues [227] also emphasized that natural categories emerge in this way, assuming that the environment constrained the categorizations, in that human knowledge could not provide correlational structure where there was none at all.

However, it is a rather strong metaphysical claim, and somewhat inconsistent with our non-realist stance, to argue that there exist only objective categories in the world. We must remember that all human categorization depends (at least partially) on human physiology: observations on the perceptual level are furnished by the sensory projections of objects, whereas observations on the linguistic level are furnished by symbolic statements about objects that in turn are furnished by sensory projections of these objects. Thus, a more sensible and somewhat weaker epistemological claim would be that some categories "through human perception" stand out as natural categories.

7.1.3 Similarity

The natural kind categories seem to depend on a notion of *similarity*, where similarity is a relation between two objects. Similar objects are grouped together to form a natural kind category. This state of affairs forces us to analyze the concept of similarity and how it can be measured. As should be evident from the last section, perceptual similarity (i.e., mainly structural similarity) is probably the most important kind of similarity when forming natural kind categories. Unfortunately, this has not been the kind of similarity typically studied by cognitive psychologists. Instead, it is often assumed that objects are described by attribute-value pairs (nominal, ordered, or numeric), i.e., the observations are on the linguistic level.

The theoretical treatment of similarity has been dominated by two kinds of models: *geometric* and *set-theoretical*. In these models two objects are regarded as similar if the difference between the properties used to characterize them is small. A fundamentally different approach is to regard objects similar if they are related by some kind of transformation taken from a set of specified transformations (cf. Kanal and Tsao [137]). In this case, objects that do not share many properties may in fact be regarded as more similar than objects that share a larger number of properties.

Geometric models (cf. Shepard [242]) tend to treat all properties as quantitative. An object is represented as a point in the coordinate space defined by the dimensions used to describe the object. The similarity between two objects is then measured, or defined by the metric distance between them, i.e., the closer they are, the more similar they are.

However, pure geometric models are inadequate for several reasons, for instance:

- The measure is only meaningful if all the selected properties are relevant for describing perceived object similarity [182].
- All selected properties are given equal weight [182].[5]
- Different properties may be based on incommensurable scales [279].
- It is more appropriate to represent some properties as qualitative [273].

Moreover, as Webb [279] points out: "...even within a single ordinal attribute, there is no guarantee that the scale employed should be linear."

The geometric models seem related to Gärdenfors' conceptual spaces described in Chapter 4. Let us try to make this relation explicit. It is possible to interpret the subconceptual level as a (low-level) feature space of a high dimensionality. Thus, it can be said to correspond to a "pure" geometric model. A conceptual space can then be seen as the resulting space when the two first problems above have been taken care of, corresponding to a "refined" geometric model.

In set-theoretical models, on the other hand, objects are represented as collections of qualitative features. The most well-known set-theoretical model is Tversky's [273] *contrast model*. It expresses the similarity between two objects as a linear combination of the measures of their common and distinctive features. However, pure set-theoretical models such as Tversky's have, more or less, the same problems as geometric models. They do not specify how relevant attributes are selected. The attributes are weighted, but how this is done is only loosely specified.[6] Moreover, by merely changing the weights any two objects can be arbitrarily similar or dissimilar. Finally, it is probably true that it is more appropriate to represent some features as quantitative rather than qualitative.

A major disadvantage with both set-theoretic and geometric models is that they cannot handle *structural similarity*. An entity can be structured in two different ways: either it is composed out of parts that themselves have separable properties, or out of parts that have identifiable relations to each other. Structural similarity, which is strongly related to perceptual similarity, has largely been ignored by the cognitive psychologists.

[5]There are some geometric models make use of weighted properties. One example is the one used in Nosofsky's generalized context model [203]. However, there are no model for *predicting* the values of the weights, instead they are computed from experimental results *after* the experiments have been conducted and are for this reason of little value for us.

[6]That the features in fact must be weighted seems to be implied by the theorem of the ugly duckling (cf. Watanabe [278]). This theorem, which is formally proved, shows that whenever objects are described in terms of logical predicates, no two objects can be inherently more similar than any other pair. In other words, for similarity to be meaningful, the predicates describing an object must be censored or weighted.

However, Goldstone [109] has suggested a primitive model for structural similarity called the Similarity as Interactive Activation and Mapping (SIAM) model, which is based on methods similar to those used in analogical reasoning for creating correspondences between the parts/features of the two objects[7]

As we have seen there are problems with "pure" similarity models, especially with the selection of relevant features. Schank and his colleagues [237] go one step further by stating that a "simple" theory for specifying the relevant features is impossible. Mainly because the relevance of features depends on the goals of the agent having the concept. They conclude:

> The process of determining which aspects of instances to be generalized are relevant must be based on an *explanation* of why certain features of a category took on the values they did, as opposed to other values that might a priori have been considered possible. (p. 640)

This suggests that the categories that humans normally use not always arise in the purely *bottom-up* fashion [124] described above. Thus, even the weak claim that categories "through human perception" stand out as natural categories may be too strong, not covering all natural categories. Even Rosch [225] admits (taking back her earlier strong claim) that some types of attributes present a problem for these claims. For instance, there exist attributes that appear to have names not meaningful prior to knowledge of the category, e.g., "have a seat" – chair. Moreover, there exist functional attributes that seem to require knowledge of humans, their activities, and the real world to be understood, e.g., "you eat on it" – table. From these examples she concludes: "That is, it appeared that the analysis of objects into attributes was a rather sophisticated activity that our subjects (and indeed a system of cultural knowledge) might well be considered to be able to impose only *after* the development of the category system." Moreover, she states that attributes are defined in such a way that the categories, once given, would appear maximally distinct from one another. Similarly, Murphy and Medin [194] have claimed that people's intuitive theories about the world guide the representational process. They placed the demand on categories that they must exhibit something called *conceptual coherence*. A coherent category is one "whose members seem to hang together, a grouping of objects that makes sense to the perceiver."

However, this emphasis on theories and explanations in category formation should not be exaggerated. Goldstone [108], for instance, argues that: "...there is much evidence that people form categories before they have developed full theories for the categories. In fact, it is the act of grouping items together in a category, on the basis of lower-level similarities, that promotes the later discovery of higher-level theories." (p.

[7] Goldstone actually speaks about scenes rather than objects when describing the method. However, it seems straight-forward to apply it also to (descriptions of) objects.

148) He also argues that categories based on similarity are the most useful since they allow us to make most inferences, i.e., they maximize the inferential function.

To sum up, the problem with a purely "syntactical" model of similarity is that it ignores both the perceptual and the theory-related constraints that exist for, at least, certain kinds of categories. However, an actual perceptual system of an embodied autonomous agent will have some built-in constraints that determine what will count as an attribute and the salience (weight) an attribute will have. This topic seems closely related to what Harnad [117] has labeled *categorical perception*. He writes:

> For certain perceptual categories, within-category differences look much smaller than between-category differences even when they are of the same size physically. For example, in color perception, differences between reds and between yellows look much smaller than equal-sized differences that cross the red/yellow boundary... Indeed, the effect of the category boundary is not merely quantitative, but qualitative. (p. 535)

Thus, in an autonomous agent the perceptual constraints, such as categorical perception, are determined by the physical properties of its sensors. Goldstone [108] seems to agree and adds another constraining factor: "Similarity is constrained by our perceptual system and by the process for integrating multiple sources of information." (p. 151)

7.1.4 Derived Categories

As pointed out earlier, natural kind categories arise in a bottom-up fashion. In contrast, *top-down* category formation is triggered by the goals of the learner. The categories formed in a top-down manner are often characterized in terms of functional features, whereas bottom-up categories are characterized in terms of their perceptual features such as structure and color. Thus, as Corter [57] points out, the two types of categories seem to be characterized by different kinds of features and feature relationships. Bottom-up categories tend to group instances that share co-occurring properties (i.e., they are "similar"), whereas top-down categories often consist of disjunctive groupings of different types of objects that may not share many properties (i.e., they do not have to be "similar"). Such categories are often referred to as *disjunctive* categories.[8] For instance, the category "things-in-my-apartment" may include such things as records, books, chairs, apples, and so forth.

Barsalou [23] suggests that many of the top-down categories, which he calls ad-hoc categories, do not have the same static nature as bottom-up categories. While bottom-

[8]We do not argue that some categories are disjunctive per se. (It would have been inconsistent with our non-realist view of properties.) The grouping of disjuncts is carried out by an agent or a set of agents.

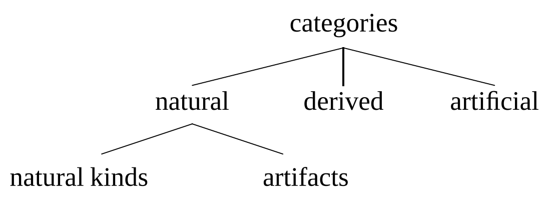


Figure 7.3: The relationships between the different kinds of categories.

up categories generally are believed to be represented by relatively permanent representations in long-term memory,[9] he states that "many ad-hoc categories may only be temporary constructs in working memory created once to support decision making related to current goal-directed behavior." As an example of an ad-hoc category he takes "activities to do in Mexico with one's grandmother". However, there also are some permanent top-down categories such as "food".

7.1.5 Artifact Categories

Not all natural categories are natural kinds. A natural division can be made between *species* (i.e., natural kinds) and *artifacts*. Rosch's examples above, "chair" and "table", which certainly are natural categories, are typical artifacts. Characteristic for artifacts is that they are made by humans to have a certain function, implying that they should be characterized in terms of their functional features. However, it seems that the instances of most artifact categories also have structural, and thus perceptual, similarities, e.g., most chairs look like each other. Moreover, some objects made for one purpose may be used for another purpose, it is for instance possible to use most chairs as tables. Thus, we can say that artifact categories differ from natural kinds in that they seem to have the potential to arise both in a bottom-up and a top-down fashion.

We can now summarize the discussion of different types of categories by suggesting a classification of categories. Figure 7.3 illustrates the hierarchical relationships between the different types of categories described above. First we have the natural categories that have evolved through everyday use. These are either natural kinds or artifacts. In contrast to natural categories there are the artificial categories, often constructed for a particular scientific experiment. In addition to these we have the derived, or ad-hoc, categories that are typically formed during problem solving activities.

[9]The representations can, of course, be modified but they are permanent in the sense that there always exists a representation of the category.

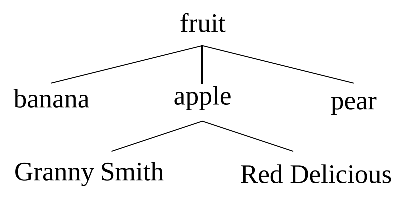


Figure 7.4: Part of a taxonomy of fruits.

7.1.6 Taxonomies

Categories can also be hierarchically organized in a different way than by their properties or parts, namely, in taxonomies. A taxonomy is a hierarchical classification scheme that shows how categories are divided into sub-categories. A part of a taxonomy is illustrated in Figure 7.4. (Figure 7.3 is, in fact, also a taxonomy.) However, this is a rather strong idealization since some categories may not belong to any taxonomy at all while others belong to several.

Apart from organizing knowledge, taxonomies also serve an important function by promoting cognitive economy. How this is possible is demonstrated by Figure 7.5 which shows a part of the fruit-taxonomy of Figure 7.4 augmented with some features of the categories. By noticing that categories on one level inherit the features from its parent category (i.e., the category on the level above), it is possible to reduce the amount of information that must stored on each level. For instance, if we know that apples are fruits and that fruits are sweet, we do not have to remember that apples are sweet. As a result of this inheritance mechanism, only the italicized features in the figure have to be memorized.

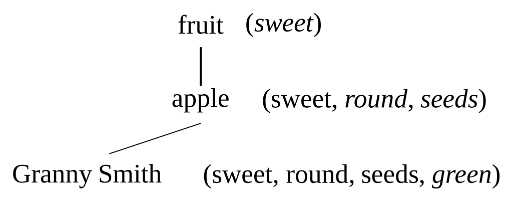

fruit (*sweet*)
apple (sweet, *round*, *seeds*)
Granny Smith (sweet, round, seeds, *green*)

Figure 7.5: Part of a part of a taxonomy of fruits augmented with features. By inheriting features from parent categories, only those in italics have to be memorized.

Rosch et al. [227] argue that there exists a “basic level” in these taxonomies. They write: “Basic categories are those which carry the most information, possess the highest cue validity[10] and are thus, the most differented from one another... Basic-level categories possess the greatest bundle of features... Basic objects are the most inclusive categories which delineate the correlational structure of the environment.” In our taxonomy of fruits (Figure 7.4) bananas, apples and pears constitute the basic level.

The basic level has some interesting properties that have consequences for both the epistemological and the inferential functions. Since the basic level is the one we prefer for categorization, the epistemological function is, in some respect, maximized at this level. In addition, the inferential function is maximized at the basic level. The basic categories have “the greatest bundle of features” and many of these features are distinctive, permitting us to infer a substantial number of properties without much perceptual effort. In contrast, superordinate categories (e.g., fruit) have relatively few properties and hence cannot enable us to make that many inferences. Although subordinate categories (e.g., Granny Smith) have many properties they have so few distinctive properties that they are more difficult to categorize perceptually. Moreover, as should be clear from the last sentence of the citation of Rosch and her colleagues above, basic level categories are the most inclusive level at which conjunctive categories appear. Superordinate categories are generally disjunctive. Finally, we should note that the question of which level is actually the basic level is context dependent in the sense that the normal basic level is the most appropriate in most situations but not all.

7.2 The Nature of AI Categories

In early ML research, categories were often presumed to be artificial; they were often constructed for a particular experiment and it was assumed that all relevant aspects of a category could be summarized by a simple definition in terms of necessary and sufficient conditions. However, today it is mandatory to test and evaluate concept learning algorithms on real-world data, forcing them to deal also with natural categories.

7.2.1 Properties

In most existing concept learning systems, objects are described by a list of attribute-value pairs. Typically, these attributes represent only global properties. Some systems, however, are able to handle also structural properties (cf. [69, 268]) that in contrast to

[10] The cue validity of a feature F with respect to a category C is the validity with which F is a predictor of this category. The cue validity of an entire category may be defined as the summation of the cue validities for that category of each of the attributes of the category.

global properties involve information about the object's parts and relationships among these parts.

An assumption often made when constructing concept learning systems, is that attributes are atomic units of description that are given to the system and are to be used as building blocks of concepts without further processing. Some systems, however, do create new attributes not present in the input data. They try to generate high-level features from the lower level features originally used to characterize the instances. This task, or problem, is often labeled *constructive induction* or the *new-term problem*. According to Rendell [221], one of the purposes of constructive induction is to transform the instance space to diminish disjuncts.[11] An overview of approaches to constructive induction is provided by Rendell [222].

7.2.2 Similarity

Most work on concept formation in ML is concerned with bottom-up concept formation, often referred to as Similarity-Based Learning (SBL). However, exceptions such as Explanation-Based Learning (EBL) [64, 189] exist. In EBL the categories are formed beforehand and a high-level description of them is given as input to the learner. The task is to transform the abstract high-level characterization into a low-level characterization (often in terms of perceptual features). Thus, no categories are actually *formed*.

In similarity-based learning, both geometric and set-theoretic models of similarity are frequently used. Geometric models are used by for instance, instance-based algorithms, such as the IB algorithms [5]. Set-theoretic models, on the other hand, are used by some memory-based algorithms [258] among others. Also combinations of these models have been used, for example by Michalski and Larson [181], who define the *syntactic distance* between two object descriptions as the sum of the syntactic distances between the values of each property in the descriptions. The syntactic distance between two property values is a number from 0 to 1: for qualitative properties, the syntactic distance is either 0 if the property values are identical or 1 if they are not identical, and for quantitative properties, the syntactic distance is the ratio of the absolute difference between the values to the total span of the domain of the property.

Models of structural similarity, on the other hand, are seldom used, most algorithms for learning structural concepts use identity as the only measure of similarity. Models based on transformations are also rare. Nagel's [198] approach to learning from examples is one of the few.

[11]The instance space consists of all possible examples and counterexamples of concepts to be learned. (The description space, on the other hand, is the set of all descriptions of instances or classes of instances that are possible using the description language of the learner [177].) Disjuncts are separate clusters of category members.

There are many algorithms that use an implicit rather than explicit similarity measure. Take for example, Fisher's COBWEB system [87] that uses a modified version of *category utility* (cf. Gluck and Corter [105]), which is an evaluation function based on information theory that favors high intra-category similarity and high inter-category differences, to evaluate formed categories. Since they are based on how properties correlate rather than on explicit similarity, measures of this kind are sometimes referred to as *correlation-based* (cf. Wrobel [292]). The LABYRINTH system by Thompson and Langley [268], a variant of COBWEB for structural descriptions, uses a similar evaluation function. However, in order to determine the best match (i.e., assess similarity) between an object and a concept, it makes use of analogical reasoning much in the same way as Goldstone suggested in the last section. A promising method for this task called the Structure-Mapping Engine (SME) is presented by Falkenheimer et al. [81].

Michalski and Stepp [182] propose an approach for measuring similarity in geometric models that, besides the two object descriptions, takes into account other objects and the set of concepts (in this case, features and background knowledge) that is available for describing categories. This measure called *conceptual cohesiveness* can also be classified as correlation-based. The background knowledge may include: definitions of property range and scale, specificity hierarchies over property values, implicative rules of constraints of one property on others, rules for the construction of new properties, suggestions or derivational rules for ranking properties by potential relevancy [259].

In machine learning systems the problem of selecting relevant features is almost always solved by letting a human operator select them. This choice of features introduces some *bias*[12] to the learning system. In some systems, however, the learning system itself has to select among the user-selected features. Several, more or less statistical, approaches for the selection of relevant attributes have been proposed, for instance, multidimensional scaling [147] and neural networks [96].[13]

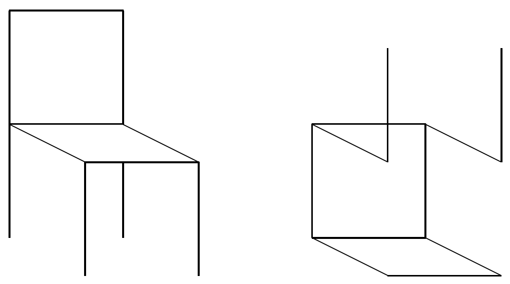
In the spirit of Schank and his colleagues [237], some experiments have been conducted that use explanations to select relevant attributes when doing top-down concept learning (EBL). However, the success has been limited, probably due to the difficulties in specifying the appropriate background knowledge.

7.2.3 Taxonomies

Taxonomies and their properties are rather well studied both in AI and in computer science in general. Take, for instance, object-oriented languages, such as Smalltalk and

[12] Actually, this is one of several types of bias. Other types are, for instance, the space of hypotheses that the system can consider, the order in that hypotheses are to be considered, and the criteria for deciding when a hypothesis is good enough (cf. Utgoff [275]).

[13] Multidimensional scaling and neural networks are used more to reduce the number of attributes, than to actually find the relevant attributes. However, these tasks seem closely related.



no. of rectangular pieces = 2
no. of cylindrical pieces = 4
made of = wood
weight = 8.6 kg
height = 95 cm
colour = black

Figure 7.6: Line drawings of two different object together with a description that represents both objects using only non-structural properties.

Simula, where the classes are members of taxonomies and where features are inherited from super-classes. The topic of taxonomies in AI, and in computer science in general, is further elaborated by Jansson [132]. However, of the existing concept learning systems it is only the conceptual clustering systems that actually construct taxonomies. Some of these systems also try to include basic-level aspects, e.g., COBWEB [88] and WITT [116].

7.3 Conclusions

In contrast to traditional AI where artificial categories often are used, an autonomous agent in a real-world environment has to deal with the same kind of categories (i.e., natural and derived) as humans do.

7.3.1 Properties

It seems that some properties are represented more naturally as nominal and some as ordered or numeric. Yet other are more structural in nature. Consequently, it would be desirable to have agents that could handle all these types of features. As we here are dealing with concepts representing objects (which by definition have structure), structural properties are of particular importance. Consider, for instance, the two objects depicted in Figure 7.6. Although the two objects are very different from each other, they are composed of exactly the same components. A description based only on non-structural properties will not be able to tell them apart. Instead, we need a description that also includes structural information (as illustrated in Figure 6.2).

The assumption that features are atomic entities readily available for the cognitive module is not compatible with our discussion in the chapter concerning world modeling on different levels of observation. In particular, the notion of perceptual features seems anomalous. This problem has also been acknowledged by Wrobel [291] who writes:

> ...we believe that any concept formation process relies on the filtered perception/interpretation of the world that the observer imposes... this means that any concept formation model relying on features must include an account of their creation. Otherwise, we would have only replaced the concept formation problem by the equally hard feature formation problem. (p. 713)

It seems that a natural candidate for solving this problem would be constructive induction. However, as Wrobel points out, all existing approaches construct their new features in terms of features already known. Thus, the new features are not more powerful than the original ones in the sense that they cannot distinguish objects that could not be distinguished with the original set of features, they are just abbreviations that allow more concise concept descriptions. Instead, he suggests that the more primitive features might be innate structures that has developed through evolution. In an autonomous agent context this would correspond to hard-wired, or pre-programmed, structures. This idea seems closely related to the concept of categorical perception.

Let us, however, suppose that some of the properties of an object are known to the learning system. How should these properties be used? First, we can note that there seems to exist properties of different types. Some properties, common to all objects of the category,[14] are characteristic or discriminant, these can be used for metaphysical and epistemological classification, e.g., for the category "human" we have, for instance, the genetic code and "walking upright" respectively. Other properties are common to all objects in the category although not characteristic or discriminant. These can be used to make inferences, e.g., having a heart. The remaining properties, sometimes called irrelevant properties, that are not common to all objects of the category, e.g., hair color, are then left over. These can be used for the representation of individual entities, but this is beyond the scope of this thesis.

7.3.2 Similarity

As we have seen, the formation of bottom-up categories has been rather well studied in AI. However, some problems remain to be solved, such as finding an appropriate similarity measure (and whether such a measure actually is necessary). In particular,

[14] One should not take "all" too literately. It may be the case that universal regularities do not exist, implying that reasoning about categories must be probabilistic in nature.

similarity measures that integrates nominal, ordered, numeric, and structural properties, need to be developed.

Top-down category *formation*, on the other hand, is hardly studied at all. Unfortunately, we do not get much help from the psychologists either. They have pointed out that there are categories that are formed in a top-down manner, but they do not give us a hint as to how the formation takes place.

There is a problem with artifact categories in that they seem to be both bottom-up and top-down categories, where the top-down-ness, and the problem, is due to the emphasis on the function of the artifact objects. The recognition of the possible functions of an object from perceptual observations seems like a very hard problem. According to Smith and Medin [251], "...one must have some knowledge that is capable of mediating between the features at the two levels; that is, to determine whether an abstract feature is perceptually *instantiated* in an object, one must have recourse to ancillary knowledge about the relation between abstract and perceptual features."[15] (p. 19) Moreover, the functions must, of course, be known in advance, pre-programmed or learned (which seems to be an even harder problem).[16] However, some ideas of how to transform structural knowledge about objects into functional knowledge are presented by Vaina and Jaulent [276]. On the other hand, it would require a very large amount of background knowledge to be able to form artifact categories in a top-down manner, and we still do not know how this should be done. The simplest solution may be to form artifact categories in a bottom-up fashion, making the assumption that perceptual similarity is enough. Thus, having bottom-up categories as the only permanent categories, and then constructing temporal top-down (derived) categories when convenient in problem solving tasks.

In sum, one might somewhat carelessly say that bottom-up categories arise due to curiosity, whereas top-down categories arise due to problem solving activities. Information about bottom-up categories is to a great extent derived from perceptual observations of the environment, whereas information about top-down categories comes from more abstract observations. Thus, a passive agent, just trying to make a description of its environment, could manage with only bottom-up categories, whereas a problem solving agent will probably also need top-down categories.

7.3.3 Taxonomies

Finally, we need to structure the categories into taxonomies to promote cognitive economy and inferential functions. Since this is a topic that has been extensively studied

[15] Some aspects of this problem is studied within explanation-based learning.

[16] It goes without saying that by letting the agent have access to observations on the linguistic level, where the function is given explicitly, this problem with functional properties disappears. However, assuming that the agent has access to such observations is too generous for most applications.

within computer science, it is probably better to concentrate on other less studied problems. However, the phenomena of basic level categories needs further studies.

Chapter 8

Representation of Categories

We are now ready to study how concepts, i.e., an agent's internal representations of categories, should be represented.[1] In this section we will discuss questions such as: How do humans represent categories? How do present AI systems represent categories? How should autonomous agents represent categories?

A number of representation schemes will be considered, mainly in terms of what kinds of properties they can handle and what model of similarity (if any) they use, but also which functions they are able to serve. Thus, although this chapter mainly is concerned with representational issues, we will also describe how a particular representation is intended to be used. In most cases, this corresponds to explaining how to use it for classifying object descriptions, or, in other words, serving epistemological functions.

8.1 Human Representation of Categories

Medin and Smith [175] present three views of human concepts: the *classical*, the *probabilistic* and the *exemplar*. These views are to a great extent theories about representation. In addition, some more recent views will be described, such as, combinations of the above views, the *explanation-based* view, and the *connectionist* view. We will begin with a discussion of the classical view and the problems it has to explain some empirical findings concerning human behavior.

[1]In the context of concepts, the word "representation" often leads to substantial confusion through its many different usages. As mentioned several times before, we have that concepts are representations of categories. In addition, a concept must be represented in the brain/computer by some representation scheme (e.g., a classical definition or a prototype), which, in turn, is represented in some representation language (e.g., a logic-based notation, a list of property-value pairs, or a neural network). (Note, however, that the same representation language can be used for representing different kinds of representation schemes.)

8.1.1 The Classical View and Its Problems

According to the classical view, all instances of a category share common features that are singly necessary and jointly sufficient for defining the category. Moreover, it says that it would suffice to represent a category by these features, thus generalizing the details of the instances of the category into a single, summary description covering all members of the category. Categorization would then be a matter of straightforward application of this "definition". For instance, a classical representation of the category "chair" could be: can be used by a person to sit on, has at least three legs, and has a back. Thus, by this definition, an object is a chair *if and only if* it can be used by a person to sit on, has at least three legs, and has a back.

However, there are some problems with this view (cf. Smith et al. [251, 249]):

- For some *natural* categories it seems not possible to find necessary and sufficient features.
- Even if a category can be defined as above we tend not to use this definition.
- There are unclear cases of category membership.
- Some instances of a category are regarded as more typical than others.
- We often think more concretely than the situation demands.

The fact that some categories do not have a classical definition is sometimes called the *ontological problem* [9]. A nice and famous example, mentioned by Wittgenstein, is the category "game". He argued that instances of "game" share many common features, but that no subset of these features can be found to be both necessary and sufficient.

Assuming that a classical definition exists for a category, it is interesting to notice that instead of using this definition we often (and are sometimes forced to) use non-necessary features to characterize a category or to categorize objects of the category. For instance, in recognizing a piece of gold, we generally cannot perceive the atomic structure of the material directly.[2] Instead, we use such features as color and weight (cf. the distinction between the epistemological and the metaphysical functions).

Not only is it in some cases unclear which particular category an object belongs to; the same person may even categorize an object differently as the context changes [173]. For example, it is sometimes hard to decide for some objects whether they are bowls or cups. In this example there is a relation between an object and a category but the same problem can arise between two levels in a taxonomy (i.e., subcategory-category relations). For instance, is a tomato a fruit or a vegetable, or is a rug a piece of furniture?

[2] We are here assuming that it is possible to provide a classical definition of gold in terms of its atomic structure.

The observation that people regard some instances of a category as more typical than others inspired the invention of the notion of *prototypes*. However, this term has been used ambiguously, often in one of the following meanings:

1. the best representative(s) or most typical instance(s) of a category
2. a description of the best representative(s) or most typical instance(s) of a category
3. a description of a category that is more appropriate for some members than it is for others.

The first of these refers to actual objects, whereas the other two refer to representations. Moreover, the second refers to a representation of a singular object, whereas the third refers to a representation of a class of objects. It is possible to make a further distinction regarding the second meaning: the described instance may be either (a) an actual instance, or (b) a constructed ideal, or average, instance that does not have to correspond to an object in the world.

The introduction of prototypes is in opposition to the classical view that regards categories as equivalence classes. It has been shown that (at least for the experiment subjects) robins and bluebirds, in contrast to penguins and bats, are prototypical birds [249]. However, the existence of prototypes does not have any clear implications for the construction of models of human category representation, processing and learning. Thus, prototypes do not specify such models, only impose constraints on them.

The last problem on the list concerns the fact that it seems that we often think about specific objects when we actually refer to a category. For example, if someone says that he had to see a dentist, it is hard not to think of a specific dentist.

From these five objections, it seems clear that the classical view cannot explain all aspects of human concepts. It has been suggested that instead of the strong demand that category shall have a classical definition, the instances of a category need only to have a sufficient amount of *family resemblance* [286, 226]. A common measure of family resemblance is the number of features that are shared by members of a category. Thus, it can be viewed as a measure of similarity, but also of typicality since typical members of a category share many attributes with other members of the category (and few with members of other categories). Conforming to these considerations, the probabilistic and the exemplar view have been presented as theories being more realistic and consistent with empirical findings.

8.1.2 The Probabilistic View

According to the probabilistic view, also called the family resemblance view, a category is represented by a summary representation in terms of properties that may be only probable, or characteristic, of its members. Membership in a category is graded rather than all-or-none; better members have more characteristic properties than the weaker ones. Thus, this kind of representation corresponds to a prototype in the third meaning as described above.

Several models of probabilistic category representation have been proposed [251]. They differ mainly in the assumptions made regarding the nature of the properties used to describe the categories. The *featural approach* assumes that all the properties used to characterize categories and objects are qualitative (often only binary). It is assumed that the probability for them to occur in instances of the category is high and that they are salient in some respect (perceptually or conceptually). A category is then represented by a list of weighted features, where the weight corresponds to the probability and/or salience of the feature occurring in an instance of the category. A probabilistic representation of the category "chair" could be, for example: (1.0) can be used of a person to sit on, (0.9) has at least three legs, (0.9) has a back, (0.7) is made of wood. An object will then, for example, be categorized as an instance of a category if it possesses some critical sum of the weighted properties included in the representation of that category. Thus, when classifying objects this approach makes use of a measure strongly related to the set-theoretical models of similarity. In this case, however, representations of instances are compared to representations of categories rather than to representations of other instances.

The *dimensional approach* assumes, in contrast to the featural approach, that all the properties are quantitative (i.e., dimensions). These dimensions, which ought to be relevant and/or salient, provide a multidimensional space. A category is then represented by a point in this space, a prototype, which is typically the "average" instance. For instance, "sit-ability" = 1.0, number of legs = 4, height = 96 cm, and so on. An object will then be categorized as an instance of a category if it is within some threshold distance of the prototype. Thus, rather than applying a definition, categorization is also in this case a matter of assessing similarity, typically by applying a geometric measure of similarity. Instances are compared to categories, but in contrast to the featural approach the representations of categories are of the same type as representations of objects. The dimensional approach as described here has, in fact, many similarities with the exemplar view (and can be seen as a hybrid between the probabilistic and the exemplar views). For instance, the point that represents the category is a prototype in the second meaning rather than the third. However, while the prototypes of the exemplar view are descriptions of actual instances (2a), the prototypes of the dimensional approach are descriptions of constructed entities (2b).

A problem with the current probabilistic approaches is that they assume that all properties are of a singular kind, either qualitative or quantitative. In fact, there is also a third approach, the *holistic approach*, which handles only structural properties. The most developed holistic approach is based on *templates*. A template is a representation which is isomorphic with the object it represents. This implies that templates are only meaningful for concrete objects (which are those we are interested in for the moment). Just as in the dimensional approach, a template representing categories corresponds to some kind of average category instance. To categorize an object is to determine whether or not it provides an overall, or holistic, match to the template representing the category, a process often referred to as *template matching*. What kind of similarity measure to use for this process is dependent of the representation language of the template. One suggestion is to represent a template as a matrix of cells where each cell is either filled or not (cf. Palmer [207]). In this case similarity between two instances may be determined by the number of common (or differing) cells. Unfortunately, there are many problems with such primitive approaches. For instance, they are extremely sensitive of the size and orientation of objects (see Section 8.2.3). It is often assumed that some kind of pre-processing mechanism standardizes the size and orientation of objects to be categorized. Moreover, as a category is represented by a singular template this approach has problems also with disjunctive categories.[3] Finally, we should note that in the cognitive science literature one can only find very primitive theories describing the holistic approach.

8.1.3 The Exemplar View

Those in favor of the exemplar view argue that categories should be represented by (some of) their individual exemplars, and that concepts should correspond to representations of these exemplars.[4] Such a representation corresponds to a prototype in the second meaning as described above. A new instance is categorized as a member of a category if it is sufficiently similar to one or more of the known exemplars of the category. Thus, also in this case categorization is a matter of assessing similarity rather than applying a definition. Just like the probabilistic view, there are both featural and dimensional approaches of the exemplar view. Featural approaches assumes qualitative properties and uses a set theoretical model of similarity whereas dimensional approaches assumes quantitative properties and uses a geometric model of similarity. The only difference between the dimensional approaches is that according to the exemplar view the prototypes correspond to actual exemplars whereas they according to the probabilistic view correspond to averaged entities constructed from the actual exemplars.

There are several models consistent with the exemplar view. One such model is the *proximity* model that simply stores all instances. An instance is categorized as a mem-

[3]However, it seems possible to extend the holistic approach by including multiple templates to handle disjunctive categories.

[4]In contrast to the classical view that seems to try to capture the intension of concepts, the exemplar view (at least partially) describes their extension.

ber of the category that contains its most similar stored exemplar. Another model is the *best examples* model that stores only selected, typical instances. This model assumes that a prototype exists for each category and that it is represented as a subset of the exemplars of the category. Yet another approach is the *context model* suggested by Medin and Schaffer [174] that in addition to individual instances also allows exemplars to be summary descriptions and thereby reduces the number of exemplars to be memorized. The context model also differs from other exemplar models in that the categorization is to some extent context dependent, i.e., attributes may have different weights in different contexts.

8.1.4 Combining the Probabilistic and Exemplar View

Another possibility is that the representation of a category contains both a probabilistic summary representation and exemplars [251]. It seems reasonable that when the first instances of a category are encountered we represent it in terms of these instances. And when further instances are encountered we apply abstraction processes to them to yield a summary representation.

This approach has some interesting features that relates to *non-monotonic reasoning* [102] and *belief revision* [94].[5] Consider a point in time where a person has both a summary and an exemplar representation of the category "bird", where the summary representation contains the feature "flies" (as very probable). How should the representation be updated when the person is confronted with a penguin? It would not be wise to alter the old summary representation too much because the fact that a random bird flies is very probable. A better solution is probably to store the penguin as an exemplar as can be done in a combined representation. However, there are many details to work out before we have a complete theory about such a combined representation.[6]

8.1.5 The Explanation-based View

The explanation-based view as described by Komatsu [146] is related to the ideas of Schank and others presented in the last chapter which argued that perceptual similar-

[5] In fact, all *incremental* concept learning methods have problems related to non-monotonic reasoning and belief revision.

[6] The possible connection between prototype-based representations and non-monotonic reasoning has been pointed out by Gärdenfors [95]. It is suggested that concepts at the conceptual level are represented as convex regions in a conceptual space. When an individual is first known as being a bird, it is believed to be a prototypical bird, located in the center of the region representing birds. In this part of the region birds do fly. If it then is learned that the individual is a penguin, the earlier location must be revised so that the individual will be located in the outskirts of the "bird-region", where most birds do not fly. However, my reflection concerns the acquisition of the representation, whereas in Gärdenfors' case the representation is already learned. Moreover, the combined approach is on the linguistic level and not restricted to convex regions.

ity alone is not sufficient to form categories. Rather, category formation is also influenced by the theories we have about the world. According to the explanation-based view the representation of a category should include information about how it is related to other concepts and about the functional and causal relationships that hold among the attributes associated with its instances. However, not many (if any) explicit suggestions have been presented of how such concepts should be represented.

8.1.6 The Connectionist View

Recall Gärdenfors' three levels of observation from Section 4.3.3. In the same way that observations can be described on different levels, it is possible to represent concepts on different levels. The methods of representation described above are all on the linguistic, or symbolic, level. One method of representing (and acquiring) concepts on a lower level is by using *neural networks*. A neural network basically consists of a number of nodes connected by weighted links. Neural networks were initially meant to be cognitive models of the brain at the level of neurons.

Pylyshyn [214] has distinguished three levels of cognitive modeling. The lowest level is concerned with the physiological mechanisms underlying thought. The highest level is concerned with the content of thought, the aspects of the world that are encoded in the mind. Between these levels are the mechanics of how a representation is formed without regard to the content of the representation. Newell [201] refers to this level as the symbol manipulation level. Thus, whereas the representations discussed earlier have all belonged to the middle level, neural networks belong to the lowest level.

During the last years there has been a growing optimism about the capacities of neural networks, both as cognitive models (e.g., the works of Grossberg and Carpenter [47] and of Edelman [76]) and as tools for pattern recognition (e.g., networks using the backpropagation algorithm [232]). However, one must keep in mind that neural networks that can be simulated on a computer (i.e., most current neural networks), are of course at the most Turing-machine-equivalent. They might be better suited (e.g., more efficient or easier to program) than symbolic systems for some problems, but are not a more powerful tool in general.

In addition, there are some problems with neural networks. For example, neural networks do not represent knowledge explicitly, something which seems crucial for the implementation of the metaphysical and inferential functions. The functions that the subsymbolic methods will be able to handle seem, at least for the moment, limited to tasks related to the epistemological function, such as perceptual categorization.[7] However, most neural network approaches assume that instances are represented by a set of binary, or numeric, features, and are thus not suited for structural descriptions.[8]

[7]Even though the opposite opinion is sometimes held (cf. Balkenius and Gärdenfors [19]).

[8]However, some attempts at incorporating structural knowledge have been made (cf. [18, 254]).

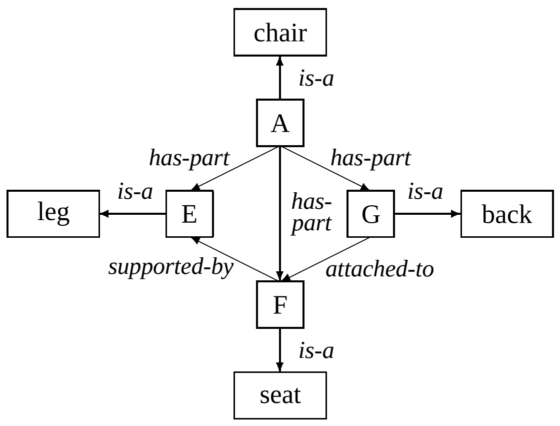


Figure 8.1: Part of a semantic network representing the category "chair". Only one leg (E) is included, leg B, C and D are supposed to be connected in the same way.

Another problem is the difficulty of introducing background (a priori) knowledge into neural networks.[9] Moreover, it is difficult to exploit and reason about both the learned knowledge and the learning process. A more detailed discussion about the properties of connectionist models in general is provided by Smolensky [253].

8.2 Representation of Categories in AI

Traditionally in AI, categories are treated as equivalence classes that can be described by necessary and sufficient conditions (or a disjunction of such conditions). Thus, AI has adopted a version of the classical view. In this section we will, after a brief review of general AI approaches for representing categories, concentrate on the different types of concepts used in the sub-fields of machine learning and computer vision.

8.2.1 General AI Category Representations

As pointed out earlier, the concept of concepts has not been a main subject for research in AI. *Semantic networks* suggested by Quillian [215], however, is one approach for

[9]There have been experiments introducing symbolic knowledge into "knowledge-based" neural networks, (cf. Towell et al. [269]. However, in my opinion these networks are rather symbolic than subsymbolic representations since every node explicitly represents something. Also, the knowledge in such networks is not distributed, which is one of the characteristic features of neural networks.

```
frame CHAIR
    number of legs: NUMBER
    made of: MATERIAL
    back: BOOLEAN default = true
```

Figure 8.2: A frame representing the category “chair”.

representing categories that has gained some attention. One of the main ideas behind semantic networks is that the meaning of a concept is determined by the ways it is connected to other concepts. The information is represented as a set of nodes, which represent categories (or objects), connected to each other by labeled arcs, which express relationships among the nodes. Figure 8.1 shows an example of a partial semantic network representing the category “chair”. As we can see, semantic networks are suited for representing also structural properties.

Another type of general structures that has been successfully used for representing different kinds of knowledge are *frames*, first discussed by Minsky [184]. Essentially, a frame is a collection (of representations) of facts that, for instance, can be used to represent a category or an instance of a category (see Figure 8.2). The contents of the frame is a list of slots that define relationships to other frames that have various functions in the definition. For instance, the definition of the slot “made of” states that the frame MATERIAL has the function of being the stuff of which a chair is made. Moreover, a slot can contain a default value that is used in the absence of other information. Thus, unless told otherwise a system using the frame in the example will infer that a chair has a back. An approach, similar to frames, for representing event categories is to use *scripts* (cf. Schank and Abelson [236]).

These representation schemes, frames in particular, are assumed to be used mainly for inferential functions. This, in contrast to the other representations that has been, and will be, presented, which mainly are used for categorization.

8.2.2 Machine Learning Category Representations

In ML, it is mainly three kinds of representation languages that have been used to represent classical concept definitions: *logic-based notations*, *decision trees*, and semantic networks. In one of the early concept learning programs, Winston [282] employed semantic network representations of structural descriptions (both of instances and categories). The network illustrated in Figure 8.1 is an example of what such a description might look like.

$$\exists\, x\ \exists\, z\ \exists\,(y \geq 3)\ [\ \text{person\_can\_sit\_on}(x) \wedge \text{type}(y) = \text{leg} \wedge \\ \text{ontop}(x,y) \wedge \text{attached\_from\_above}(z,x)\ ]$$

Figure 8.3: Logic-based category representation (conjunctive).

Logic-based notations have been used in, for instance, the AQ-algorithms developed by Michalski [176]. We have, in fact, already seen an example of a representation in a logic-based notation, namely the BCR part of a two-tiered representation of the category "chair" from Chapter 6 (repeated in Figure 8.3). We see here that logic-based notations are able to represent also structural properties (e.g., ontop(x,y)).

A disadvantage with the pure classical view that has not been mentioned, is that classical definitions, being limited to conjunctive descriptions, are not able to represent disjunctive category descriptions. For example, if we want our description of chairs also to cover wheel-chairs, we need to augment it with a disjunction as illustrated in Figure 8.4.[10] In what follows, however, we will regard disjunctive descriptions of this kind as belonging to the classical view as well.

$$\exists\, x\ \exists\, z\ \exists\,(y \geq 3)\ [\ \text{person\_can\_sit\_on}(x) \wedge \text{type}(y) = \text{leg} \wedge \\ \text{ontop}(x,y) \wedge \text{attached\_from\_above}(z,x)\ ]\ \bigvee \\ [\ \text{person\_can\_sit\_on}(x) \wedge \text{type}(y) = \text{wheel} \wedge \\ \text{ontop}(x,y) \wedge \text{attached\_from\_above}(z,x)\ ]$$

Figure 8.4: Disjunctive logic-based representation.

The most popular way of representing classical concept definitions is probably by using decision trees. The first attempt to learn decision trees resulted in the CLS program by Hunt and his colleagues [128]. It is, however, the ID3 algorithm by Quinlan [217] that can be regarded as the source for the present popularity of induction of decision trees. Figure 8.5 shows an example of a decision tree describing the category "chair". At each node in the tree there is a test for sorting instances down the alternative

[10]This is, of course, dependent on the description language. If we have term that covers both legs and wheels, a conjunctive description would be sufficient.

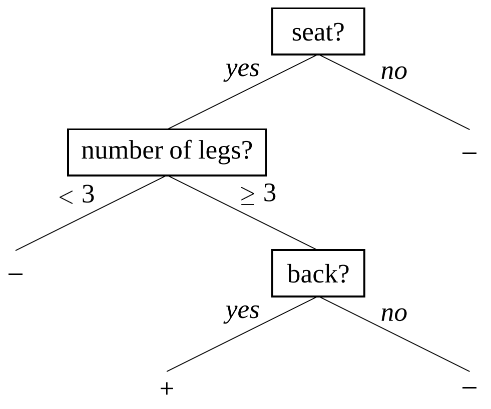


Figure 8.5: Decision tree representation of the category “chair”.

branches. The terminal nodes are marked with either a “+”, to indicate category membership, or a “–”, to indicate non-membership. In this example, the tree corresponds to a conjunctive description since only one terminal node is marked with a “+”. It is, however, possible to achieve a disjunctive representation by having several such nodes. A disadvantage with decision trees is their not being suited for representing structural properties.

Non-Classical Representations

Within the last few years, several ML experiments with non-classical representations have been carried out. We will here concentrate on those inspired by the exemplar and the probabilistic view. Regarding the connectionist view it is difficult to find a sharp boundary between the cognitive modeling and the engineering approach. Concepts are represented as a collection of nodes connected by weighted links. The ML experiments inspired by the explanation-based view, on the other hand, typically relies on classical definitions in some logic-based notation.

Beginning with those who are influenced by the exemplar view, Kibler and Aha [139] have experimented both with the proximity model where all instances are stored, and with selected examples models where only a subset of the instances is stored. Systems using this kind of representation often employ some version of the *nearest neighbor* algorithm to classify unknown instances. That is, a novel instance is classified according to its most similar known instance. In contrast to most of the approaches inspired by the exemplar view, which typically use geometric or set-theoretical models of similarity, Nagel [198] suggests a best examples model that employs a transformation-based similarity model. In addition to the prototype(s), transformations are stored that transform less typical instances to a prototype. Learning algorithms that use specific instances rather than abstractions to represent categories have been labeled *instance-based* by Aha and his colleagues [5].

A follower of the probabilistic view is, for instance, de la Maza [61] who calls his type of representation *augmented prototypes*. These are created from prototypes which consist of one vector for every attribute that is used to describe the examples. If the attribute is nominal, the vector contains the frequency of each value that the attribute can take on. If the attribute is numeric the vector contains the mean and standard deviation of the attribute. Thus, this is a hybrid approach that combines the featural and the dimensional approach, but cannot handle structural attributes. The augmented prototypes are created by adding a weight to each attribute that are computed by comparing the prototypes.

Another probabilistic approach is described by Musgrove and Phelps [196]. They have adopted the dimensional approach where the prototype reflects the average member of the category. Moreover, they use multidimensional scaling to reduce the number of dimensions. A featural approach is taken by Fisher [88] in COBWEB where he uses a *probabilistic concept tree* to represent a taxonomy of probabilistic concepts.

8.2.3 Computer Vision Category Representations

The typical goal of *model-based* object recognition systems for robot vision (i.e., the subfield of computer vision that will be regarded here) is to “recognize the identity, position, and orientation of randomly oriented industrial parts” [54]. A typical scenario is the “bin-picking” problem in which the parts to be identified are disorderly placed in a bin. However, there is a problem with comparing the representations that such systems use when recognizing objects with the category representations we have discussed earlier. Since the objects to be recognized typically are industrial parts, they have almost the same shape, whereby the representations (or models, as they often are called) used in this task often are not representing categories, but rather can be seen as representations of specific objects. That is, they are categories, but on a much lower level of abstraction than, for instance, the basic level (e.g., the category might be wrenches of a particular type and brand, rather than wrenches in general). Thus, almost no generalization takes place and representations are mainly of a conjunctive nature.

According to Chin and Dyer [54], there have been three types of representations used in model-based vision systems: 2-D, $2\frac{1}{2}$-D, and 3-D object (or category) models. A 2-D model consists typically of shape features derived from the silhouette (i.e., the set of boundaries) formed by the gray-scale (or binary) image of an object. 2-D models can be used when the objects to be recognized have a simple structure and are presented against a high-contrast background. Moreover, systems based on such models are often only able to recognize the objects from a few fixed viewpoints and typically demand that they are not occluded by other objects. Thus, this class of representation is only appropriate for tasks where the environment can be completely controlled. The advantage with 2-D models is that they are relatively easy to construct automatically.

In conformity with 2-D models, $2\frac{1}{2}$-D models are viewer-centered representations. In addition, however, they try to capture surface properties of the object such as range (depth) and surface orientation. Thus, $2\frac{1}{2}$-D models are descriptions of surfaces rather than boundaries. Disadvantages with such models are that they, just as 2-D models, are viewpoint-specific and that the additional step of deriving the surface description makes them harder to construct automatically.

In contrast to 2-D and $2\frac{1}{2}$-D models, 3-D models are viewpoint-independent representation. They are often volumetric representations that allow a complete object description from an unconstrained viewpoint using either sweep models, surface models, or volume primitives. Representations that have been used are for instance: generalized cylinders ("sweep representations") [38], surface patches [243] (see [34] for a recent review of surface-based representations), superquadrics (superellipsoids) [22], geons (a subset of the generalized cylinders) [29, 68].

In a (3-D) model-based vision system, the recognition of objects consists in matching the input image with a set of models that are typically preprogrammed using a CAD system. Some critical assumptions often made, are (1) that the objects (categories) to be recognized are exactly specified, with known tolerances on dimensions and features, (2) that the number of objects (categories) is usually small (less than 50). Thus, most existing approaches are expected to function only in very controlled environments.

Another kind of 3-D model is multi-view feature representations in which a set of 2-D or $2\frac{1}{2}$-D descriptions (one for each relevant view), also called *characteristic views*, are combined into a single composite model. Thus, rather than trying to capture the *shape* of objects, they try to describe the *appearance* of objects. An interesting approach called *appearance models* is described by Murase and Nayar [193]. The authors argue that since "the appearance of an object is the combined effect of its shape, reflectance properties, pose in the scene, and the illumination conditions", recognizing an object from a brightness image (such as the output from a video-camera) is more a problem of appearance matching rather than shape matching. The appearance model is constructed from a large set of images of the object with varying pose and illumination. This set is then compressed into an eigenspace (a low-dimensional subspace) in which the object is represented as a hypersurface. To recognize an unknown object, you only need to check which hypersurface the image of the object is projected onto (the exact position can, in addition, be used to determine pose and illumination). However, as we shall see in the next chapter there are some assumptions made regarding the learning of these models that makes it hard to employ this approach directly.

All the model-based approaches presented in this section have (albeit to a varying degree) adopted the holistic approach of the probabilistic view. They use templates in one form or another to represent categories and objects are categorized by template matching. The inherent limitations of the holistic approach pointed out in Section 8.1.2 are, of course, valid also in this case. For instance, they are not designed for representing disjunctive categories, and only structural properties are represented in a natural way (other kinds of properties can only be represented implicitly, if at all).

8.3 Representation and Function

So, how should autonomous agents represent categories? Let us analyze this problem in terms of the functions that the concepts should be able to serve. From the above review of existing approaches, it should be clear that most of them concern representations to be used in some categorization task. Thus, they can be said to serve the epistemological function.[11]

Although there may be some categories that can be characterized by a classical definition, such a definition is often based on features that under normal circumstances are impossible, or at least difficult, to detect by perception, such as atomic structure, genetic code or functionality. Thus, these definitions are not adequate for perceptual classification,[12] and consequently not appropriate representations for supporting the epistemological function. Instead, the implementation of the epistemological function seems to demand some kind of prototype-based, possibly subsymbolic, representation. Moreover, since structural relations (between parts) are important when perceptually categorizing objects, a representation supporting the epistemological function should also be able to represent structural relationships. As we have seen, only a few of the representations used in machine learning have the ability to describe structural relationships, whereas in computer vision this is compulsory. On the other hand, object models used in computer vision often lack the ability to represent global features such as color, texture, and function that can be useful in object recognition.

Smith, Medin and Rips [252] suggest that there are two kinds of epistemological categorization. One that makes use of the properties in the core of the concept and one that uses the identification procedure. They write:

> Specifically, identification properties are often useful for a ‘quick and dirty’ categorization of objects, and such properties tend to be salient and easy

[11] Even if some researchers probably would argue that their systems perform the metaphysical function.

[12] However, in traditional AI it is very common to try to make a classical definition of a category based directly on the perceptual data.

> to compute though not perfectly diagnostic of category membership: core properties, on the other hand, are more diagnostic of category membership, but they tend to be relatively hidden and hence less accessible for rapid categorization. (p. 267)

They illustrate this by the problem of identifying the gender of a person (i.e., categorizing instances of the categories “males” and “females” respectively). Categorization by identification properties may then take into account style of clothing, hair, voice and so on, whereas categorization by core properties may involve what kind of sexual organ the instance has. This latter kind of categorization bears a strong resemblance to what we have called metaphysical categorization.[13] Thus, the distinction between categorization by core properties and metaphysical categorization is clearly unclear. However, since it is required that the rules that determine category membership are explicitly represented, the implementation of the metaphysical function demands by definition a classical definition.

To implement the inferential function, on the other hand, it seems that we must have some “encyclopedic” knowledge about the category and its members. This knowledge is naturally seen as a collection of universal or probabilistic rules. Kirsh [141] has called this collection “a package of associated glop”. Closest at hand is, of course, the representing of this kind of knowledge in some logic-based notation and/or by frames. However, results from the research of the CYC project led by Lenat [158] indicate that it is necessary to combine several of these representation languages to capture all the encyclopedic knowledge that might be relevant for an autonomous agent.[14] Another possibility would be to represent some of this knowledge by diagrammatic, or pictorial representations. However, research in this area has only just begun (cf. [50]).

The probably most important reflection on the review above is that almost all work on category representation in AI has assumed that a single and simple structure, such as a logic-based description, a decision tree, or an instance-based description, could capture all the relevant aspects of a concept. This opinion seems to be shared by the majority of the members of the cognitive science community.[15] However, the above discussion should make clear that this is not possible except in very restricted domains. This implies that a richer, composite representation is needed that is structured according to the desired functions of the concept. The insight that multiple category representations sometimes are required, has only on very few occasions been explicitly expressed in the AI-literature. As an example, Flann and Dietterich [91] write:

[13] Although one might suggest that metaphysical categorization would take into account the genes of the object.

[14] Using the terminology above, the long-term goal of the CYC project can be described as capturing all the encyclopedic knowledge of every category known to man.

[15] Surprisingly, even in the debate in Cognition [223, 252, 224] where the need for concepts to serve several functions was stressed, it was assumed that categories should be represented by a single structure.

> The performance task for which a concept definition is to be learned may require a structural representation (e.g., for efficient recognition), a functional representation (e.g., for planning), or a behavioral representation (e.g., for simulation or prediction). (p. 461)

In a similar vein, Matheus [168] draws the conclusion that "...there are purposes for which a single representation simply cannot satisfy all requirements." (p. 42) In more general terms, Sloman [246] points out that different purposes require different kinds of knowledge representations. In the next section we will present some AI approaches that make use of multiple, or composite representations.

8.4 Composite Category Representations

There are at least two senses in which a category representation can be composite:

- the components merely use different vocabularies
- the components are represented in fundamentally different ways.

Flann and Dietterich [91] present an approach of the first kind where the components use the same fundamental representation but with different vocabularies. A concept consists of two parts: the *learning representation*, which is used to facilitate effective (i.e., incremental) learning, and the *performance representation*, which is used to facilitate task performance. This approach has been applied to the learning of concepts in board games (e.g., "skewer" and "knight-fork" in chess). In this case the performance task was the recognition of board positions that are instances of such categories. Since these kinds of concepts are naturally functional (in the sense that the similarity between the instances is mainly of a functional nature whereas they may be very dissimilar according to structure, cf. derived categories), the learning representation used a functional vocabulary. The performance representation, on the other hand, was represented in a structural vocabulary that permits efficient matching against board positions.

An approach to multiple category representation where the components are represented in fundamentally different ways[16] is, as we have seen earlier, taken by Michalski and his colleagues [178, 27]. Their representation has two components, the *base concept representation* (BCR) and the *inferential concept interpretation* (ICI). The BCR is a classical representation that is supposed to capture typical and relevant aspects of the category, whereas the ICI is a set of inference rules that should handle exceptional or borderline cases. When categorizing an unknown object, the object is first matched against the BCR. Then, depending on the outcome, the ICI either extends or specializes the base concept representation to see if the object really belongs to the category.

[16]However, they use the same representation *language* (a variation of predicate logic).

Yet another approach is Rendell's PLS [219] which makes use of three different kinds of representation: an exemplar-based table of counts that is used for learning, a probabilistic region representation that is used for generalization, and an evaluation function that is used for prediction. However, common to these approaches is the fact that they do not support all the desired functions presented in Chapter 6. In fact, they support only a single performance task; either categorization or prediction. The supporting of all the desired functions, on the other hand, implies that the concept must be used for several performance tasks including, for instance, categorization, communication, and prediction. Thus, none of the presented multiple category representations is powerful enough to meet the demands required by an autonomous agent.

8.5 A Novel Framework for Composite Concepts

The discussion above makes clear that it is not possible for a single and simple structure to capture all the relevant aspects of a concept. We need a richer composite representation that, in some way, is structured according to the functions of the concept to be represented. Such a structure containing five components was proposed in paper III as a candidate for the representation of concepts by autonomous agents. Three of the components, the *internal designator*, the *epistemological component*, and the *inferential component*, are compulsory. One, the *external designator*, is necessary for communicating agents in a multi-agent scenario. The last one, the *metaphysical component*, is not always necessary and is perhaps not even adequate as it is not clear whether metaphysical definitions actually exists for all concepts and might, besides, be irrelevant for an autonomous agent. On the other hand, since perceptual classification often is context dependent, it might be convenient to have more than one epistemological representation. For instance, in daylight a gnat can be recognized by its look, whereas it must be recognized by its sound when it is dark. Similarly, it is useful to have several external designators in domains where many communication languages are used.

The idea of this composite representation is illustrated in Figure 8.6 by using the category "chair". For the external designator it is natural to choose "chair" (in an environment where communication is based on the English language, that is). However, in multi-agent systems it is becoming common to specify *ontologies* to which the agents are committed.[17] Then, we should, of course, choose the term used in the ontology for the external designator. The choice of the internal designator, on the other hand, is entirely up to the system, it should be as convenient and effective as possible for the

[17]According to Gruber [111], an ontology is an explicit specification of a conceptualization. It describes the concepts, or rather the terms, and the relations between these terms that exist in a domain. An agent commits to an ontology if its observable actions are consistent with the definitions in the ontology. Thus, a common ontology defines the vocabulary with which queries and assertions are exchanged among agents.

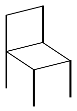
External designator:	chair
Internal designator:	thing.artifact.gfj65
Epistemological comp.:	
Metaphysical comp.:	can seat one person
Inferential component:	Number of legs is usually four, often made of wood have a back, can be used to sit on, ...

Figure 8.6: Composite representation of the category “chair”.

system. In the figure, the epistemological representation is a 3-D model of a prototypical chair (i.e., a template), but any representation that can be used by the perceptual system to successfully identify members of the category would be adequate. For the metaphysical representation we have chosen that something is a chair if it could seat one person. If the concept belongs to a common ontology in which it is formally defined, we should, of course, use that definition. Finally, the encyclopedic knowledge in the inferential representation includes the facts that a chair usually has four legs, and is often made of wood and so on.[18]

Of course there are no sharp distinctions between what types of information is included in these representations. They may even contain redundant information. For example, besides being a part of the epistemological representation, the fact that chairs have legs is a rather natural part of the encyclopedic knowledge represented in the inferential representation. However, the fact is not represented in the same way in these representations. For instance, it may be implicitly represented in a prototype-based representation for the epistemological representation and explicitly represented in a logic-based representation for the inferential representation.

This composite structure enables concepts to serve all the functions listed earlier. The epistemological, metaphysical and inferential representations support the epistemological, metaphysical and inferential functions respectively. The internal designator supports the intrapersonal stability, whereas the external designator supports both the interpersonal stability and the linguistic function.

[18]Note that in the metaphysical and inferential representations in the figure, it is only the *contents* of the representations that are described (in natural language).

Depending on the situation, the composite category representation is *accessed*, or retrieved in different ways. External “stimuli” in the form of direct perception of objects access the concept via the epistemological representation. If, on the other hand, the stimulus is on the linguistic level, as when communicating with other agents, the concept is accessed via the external designator. Finally, if the stimulus is internal, like when the agent is performing (symbolic) reasoning, the concept is accessed via the internal designator. Thus, the fundamental idea here bears strong resemblance with one that inspired Minsky [184] to introduce the concept of frames, namely that: ”When one encounters a new situation... one selects from memory a substantial structure...” (p. 212). In fact, one might see the composite structure suggested above as a “meta-frame” where the five parts correspond to slots to be filled in. However, the composite structure is not committed to a singular representation language, rather it is intended to promote the use of different representation languages. Moreover, it is more structured than a frame, and can in this sense be regarded as a specialization of the frame concept.

An issue that is not yet discussed is how these concept structures should be organized and how they relate to each other. In Chapter 7 we saw that categories can be hierarchically organized in taxonomies. As mentioned earlier, taxonomies serve an important function by promoting cognitive economy. Since categories inherit properties from their superordinate categories, it is possible to reduce the amount of information that have to be stored at each level in the hierarchy.[19] Thus, we need to complete the composite representation suggested above with taxonomical information, so that the concepts together form a tree-structure.

We also should note how the idea of the composite description stresses the possibility of having a concept in different degrees. Depending on how many parts of the description the agent has, it could be said to have the concept to a lesser or larger degree. For instance, it could only have the external designator (i.e., the agent only knows the word commonly used to refer to the category), or the epistemological representation (i.e., the agent is able recognize instances of the category, but does not know anything about the category explicitly). Another factor that determines the degree to which the agent has the concept is how well developed the representations are, for instance, the amount of encyclopedic knowledge contained in the inferential representation.

8.5.1 Toward a Solution to the Symbol Grounding Problem

The symbol grounding problem is described by Harnad in [118]. It concerns the meanings of the symbols in (physical) symbol systems. Traditional AI systems manipulate

[19] However, it seems that it is mainly encyclopedic knowledge that can be inherited in this manner. It is not clear how this could be done with classification knowledge (epistemological and metaphysical). If it is possible, it could significantly increase the effectiveness of the classification process (as is done in, for instance, Lebowitz’s UNIMEM [153]).

symbols that are systematically interpretable as meaning something. The problem of concern is that the interpretations are made by the mind of an external interpreter rather than being intrinsic to the symbol manipulating system. The system itself has no idea of what the symbols stand for, their meaning being totally dependent on the external operator.

A possible solution to this problem would be to attempt to *describe* the meaning of the symbols in a more powerful language. However, as described in Section 5.4 this would only lead to another set of symbols, ones which would likewise need to be interpreted, and in the end to an infinite regression.

Harnad's Solution

Harnad suggests in the same article ([118]) a solution to this problem. According to him, the meaning of the system's symbols should be grounded in its ability to identify and manipulate the objects that they are interpretable as standing for. He proposes a hybrid system with both symbolic and connectionist components, stating: "In a pure symbolic model the crucial connection between the symbols and their referents is missing ..." (p. 344)

He argues that three kinds of representations are necessary: *iconic representations*, which are the sensor projections of the perceived entities, *categorical representations*, which are "learned and innate feature detectors that pick out the invariant features of object and event categories from their sensory projections",[20] and *symbolic representations*, which consist of symbol strings describing category membership. Both the iconic and the categorical representations are assumed to be non-symbolic.

He concludes that a neural network is the most suitable for learning the invariant features underlying categorical representations and for connecting names to the icons of the entities they stand for. The function of the network then is to pick out the objects to which the symbols refer. Concerning Harnad's approach, one can remark that, although it seems clear that a pure symbolic system does not suffice (since sensors do not provide symbolic representations), regarding connectionist networks alone as being capable of serving this function appears too limited.

A More General Solution

As described in paper IV, the problem of symbol grounding becomes easier to resolve if one views it in terms of the general concept representation framework presented above. It is essentially the perception system, through its use of epistemological representations that are parts of the same structure as the corresponding symbols, which permits

[20] Harnad seems here to assume that categorization always is based on invariant features, a position assuming the classical view.

grounding, or the connection between symbols (designators) and their referents (objects in the world), to be carried out. An object that is encountered is matched with the epistemological representation. This activates a larger knowledge structure, the composite concept representation, of which the epistemological representation is a part. This structure contains, among other things (encyclopedic knowledge, for instance), the designator.

The main point to be made is that the epistemological representation does not have to be a neural network. Rather, it can be virtually any representation the vision system can successfully use in order to identify (categorize) objects.

Wrobel [291] has also pointed out the importance of grounding in concept formation, and the need of studying it within the context of acting agents. In contrast to the approach here, however, he chose to study the problem in a process-control setting, since he found vision and robotics too difficult and, surprisingly enough, unrelated to the problem. Wrobel admits, nevertheless, that since concept formation by physical agents takes place in a three-dimensional environment, it may require qualitatively different methods. His model allows only very simple numeric or nominal sensors.

Note that we are here discussing the meaning of symbols referring to (classes of) concrete objects, which are directly perceivable by the perceptual system. Symbols referring to more abstract entities, on the other hand, probably get their meaning more from internal structure than from external grounding (cf. Sloman [247]).

8.6 Conclusions

The main conclusion of this chapter is that it is necessary for an autonomous agent to represent categories by composite representations where the different components are chosen according to the function they should serve. This stems from a fresh view on concepts. Concepts should not only be used for some limited classification task. Instead, they should provide the basis for most of an agent's cognitive tasks. However, since representation of categories to be used for categorization is the thing most often discussed in the literature, it also has been the main topic here. In particular, representations for epistemological categorization have been investigated. Since the relevance of the metaphysical component is uncertain, we will not discuss it in any detail in the following chapters. Moreover, it should be noted that the ideas presented here are clearly not fully developed; rather, they suggest possible starting points for future research.

As we have seen, the AI community has already studied all of the well developed psychological models of category representation. The only approach that has not been implemented yet (at least to the author's knowledge) is the combined exemplar and probabilistic model. Even though it has not been studied in any depth in cognitive psychology either, it might be a candidate, at least from the engineering approach point of view, for the epistemological representation of categories. As the probabilistic representations seem to have trouble with atypical instances, it would be interesting to experiment with implementations of a combination of the probabilistic view and the exemplar view, which seems to handle such instances quite well. Moreover, since a combination does not have to store as many instances as an exemplar representation, it requires less memory and it probably categorizes faster, since fewer comparisons between instances are needed.

The main problem with most suggestions for epistemological category representations is that they only are able to handle one kind of properties, typically qualitative or quantitative. We have argued that it is necessary, at least for the epistemological component, to also be able to represent structural knowledge. However, as there are many different kinds of structural representations, the epistemological representation must be adapted to the agent's perceptual system as well, i.e., it must be in a form that the perceptual system can use.

An issue that has not been dealt with, is the problem that arises when the number of concepts grows. Having thousands (or more) of concepts will turn the speed issue into an acute problem for the performance of the epistemological function. An efficient way of indexing the concepts is necessary. This seems to require a hierarchical organization of the concepts of the kind we have mentioned earlier. Some interesting work has been carried out regarding this issue (cf. Sengupta and Boyer [240]).

Chapter 9

Concept Acquisition

Finally, we have reached the stage where we are able to discuss how concepts can and should be acquired. We will consider theories of human concept acquisition and approaches to concept acquisition taken within AI (i.e., machine learning, computer vision, and pattern recognition) as well as the theoretical studies of what can actually be learned. In this chapter, as in the earlier, the different approaches will in most cases be evaluated qualitatively rather than quantitatively. Moreover, it should be stressed already at this early stage that most, if not all, work on concept acquisition regards the learning of representations that corresponds to what we in the previous chapter referred to as epistemological components. That is, representations that should be used for categorizing objects.

9.1 Human Concept Acquisition

According to Atkinson et al. [16], humans learn about categories in two different ways:

- by being explicitly taught
- by learning through experience.

Unfortunately, they do not elaborate this distinction any further, and it has been hard to find any other discussions concerning this topic.[1] However, it seems reasonable to believe that it is possible to be explicitly taught about categories both on the linguistic level, i.e., *learning by description*, and on a sublinguistic (perceptual) level, i.e., *learning by acquaintance*. Examples of learning on the linguistic level are when you learn

[1] In fact, there is not much written (lately, at least) about human concept acquisition. In the last decades the researchers in the field seem to have focused on representation and classification processes. Therefore, the main part of this section is the author's own interpretations and elaborations of the few notes on the topic that have been found.

something reading a book or by being told something by some kind of teacher. It seems likely that we learn metaphysical, and to some extent inferential, knowledge of concepts in this way. As an example of being explicitly taught on the perceptual level we have the situation when a teacher shows an actual exemplar of a category (cf. ostensive definitions).[2] Thus, it is primarily epistemological, but also inferential, knowledge that is learned in this way.

When learning from experience, there is no teacher available to help you with the classification. This kind of learning is often called *unsupervised learning* and is contrasted to *supervised learning* that strongly relates to learning by being explicitly taught. For example, if you are confronted with an instance of a category you know rather well, but this instance is different in some respect from the ones you have previously been acquainted with, you might nevertheless “guess” what category it belongs to and, thus, learn something about the category. Another situation is when you are confronted with an instance of a category you know nothing about. You may then form a new category based on that instance. Thus, there are two cases of learning from experience, it can either be learning something about a known category or about an unknown category. Note that the input when learning through experience often is on the perceptual level.

There is yet another way of learning about categories that is, in a way, orthogonal to the others, namely, *learning by experimentation*. It could be performed by actually making experiments or, perhaps more commonly, by asking questions (i.e., “Is this an elephant?” or “What features characterize an elephant?”). This type of learning seems to bear some resemblance to scientific discovery. However, it is important to remember that in real life we do not acquire a concept in just one of these ways. On the contrary, it is the author’s opinion that we use them all alternatively. Which kind of learning that is the appropriate one in a particular situation is, of course, to a great extent determined by the situation, e.g., whether there is a teacher present or not.

There are several other restrictions that the environment imposes on the concept acquisition process. For instance, it must be *incremental*, since we do not encounter all instances of a category at one point in time. Instead, we encounter instances now and then, incorporating it into our “bulk of knowledge of concepts”. Thus, concepts are acquired in a gradual fashion, by interacting with the environment over time. In more technical terms, the learning system must be able to switch modes between learning and classification without destroying any previous learning. Moreover, we do not learn one concept at a time, concepts are rather acquired in *parallel*.[3] Yet another constraint is that the learning must be accomplished relatively *fast* in the sense that we are able to learn a fairly useful category representation just by encountering instances of the

[2]The explicitness in the last example is weaker than in the examples of linguistic level learning. Thus, it would be more appropriate to place this type of learning between the two main categories above.

[3]Here we refer to the normal, rather *passive*, concept acquisition process. However, in some situations we adopt a more *active* strategy, where we concentrate on one concept at the time.

category on one or a few occasions. For instance, if we are hurt once by an animal of a (to us) unknown species, we are often able to recognize other animals of this species later, and infer that it is able to hurt us.

As Schank et al. [237] point out, any dynamic and autonomous theory of concept acquisition must specify at least three processes:

1. Deciding when to create a new concept.
2. Deciding when to modify a concept.
3. Deciding what part of the concept to change.

Theories of learning by being explicitly taught, however, do not have to specify the first process since it is assumed that this is done by the teacher (i.e., the category is already formed).

9.1.1 Theories of Concept Acquisition

As pointed out earlier, all our knowledge about categories cannot be innate. However, it is possible, and even plausible, that some knowledge about categories is innate. Different researchers emphasize this to different degrees. Fodor's [92] theories of cognition, for instance, rely heavily on innate knowledge.

If it is not the case that all concepts are innate, then some of them must be acquired in some way. How this is done has, of course, been the subject of research in cognitive psychology. The three most predominant psychological theories of human concept acquisition has been:

- the association theory
- the hypothesis testing theory
- the exemplar strategy.

The *association* theory as described by Solso [257] seems rather outdated, with its roots in stimulus-response psychology. It holds that the learning of a category representation is the result of (1) reinforcing the correct pairing of a stimulus with the response of identifying it as a category, and (2) non-reinforcing (punishment) the incorrect pairing of a stimulus with a response of identifying it as a category. This theory seems to cover only the case of being explicitly taught something about the category on the perceptual level. Moreover, it is extremely vague and therefore consistent with most other theories of concept acquisition. For instance, it does not specify what part of the concept is being changed.

The theory of *hypothesis testing* states that “we hypothesize what properties are critical for determining whether an item belongs to a category, analyze any potential instance for these critical properties, and then maintain our hypothesis if it leads to correct decisions.” [16] In other words, it adopts the classical view and assumes that the category can be characterized by a classical definition, and it seems to assume that all instances of the category are concurrently available for analysis. These assumptions are too strong for most learning situations. Moreover, the theory does not specify when to create a new concept, is non-incremental, and learns only one concept at a time. In fact, the hypothesis testing theory seems more like a model of learning by experimentation, like when a scientist is doing experiments.

Finally, the *exemplar strategy* simply states that when encountering a known instance of a category a representation of it is stored. Thus, this theory is consistent with the exemplar view. However, since it seems implausible that we remember every instance we ever encountered, this simple version of the theory has to be modified in some way. Thus, several questions remains open, for example: How many, and which, instances should be memorized? Moreover, the strategy is only specified for learning by being explicitly taught. However, it seems possible to extend the theory to include learning from experience, but then, *when* to create a new concept must be specified. Advantages with the exemplar strategy are its incremental nature and that it accounts for the acquisition of many concepts at the time. Thus, the exemplar strategy is the only theory of the three that has at least a chance of being adequate.

Neural Network Approaches

In the last years neural network models of concept learning has become popular among cognitive psychologists.[4] The simplest type of neural network for supervised concept learning is the (single layer) *perceptron*. It consists of a number of input nodes, each corresponding to a feature in a feature vector, which are connected to an output node. The perceptron is able to classify the input vector into either of two categories and can be used for both continuously valued (dimensions) and binary (features) inputs. A detailed analysis of the capabilities and limitations of perceptrons is provided by Minsky and Papert [186]. It is, for instance, proved that if a perceptron can classify a series of inputs, then it is also able to learn that classification. A severe limitation is, however, that perceptrons can only classify linearly separable categories (i.e., categories that can be separated by a straight line in a 2-dimensional feature space).

The multi-layer perceptron is a generalization of the single layer perceptron that is able to learn classifications of categories that are not linearly separable. It has one or more additional layer(s) of nodes, called hidden layer(s), between the input and the

[4] As mentioned in the last chapter, the research in neural networks belongs to the engineering approach just as much as to the cognitive modeling approach.

output nodes. A two-layer perceptron can form any convex region in the feature space and a three-layer perceptron can form arbitrarily complex regions. The algorithm most often used for learning multi-layer perceptrons is called the *backpropagation algorithm* [232]. Although it has not been proven that this algorithm can learn every classification the network can represent, it has been shown to be successful for many problems. A disadvantage is, however, that the algorithm is non-incremental.[5] Like most other neural network approaches the backpropagation algorithm has scaling problems, which means that, when the number of training examples and/or the number of categories to be learned gets larger, the algorithm will be too slow and use too much memory. One example of this approach is Gluck and Bower's [104] *adaptive network model*, which is a two-layer perceptron trained by a backpropagation algorithm. Also hybrid approaches combining neural network models with exemplar strategies has been suggested (e.g., Estes [79] and Taraban and Palacios [265]).

Kohonen's [144] *self-organizing feature maps* is a neural network approach to unsupervised learning. These networks consist of a number of input nodes that are connected to each of the nodes in a (typically) two-dimensional array of output nodes. The result after the learning phase, which is based of a winner-takes-all principle, is a mapping where similar inputs are mapped to nearby output nodes where each output node, or rather neighborhood of nodes, represents a category. These networks are able to handle both continuously valued and binary inputs. In contrast to the other conceptual clustering systems presented here, the self-organizing feature maps do not form a hierarchy of categories.[6] The network decides by itself the number of categories to be formed.

Another neural network approach to unsupervised learning is Grossberg and Carpenter's [47] adaptive resonance theory (ART) networks. The structure and function of these networks are to a higher degree than most other neural networks based on biological and behavioral data. The desire to replicate learning in humans, who actually are autonomous agents, has resulted in a network with some interesting features. For instance, the ART networks learn in an incremental fashion. Another positive aspect is that when fast-learning[7] is used, it requires, in contrast to most other networks, only one pass (at most) through the training set to learn the category representation. Furthermore, the network has proven to be stable and does not suffer from any convergence problems (e.g., local minima). While ART-1 only takes binary properties, ART-2 can deal also with numeric properties. However, it does only form categories at one level and to decide the number of categories to be formed (or, rather, the metrical size of the categories), one must choose a vigilance threshold. ART-MAP [48] is a supervised version of the adaptive resonance theory.

[5] However, some variants of the algorithm are claimed to behave incrementally.

[6] At least not the kind of hierarchies that have been described here.

[7] The ART networks have two training schemes often referred to as fast and slow learning.

9.2 AI Methods for Concept Acquisition

The concept acquisition process of an artificial autonomous agent is from a general point of view restricted by the environment in the same way as that of a human. Thus, from the earlier discussion we can conclude that for artificial autonomous agents the concept acquisition process must be incremental and relatively fast, concepts must be acquired in parallel, and several methods must be employed simultaneously.

9.2.1 ML Methods for Concept Acquisition

In AI, several ways of learning about categories have been studied, the most predominant being:

- direct implanting of knowledge
- learning from examples
- learning by observation
- learning by discovery
- learning by deduction.

The following sections will describe these paradigms in greater detail.

Direct Implanting of Knowledge

Direct implanting of knowledge is the extreme, almost trivial, case of concept acquisition in which the learner does not perform any inference at all on the information provided. It includes learning by direct memorization of given category descriptions and the case when the descriptions are programmed directly into the system. The latter can, from the perspective of an autonomous agent, be seen as a way of incorporating innate, or a priori, knowledge about concepts into the agent. However, one should be careful when doing this since the category representation probably will to some extent reflect the programmer's conception of the category which may not necessarily coincide with what would be optimal for the system. This is related to the problems of symbol grounding.

Learning by instruction, or learning by being told, is similar to direct implanting of knowledge in that the learner acquires concepts (explicitly described on the linguistic level) from a teacher, database, textbook or some other organized source. However, this form of learning, in contrast to direct implanting of knowledge, requires selecting the relevant information and/or transforming this information to a usable form.

Learning from Examples

Learning from examples is by far the most studied type of learning in AI and can be seen as a parallel to learning by being explicitly taught. From a number of preclassified examples and counterexamples of the category provided by some kind of teacher, the learner induces a category description. Since there is a teacher present to guide the learning process, this type of learning is an instance of supervised learning. Thus, it is the teacher who decides when to create a new concept. From the classical viewpoint, the task for this type of learning can be seen as finding a concept definition (i.e., description) consistent with all positive examples but no negative examples in the training set.

Some of the systems learning from examples can be viewed as carrying out a search through a space of possible category descriptions. This space can be partially ordered, with the most general description at one end and the most specific at the other. The most general description has no features specified, corresponding to the set of all possible instances, whereas the most specific descriptions have all features specified, corresponding to individual instances. There are basically two strategies for searching the space of category descriptions. In the general-to-specific, or model-driven, strategy, one begins with the most general description as the hypothesis of the correct category description, and as new instances are encountered, more specific descriptions (i.e., hypotheses) are produced. The specific-to-general, or data-driven, strategy, on the other hand, begins with a very specific description, typically a description of the first instance encountered, moving to more general descriptions as new instances are observed. Some systems use one or the other of these strategies, while more sophisticated systems, like those based on *version spaces* developed by Mitchell [188], combine the two strategies. Since there is no inherent non-incrementality in this approach, it seems possible to make systems that learn incrementally based on this approach.[8] The version spaces approach learns classical definitions in a logic-related notation and is suited for nominal properties but have problems with ordered and structural properties. Learning systems of this kind typically learn just one concept at a time, without considering other known category descriptions. An exception to this is AQ11 [181, 179] by Michalski and his colleagues, which learns multiple concepts. Another exception is a system by Gross [110] that incrementally learns multiple concepts where the category description currently learned is constrained by the descriptions of the other categories. However, this system can be interpreted as learning by experimentation, since it is the system itself that selects the next instance to be analyzed from a given description space. This instance is then classified by an oracle. The introduction of an oracle being able to classify every possible

[8]However, some systems, version spaces for instance, have several competing hypotheses at some stages in the learning process. Having several hypotheses makes it difficult to use the concept and requires more memory space. Nevertheless, the memory requirements of such systems are substantially less than for systems that must memorize all instances, such as Winston's [282].

instance makes the learning easier but is an unrealistic assumption in the context of autonomous agents. However, this algorithm seems to be consistent with the hypothesis testing theory of human concept acquisition.

A different kind of learning-from-examples system is the so called top-down induction of decision trees (TDIDT) systems, such as ID3 [217]. These systems accept instances represented as lists of attribute-value pairs as input and produce a decision tree as output. TDIDT systems, which are model-driven, begin with the root of the tree and create the decision tree in a top-down manner, one branch at a time. At each node they use some evaluation function, often based on information theory, to determine the most discriminating attribute. The evaluation function is based on the number of positive and negative instances associated with the values of each attribute. An advantage of TDIDT systems is that they carry out very little search, relying on the evaluation function instead. A serious limitation is, however, their non-incremental nature. To incorporate new instances, the tree often has to be recomputed from scratch. Moreover, since they require feature-vector representations of the training instances, they are not suited for dealing with structural information.

While all systems described above have adopted the classical view, we will now look at some systems that use non-classical representations. Kibler and Aha [139] describe three algorithms that learn from examples using an exemplar representation of categories. The *proximity* algorithm, sometimes called IB1 [5], simply stores all training instances. As all computation takes place in the classification phase (in which the similarity to each training instance is computed), learning of this kind are sometimes referred to as *lazy learning*. The *growth*, or additive, algorithm stores only those training instances that would not be correctly classified if they were not stored. These two algorithms are incremental in contrast to the third, the *shrink*, or subtractive, algorithm. The shrink algorithm begins by placing all the training instances into the category representation, and then continues by testing each instance in turn to see if it would be correctly classified using only the remaining instances.

Nagel [198] presents another system that learns incrementally from examples using an exemplar representation. When a positive instance is presented to the system, the system will try to find a sequence of transformations that transforms the instance into a prototypical instance. The new transformations are then stored as a part of the category description to be used for assimilating new instances.[9] De la Maza's PROTO-TO system [61] also learns incrementally from examples but uses a probabilistic representation. It groups the instances according to their categories and then builds a prototype for each category. The prototypes are then augmented, weighting each attribute in order to form a probabilistic representation.

[9]How the prototypes are learned in the first place is not described in the material that, for the moment, is available to me (i.e., [197]).

Learning by Observation

In contrast to learning from examples, a system performing learning by observation has to form categories by itself. Thus, it is the learner that decides when to create new concepts. In learning from examples the categories were formed beforehand and there was a teacher present who provided examples of them (where the category membership was explicitly stated) to the learning system. Thus, the learning process was supervised. In, learning by observation, on the other hand, the learning is unsupervised. Typically, the learner is given a number of descriptions of entities (with unknown category membership). It groups the entities into categories based on their features, a process often referred to as *aggregation*. When this is done, the system creates descriptions of the categories, a process often called *characterization*.

Tasks similar to the aggregation task has been studied for a long time within statistics under the labels *cluster analysis* and *numerical taxonomy*. In these contexts the observations, or instances, are typically described by a number, *n*, of numerical features and treated as points in an n-dimensional space. Some clustering algorithms build hierarchies of categories, i.e., hierarchical methods, whereas other algorithms cluster the observations only at one level, i.e., optimization methods. It is common to further divide the hierarchical methods into agglomerative and divisive methods. *Agglomerative methods* work in a bottom-up fashion, beginning with joining similar separate observations together to form small clusters. By recursively forming larger and larger clusters, the process eventually halts when all observations belong to a single universal cluster. In this way the algorithm builds, step by step, a hierarchy of clusters. *Divisive methods* work in the opposite way, beginning with a single universal cluster and then repeatedly breaking it into smaller and smaller clusters until only single observations exist in each cluster. *Optimization methods*, on the other hand, form clusters at a singular level by optimizing the clustering according to user-provided information such as the number of clusters to form and desired cluster size. More details on clustering algorithms are provided by Jain and Dubes [131].

The clustering algorithms described above all use some kind of similarity measure for the aggregation task which, in turn, depends on some kind of distance metric. As pointed out in Section 7.3.2, there are several problems associated with such metrics. An interesting optimization clustering technique that does not use a distance metric is suggested by Matthews and Hearne [171]. The clusterings are instead optimized on the intended function of the clustering, which, according to the authors, is the prediction of unknown feature values. Thus, this approach aims at maximizing the utility of the clustering and is thus related to the implicit similarity measures discussed in Section 7.2.2.

The characterization task, creating descriptions of the categories, is in principle equivalent to the task of learning from examples as described above. This suggests that one way to perform learning from observation would be to employ a statistical clustering algorithm for the aggregation task and then to use one of the algorithms presented in the last section to create descriptions of the categories. There are, however, some problems associated with such an approach. Besides being limited to numerical feature values, the aggregation step would be totally independent from the characterization step, not taking into account the language used to describe the resulting concepts. This would result in clusters that may not be well characterized (i.e., comprehensible by humans) in the chosen description language. Rather than relying on a pure metric similarity measure Michalski and his colleagues [182] introduced the notion of conceptual cohesiveness described in Section 7.2.2.

Systems which integrate the aggregation and characterization steps are commonly called *conceptual clustering* systems. Some of the most influential are CLUSTER/2 [182, 260] and RUMMAGE [86] which both characterize the formed categories by classical descriptions. Although these systems learn in a non-incremental fashion, incremental systems that use classical concept definitions, like UNIMEM [152], exist. As should be clear from above, conceptual clustering systems form concepts in parallel (i.e., many at the time). Moreover, while other types of systems usually learn concepts at a single level, most conceptual clustering systems structure the created concepts into taxonomies (cf. hierarchical clustering methods). Whereas conceptual clustering systems typically do not form structural concepts, CLUSTER/S [261], which is version of CLUSTER/2, do.

An example of an approach that uses non-classical representations is the PLANC system by Musgrove and Phelps [196] that learns from observation by a clustering algorithm that first applies multidimensional scaling to reduce the dimensionality of the input data. When the clusters are detected, their members are used to produce a prototype (i.e., a hypothetical average member). Whereas this system is non-incremental, COBWEB [87, 88] acquires concepts in an incremental fashion. COBWEB builds a probabilistic concept tree. As mentioned in Section 7.2.2 it uses category utility as an evaluation measure of clusterings in the aggregation task, instead of a distance metric. Since, this measure was originally developed as a means of predicting the basic level in human taxonomies, one can interpret COBWEB's strategy as trying to form basic levels on every level, beginning at the top of the tree. It is similar to Matthews and Hearne's approach in that it tries to maximize the predictive ability of the clustering. LABYRINTH [267, 268] is an adaption of COBWEB able to handle also structural descriptions.

Learning by Discovery

Learning by discovery is, like learning by observation, a kind of unsupervised learning. However, systems that learn by discovery are more active in their search for new categories than systems learning by observation. They exploit their domain, sometimes by experiments, rather than passively accepting some given descriptions of instances.

The most famous system of this kind is Lenat's AM system [155, 156]. AM works in the domain of mathematics and searches for and develops new "interesting" categories after being given a set of heuristic rules and basic concepts. It uses a "generate-and-test" strategy to form hypotheses on the basis of a small number of examples and then tests the hypotheses on a larger set to see if they appear to hold. Surprisingly, the AM system worked (or appeared to work) very well. From a few basic categories of set theory it discovered a good portion of standard number theory. However, outside this domain AM does not work very well. Two of the reasons for this are that there are difficulties in specifying heuristics for other less well-known domains, and that in the implementation of AM implicit knowledge about number theory was built-in. Moreover, even though AM initially performed well in the domain of number theory, its performance decreased after a while and it was not able to discover any new interesting categories. This was due to the static nature of the heuristics, which did not change when the system's knowledge about the domain increased, resulting in a static system. Thus, if such a system is to be more dynamic, it must also be able to reason and manipulate with its heuristics. For a more comprehensive discussion of this topic, see Lenat and Brown [157].

Deductive Learning

In deductive learning, the learner acquires a category description by deducing it from the knowledge given and/or already possessed (i.e., background knowledge). The most investigated kind of deductive learning is *explanation-based learning* (EBL) [189, 64] that transforms a given abstract category description (often based on non-perceptual features) to an operational description (often based on perceptual features) using a category example (described by operational, or perceptual features) and some background knowledge for guidance.

The standard example of EBL concerns the category "cup". In this example the abstract category description is a classical definition that says that a cup is an open, stable and liftable vessel. The background knowledge includes information such as: if something is light and has a handle then it is liftable, if something has a flat bottom then it is stable, and so on. Given this and an example of a cup in terms of perceptual features (e.g., light, has a handle) and the operationality criterion that the category description must be expressed in terms of the perceptual features used in the example, the EBL-system produces a description of the category "cup" that includes the facts that a cup is light, has a handle, has a flat bottom, and so on. Seen in the light of the core versus identification procedure view of concepts described in Section 7.1.1, we can interpret the function of an EBL system as taking the core of the concept (among other things) as input and producing an identification procedure as output.

This form of learning is clearly a kind of top-down learning, since the learning is triggered by the goals of the learner. It can, as has pointed out earlier, be seen as nothing but a reformulation of category descriptions, since an abstract description of the category is given to the system. Thus, no new categories are created.

9.2.2 Computer Vision Approaches

The models learned by vision systems are intended to be used for perceptual recognition tasks and can thus be interpreted as being epistemological representations. However, as mentioned earlier, they are often interpreted as models of particular objects rather than categories.

In the past, there has only been a few studies of the learning of concepts (i.e., object models) by direct perception of objects. However, the AAAI Fall Symposium on Machine Learning in Computer Vision in 1993 [2] showed that this state of affairs is changing. In this symposium a number of approaches were presented covering both the more traditional 2-D and 3-D models as well as characteristic views models. However, most of the research was still in progress and had not yet been applied to realistic situations (i.e., receiving data directly from video cameras, or other sensors, that perceive non-laboratory scenes). Typically, the systems were given line-drawings as input[10] and/or could not cope with occluded objects.

2–3-D Model Approaches

One of the earliest studies of the learning of object models from real images was made by Connell and Brady [56]. Their system, of which the learning component is a modified version of Winston's ANALOGY [283] program, learns semantic network representations from examples. Input to the system is a set of grey-scale images of real objects such as hammers and airplanes. However, the system is limited to learning 2-D shapes and only works well with objects composed of elongated pieces.

Another approach using semantic network representation is suggested by Dey et al. [67]. It is an unsupervised system that incrementally acquires a hierarchy of concepts, but seems to have the same weaknesses as Connell and Brady's system. In general, it is interesting to note that so few, if any, of the learning vision systems use 3-D volumetric representations.

An interesting idea within this paradigm is, however, the adaptive feature extraction framework presented by Bhandaru, Draper, and Lesser [28]. It integrates feature extraction and learning object models, instead of treating these as separate processes.

[10]This applies more to the 2-D and 3-D than the characteristic view approaches which were often given real images.

Characteristic View Approaches

These systems are based on the principle that having enough 2-D views of an object is equivalent to having its 3-D structure specified. As an example, Pope and Lowe [213] present an approach that learns a set of characteristic views from pre-classified examples. An incremental conceptual clustering, similar to COBWEB, is used to construct (identify) the characteristic views from the examples. When recognizing (classifying) new images a probabilistic similarity metric is used to compare the image with the characteristic views. Another approach is suggested by Poggio and Edelman [211], who have implemented a neural network that learns to transform an image from any viewpoint into a standard view. From this standard view it is easy to recognize which category the object belongs to. In the article, however, they only tested the network on simple line-drawings.

Murase and Nayar's approach to learning appearance models, a continuous version of characteristic views, from examples was presented in Section 8.2.3. While it seemed to have some interesting features, there are some problems with the approach that makes it problematic to apply directly. For instance, it is probably difficult to obtain all the images necessary for learning an adequate representation. Moreover, it is assumed that objects are not occluded by other objects and that they can be segmented from the background of the scene. Furthermore, the learning is batch-oriented and it seems not possible that it could be performed in an incremental fashion.

In general, it seems very hard to introduce symbolic knowledge (i.e., direct implanting of knowledge) to this kind of systems. The representation of objects, or categories, is on a sub-symbolic level and, moreover, heavily dependent on the system's sensors.

9.2.3 Pattern Recognition

Pattern recognition methods are often divided into *statistical* and *syntactical*, or structural, methods. Whereas we have already described some statistical methods, the syntactical approach will provide a new view on the problem of concept representation and acquisition.

A fundamental question that should be answered before we continue, is whether a pattern can be interpreted as a concept (or, rather, as a description of an instance). Actually, we made such an interpretation when we discussed neural networks. Neural networks (as well as some other systems described above) are in fact a kind of pattern recognizers. The features that are used to characterize an object constitutes a pattern. Thus, if a pattern recognition system can recognize a pattern corresponding to an object, it can ideally also recognize the object, and consequently the system would implement the epistemological (or metaphysical) function.

Statistical Pattern Recognition

Statistical methods [72] are based on statistical studies of the input (i.e., feature vectors) in order to recognize patterns. There are two major types of methods, parametric, or Bayesian, and non-parametric. The aim in *parametric* methods is to decide, on the basis of a model for the distribution of the instances of each category, to which category an unknown instance has the greatest probability of belonging. The decisions are based on statistical decision theory. In the basic versions of the methods, it is required that the statistical distributions, $p(v|c)$ (where v is the input vector, the instance, and c is the category), are known. Since this assumption is too strong in most cases, a parametric system often has to *estimate* the distributions from the training instances. If the general form of the distribution is known (e.g., Gaussian), this task reduces to the estimation of a finite number of parameters. The disadvantages with this approach are that the distribution of instances within the category often is not sufficiently regular, and that it requires a large number of training instances [183].

The *non-parametric* methods, on the other hand, do not assume that the form of the underlying statistical distribution is known, rather they try to estimate it using the given training instances only. Their aim is to find the boundaries of the different categories in the description space, so that a series of simple tests will suffice to categorize an unknown instance into one of the known categories. One way of doing this is to learn a *linear classifier*, i.e., to find a hyperplane that divides the description space into two parts (in the two-category case). The perceptron, described in the last chapter, is an example of a linear classifier. This approach demands, of course, that the categories be linearly separable. Another popular method is that of *nearest neighbor* in which the unknown instance is categorized into the category of its nearest neighbor. In fact, this is exactly the same algorithm as the instance-based proximity algorithm described above.

The methods described above are all supervised (cf. learning from examples) algorithms. However, statistical clustering algorithms as described on page 127, are unsupervised methods that sometimes also are regarded as pattern recognition methods. A more sophisticated algorithm, called AUTOCLASS is provided by Cheeseman et al. [52]. It is based on the parametric (Bayesian) approach and “automatically determines the most probable number of classes, their probabilistic descriptions, and the probability that each object is a member of each class.” Thus, rather than assigning the instances to specific categories, it derives probabilities of the instances belonging to a category. This is not to be confused with probabilistic representations, as described in Chapter 8, which vary in their degree of membership, not their probability of membership. The algorithm need not be told the number of clusters to form,[11] and need, according to Cheeseman et al. [53], no ad hoc similarity measure or clustering quality

[11] It will arise naturally because of an optimization of a trade-off situation between forming small and large clusters

criterion. Another interesting feature of AUTOCLASS is that it can easily be adapted to perform supervised learning. On the other hand, a clear disadvantage is that the algorithm is inherently non-incremental. However, an incremental algorithm that also relies on the Bayesian method, is presented by Anderson and Matessa [11]. In contrast to AUTOCLASS, this algorithm assigns each instance to a specific category.

A disadvantage with statistical pattern recognition methods in general, is that they assume feature vector representations, and are thus not suitable for structural information. Syntactical methods, on the other hand, emphasize this kind of information rather than the metric properties emphasized by the statistical methods.

Syntactical Pattern Recognition

In syntactical methods patterns (i.e., instances) are typically viewed as sentences in some formal language. Consequently, categories are represented by grammars in that language. The recognition of an instance as a member of a category is accomplished by parsing the sentence corresponding to the instance to determine whether or not it is syntactically derivable using the grammar corresponding to the category. To learn a concept then corresponds to inferring a grammar from a number of sentences from which all sentences can be derived.

There are many different types of grammar inference algorithms (cf. [183]). However, most of these learn only from examples (supervised learning) and just one concept at a time. Moreover, since traditional formal languages typically are used, they learn concepts that correspond to classical definitions. Despite these negative aspects, it may be fruitful to try to apply, or combine, the theories developed within this paradigm to (with) the ML framework. As pointed out by Honavar [126], a particularly interesting approach would be to use template matching, described in the last chapter, wherein an instance is matched to one or more stored instance(s) for each of the categories. This approach is very similar to the instance-based algorithms discussed above (and to the non-parametric statistical method, called nearest neighbor), but is able to deal also with structural information. As with other algorithms of this kind, some kind of similarity measure is needed when matching two instances.

A serious limitation of syntactical methods is that representation of instances by normal sentences (i.e., linear strings) permits the encoding of only a single structural relation between the representational primitives; a primitive can only precede or succeed another [126]. To represent, for instance, complex structural relationships of 2- or 3-dimensional objects, more powerful grammars are needed, such as web grammars and tree grammars. A generalized similarity measure, applicable to arbitrarily structured patterns, is presented by Honavar [125]. In line with the models of structural similarity presented earlier, this approach defines similarity between two descriptions in terms of how difficult it is to transform one description into the other.

9.3 What can be learned?

In the field of *formal learning theory* the term *inductive inference* has come to denote the process of hypothesizing a general rule (e.g., a classical concept definition) from positive examples of the rule.

Although there exist several paradigms within formal learning theory, we will here concentrate on the two most dominant. Traditionally, most of the research has been carried out within the paradigm formulated by Solomonoff [256] and established by Gold [107], which will here be referred to as Gold's paradigm. However, in the mid eighties Valiant [277] formulated a paradigm that was more closely related to the on-going research on concept learning in ML. In short, one can say that Gold's approach addresses the question of whether there exists an algorithm for learning the concept definition (i.e., computability), whereas Valiant's approach addresses the question of whether there exists an *efficient* algorithm (i.e., computational complexity).

According to Angluin and Smith [12], the following items must be specified to define an inductive inference problem:

- the class of rules being considered
- the hypothesis space
- for each rule, its set of examples, and the sequences of examples that constitute admissible presentations of the rule
- the class of inference methods under consideration
- the criteria for a successful inference.

The class of rules is usually a class of functions, boolean expressions, or formal languages. The hypothesis space is a set of descriptions such that each rule in the class has at least one description in the hypothesis space.

9.3.1 Gold's Paradigm

Gold's article presents an investigation, motivated by the psycholinguists' study of the acquisition of grammar by children, concerning to what extent different classes of languages can be learned from positive examples only.[12] Other names for this paradigm are *the identification paradigm* [204] and *grammatical inference* [151]. The last name

[12]If one wants to study the actual learning process of natural language grammar by children this model seems to be a poor one, since it only takes into account the syntactic component and does not bother about the semantic and pragmatic issues. Furthermore, it does not seem plausible that this type of learning is based only on positive examples. (Although Chomsky and others believe it is.)

suggests that it is related to learning in syntactical (structural) pattern recognition. In fact, this is to a great extent a theoretical treatment of the syntactic approach. However, Blum and Blum [33] later transferred the posing of the problem to identification of functions instead of grammars.

In his paper Gold introduces *identification in the limit* as a criteria of success, which can be formulated as follows. Suppose that *M* is an inductive inference method attempting to describe some unknown rule *R*. If *M* is run repeatedly on larger and larger collections of examples of *R*, an infinite sequence of *M*'s conjectures is generated. If there exists a number, *n*, such that the *n*th conjecture is a correct description of *R* and that all the conjectures that follow are the same as the *n*th, then *M* is said to identify *R* (correctly) in the limit on this sequence of examples. Note that Gold here views inductive inference as an infinite process, and that *M* cannot determine whether it has converged to a correct hypothesis (conclusion).

Identification by enumeration is an example of an inference method. It systematically searches through the space of possible rules until one is found that agrees with all the data so far. Suppose that a particular domain of rules (category descriptions) is specified, and that there is an enumeration of descriptions $(d_1, d_2, ...)$ such that each rule in the domain has one or more descriptions in this enumeration. Given any collection of examples, we just have to work through the list of descriptions until we find the first description that is compatible with the given examples and then conjecture it. The method is guaranteed to identify in the limit all the rules in the domain if the following conditions are satisfied:

- A correct hypothesis is always compatible with the examples given.
- Any incorrect hypothesis is incompatible with some sufficiently large collection of examples (and with all larger collections).

The method is computable if the enumeration $(d_1, d_2, ...)$ is computable and if it is possible to compute whether a given description and a given collection of examples are compatible.

However, since the issue of computational feasibility has not been central to this paradigm, many of the positive results (i.e., that something is learnable) have relied on algorithms, such as identification by enumeration, that are intractable with respect to time or/and space. Moreover, many of the negative results have been due to the fact that the learning domains have been too general to allow any algorithm to (ever) distinguish the target concept from among other possible hypotheses [209].

9.3.2 Valiant's Paradigm

Valiant's paradigm, often referred to as *Probably Approximately Correct (*PAC*) learning*, has become more popular than Gold's since it addresses more of the requirements that are typically placed on a learning algorithm in practice. Since its criteria of success is that a good approximation of the target concept should to be found with high probability (rather than exact identification), it allows greater emphasis on computational efficiency.

Valiant's framework considers the learning of category descriptions in the form of Boolean vectors of features. The learning system is given positive and negative examples of the target concept. The learner must produce, with at least the probability 1 – θ, a category description that incorrectly classifies future examples with probability lesser than or equal to ε. If there exists an algorithm that is able to accomplish this task for any target concept in the concept class in time polynomial in the size of the target category description, $\frac{1}{\theta}$, and $\frac{1}{\varepsilon}$, then the class is said to be learnable.[13] An example of a concept class that has proven to be learnable is the class of conjunctions of boolean variables. For more results PAC-learnability see, for instance, Kearns et al. [138].

9.3.3 Critique of Current Formal Learning Theory

The obvious critique of current formal learning theory is that it only regards classical concept definitions. Moreover, most approaches do only regard learning from examples. Pitt and Reinke [210], however, have considered unsupervised learning. They have developed a formalism for analyzing conceptual clustering algorithms. As criteria for success they have chosen the ability to efficiently produce a clustering that maximizes a given objective function based on individual cluster tightness and overall distance between clusters. They show that under a wide variety of conditions, the agglomerative-hierarchical algorithm can be used to find an optimal solution in polynomial time.

Although Valiant's paradigm is more closely related to the current research in ML than Gold's, it has been criticized for not being so to a sufficient degree. For instance, Buntine [45] claims that it can produce overly-conservative estimates of errors and that it fails to match the induction process as it is often implemented. Thus, while being possibly interesting from a theoretical viewpoint, the current approaches to formal learning theory are only of limited interest in the context of autonomous agents.

9.3.4 The Conservation Law for Generalization Performance

There is, however, one theoretical result that one should keep in mind when constructing concept learning algorithms, namely the *conservation law of generalization performance* (also referred to as the "no free lunch theorem"). It is described by Schaffer [235] in the following way: for any learner L,

[13] This is a simplification of Valiant's original formalization. My hope is, however, that it probably approximately resembles his central ideas.

$$\sum_S GP(S) = 0 \qquad \text{for every } D \text{ and } n,$$

where S is a learning situation defined by: D, the distribution of instances, C, the mapping from instances to categories, and n, the number of training instances. The generalization performance, GP is defined as the expected prediction performance on instances *not* represented in the training set.

The conservation law says that the total generalization performance over all learning situations is null, i.e., positive performance in some situations are exactly balanced by negative performance in others. Thus, in theory, no learning algorithm is better than any other for generalization. At first glance, this seems as a quite negative result, but what implications does the law actually have for real world problems?

One assumption of the conservation law is that we have no prior knowledge about C. However, it seems probable that the occurrence of mappings between instances and categories useful in the real world is not uniformly distributed. As a consequence, we should construct concept learning algorithms that exploit these regularities since such algorithms would have positive generalization performance on real world problems. Fortunately, this is what has happened; researchers have developed algorithms that performs well on real world problems. But then, what knowledge about C is it that is used to achieve positive generalization performance on real world problems? In my opinion, the most important is the fact that similar instances often belongs to the same category. In fact, it seems that all algorithms make this assumption (sometimes implicitly).

Thus, for autonomous agents an algorithm for learning bottom-up categories (natural kind etc. cf. Chapter 7) should be good at learning categories in which the instances are similar. For learning top-down categories, on the other hand, it is not obvious what kind of prior knowledge is relevant. Although the practical consequences of the conservation law probably will not be noticeable, it is a useful reminder that pure induction (i.e., generalization) is not computable.

9.4 Requirements for Autonomous Concept Learning

In this section we will discuss the requirements for an autonomous concept learning system, the extent these demands have been met by existing AI systems, and what could be done to meet them.

9.4.1 Incrementality

In early ML research, most methods were non-incremental, or batch-oriented, in the sense that they had two separate phases, an initial training phase followed by a classification phase. An autonomous agent, however, should be able to learn new concepts throughout its life-cycle, i.e., it should never stop learning. It would, of course, be possible to let the agent use a non-incremental learning system, memorizing all previously encountered objects, and recomputing the whole learning procedure every time a new object is encountered. For practical reasons, such as time and memory requirements, this is not an adequate solution to the problem. Instead, the learning system must be able to switch modes between learning and classification without destroying any previous learning. In recent years, however, many incremental systems have been constructed. For each approach to concept acquisition there often also exist an incremental version, unless the approach is inherently non-incremental.

Incremental learning is also the basic feature of the *active agent* paradigm suggested by Pachowicz [205]. In addition, he suggests a closer integration between vision and learning than just finding bridges between stand alone vision and learning modules. In the article he points out some important problems along with some hints for the solution of these.

9.4.2 Learning Multiple Concepts

There exist many systems that learn more than one concept at a time, for instance, conceptual clustering systems and some non-traditional learning-from-example systems. However, in traditional learning-from-example systems, knowledge about known categories and taxonomies is typically not used to constrain the hypothesis space. How this should be done seems like an important area of research (especially if one is interested in the metaphysical functions of concepts).

9.4.3 Fast Learning

The learning speed of algorithms has been a topic that for obvious reasons has received considerable attention. Many empirical comparisons of different algorithms have been conducted (cf. [281, 190, 280, 89]) and the general conclusion of these studies is that traditional symbolic methods (e.g., ID3) learn faster than neural network approaches (e.g., backpropagation).

However, the number of training examples used in these comparisons is typically very large (100 – 10000 instances). From an autonomous agent point of view, it is equally interesting to study learning from just a few training instances of each particular category. It seems that an empirical comparison of this kind has never been made, but a qualitative statement is provided by Fisher and Pazzani [90] who write that:

> ...specific-to-general learners can more quickly exploit relevant information for purposes of prediction than can their general-to-specific counterparts. For example, a system can ideally distinguish `mice` from `men` with high, but not perfect, accuracy after a single example of each." (p. 30)

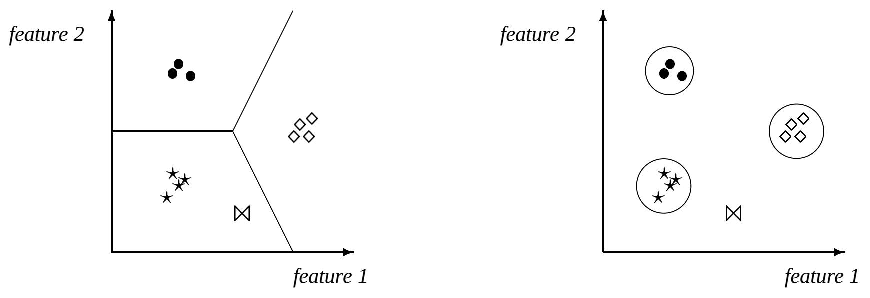


Figure 9.1: Discriminative vs. characteristic category descriptions.

Thus, most neural networks (with the exception of ART-networks) learn too slowly in the sense that they need many instances (and epochs) to learn a fairly good representation. The reason is that since the weights in neural networks are often randomly chosen, the behavior of the net is unpredictable in early stages leading to misclassifications. On the other extreme we have, for instance, instance-based algorithms that under favorable circumstances (i.e., conjunctive categories that cover convex areas in the description space) only need a single prototypical training instance of each category to achieve acceptable classification performance.

9.4.4 Characteristic Descriptions

One of the key problems for an autonomous learning system is to decide when to create a new concept. The learning system needs to know when it encounters an instance of an unknown category; if the system knows, or believes that the current observation belongs to an (for the agent) unknown category, it seems rational to create a new category based on this instance. The only way of knowing this is, according to Smyth and Mellstrom [255], to learn *generative*, or *characteristic*, category descriptions that try to capture the similarities between the members of the category instead of learning *discriminative* descriptions that are representations of the boundaries between categories. The difference between these kinds of descriptions is illustrated in Figure 9.1. It shows some instances of three known categories ($\star$, $\bullet$, and $\diamond$), and examples of possible category boundaries of the concepts learned by a system using discriminative descriptions (to the left) and by a system using characteristic descriptions (to the right). In this case, a member of an unknown category ($\bowtie$) will be categorized wrongly by a system using discriminative descriptions, whereas it will be regarded as a member of a novel category by a system using characteristic descriptions. In other words, whereas systems that learn discriminative descriptions tend to overgeneralize, the degree of generalization can be controlled, or at least limited, by systems learning characteristic descriptions.

Decision trees constructed by ID3-like algorithms, as well as, for instance, nearest neighbor algorithms and neural networks learned by backpropagation,[14] suffer from this inability of detecting examples of categories not present in the training set. Of course, there exist methods for learning characteristic descriptions from examples, for instance, the algorithm presented by Smyth and Mellstrom [255], some neural net algorithms (e.g., ART-MAP), and certain kinds of instance-based methods. The problem with these is that they do not learn explicit rules, which is desired in many practical applications such as the coin classification task outlined earlier. However, as has been shown by Holte et al. [123], the CN2 algorithm can be modified to learn characteristic descriptions in the form of rule-based *maximum specific descriptions*.

It is possible make distinctions between different kinds of algorithms learning characteristic descriptions in terms of how much they generalize. On one end we have those that learn the most general descriptions that do not cover any description of other concepts (e.g., AQ-11 [181, 179]), and on the other end we have those that, just as Holte's modification of CN2, learns the most specific descriptions possible. In what comes, I will argue that both these extreme approaches are inadequate as they tend to over-generalize and under-generalize respectively. Moreover, I will argue that the ability of controlling the degree of generalization is essential in most real world applications. Therefore, I have developed a novel method of learning characteristic descriptions with this ability which will be presented in the next section.

9.5 A Method for Learning Characteristic Descriptions

The ability of detecting instances of unknown categories is relevant not only for autonomous agents but has nevertheless been largely ignored in the machine learning literature. In theoretical contexts this problem is often regarded as being of minor importance, mainly because it is assumed that the problem domains under study are closed, i.e., all relevant information is known in advance. However, in many practical applications it cannot be assumed that every category is represented in the set of training examples, i.e., they are *open domains* [129], and sometimes the cost of a misclassification is too high. What is needed in such situations is the ability to reject instances of categories that the system has not been trained on. For example, consider the decision mechanism in a coin-sorting machine of the kind often used in bank offices. Its task is to sort and count a limited number of different coins (for instance, a particular country's),

[14] At least, in the original versions of these algorithms.

and to reject all other coins. Supposing that this decision mechanism is to be learned, it is for practical reasons impossible to train the learning system on every possible kind of coin, genuine or faked. Rather, it is desired that the system should be trained only on the kinds of coins it is supposed to accept. Another example are decision support systems, for instance in medical diagnosis, where the cost of a misclassification often is very high — it is better to remain silent than to give an incorrect diagnosis.

Two methods of learning characteristic descriptions will be presented and applied to the ID3 algorithm for learning characteristic decision trees. The first method is a straightforward adaption of the idea of maximum specific descriptions as described above to the decision tree domain. The second method is a novel approach (first presented in paper VIII) that makes use of the information about the statistical distribution of the feature values that can be extracted from the training examples. To show the generality of the new approach, it is applied also to IB1.

9.5.1 A Maximum Specific Description Approach

Both methods are based on the idea of augmenting each leaf of the tree resulting from a decision tree algorithm with a subtree. The purpose of these subtrees is to impose further restrictions on the feature values. A lower and a upper limit are computed for every feature. These will serve as tests: if the feature value of the instance to be classified is below the lower limit or above the upper limit for one or more of the features, the instance will be rejected, i.e., regarded as belonging to a novel class, otherwise it will be classified according to the original decision tree. Thus, when a new instance is to be classified, the decision tree is first applied as usual, and then, when a leaf would have been reached, every feature of the instance is checked to see if it belongs to the interval defined by the lower and the upper limit. If all features of the new instance are inside their interval the classification is still valid, otherwise the instance will be rejected. In what follows, the multidimensional space formed by these intervals will be referred to as the *acceptance region*.

In the first method we compute the minimum and maximum feature value from the training instances of the leaf and let these be the lower and upper limits respectively. This approach will yield an acceptance region corresponding to the maximum specific description (cf. the modification of CN2 by Holte et al. [123]).

While being intuitive and straight-forward, this method is also rather static in the sense that there is no way of controlling the size of the acceptance regions, i.e., the degree of generalization. Such an ability would be desirable, for example, when some instances that would have been correctly classified by the original decision tree are rejected by the augmented tree (which happens if any of its feature values is on the wrong side of a limit). Actually, there is a trade-off between the number of failures of this kind and the number of misclassified instances. How it should be balanced is, of course, dependent of the application (i.e., the costs of misclassification and rejection). Since it is impossible in the above method to balance this trade-off, a more dynamic method in which it can be controlled has been developed.

9.5.2 A Novel Approach Based on Statistical Distribution

The central idea of this novel method is to make use of statistical information concerning the distribution of the feature values of the instances in a leaf. For every feature we compute the lower and the upper limits so that the probability that a particular feature value (of an instance belonging to this leaf) belongs to the interval between these limits is 1–α. In this way we can control the degree of generalization and, consequently, the above mentioned trade-off by choosing an appropriate α-value. The lesser the α-value is, the more misclassified and less rejected instances. Thus, if it is important not to misclassify instances and a high number of rejected (not classified) instances are acceptable, a high α-value should be selected.

It turns out that only very simple statistical methods are needed to compute such limits. Assuming that X is a normally distributed stochastic variable, we have that:

$$P(m - \lambda_{\frac{\alpha}{2}}\sigma < x < m + \lambda_{\frac{\alpha}{2}}\sigma) = 1 - \alpha$$

where m is the mean, σ is the standard deviation, and λ is a critical value depending on α (for instance $\lambda_{0.025}$ = 1.960). Thus, we have, for instance, that the probability of an observation being larger than m–1.96σ and smaller than m+1.96σ is 95%.[15]

In order to follow this line of argument we have to assume that the feature values of each category (or each leaf if it is a disjunctive concept) are normally distributed. This assumption seems not too strong for most applications. However, as we cannot assume that the actual values of m and σ are known, they have to be estimated. A simple way of doing this is to compute the mean and the standard deviation of the training instances ($x_1, ..., x_n$) belonging to the leaf:

$$m^* = \frac{\Sigma x_i}{n}, \quad \sigma^* = \sqrt{\frac{\Sigma(x_i - \overline{x})^2}{n-1}}$$

To get a nice interpretation of the interval between the upper and lower limit, we have to assume that these estimates are equal to the actual values of m and σ. This is, of course, too optimistic, but it seems reasonable to believe (as will be shown in Section 9.5.4) that the method is of practical value also without this interpretation. Anyway, the intended statistical interpretation suggests that the probability of a feature of an instance of a category belonging to the (one-dimensional) acceptance region defined by the lower and the upper limit for α = 0.01 is 99%.

[15]This type of intervals should not be confused with the confidence intervals used for interval estimation of parameters.

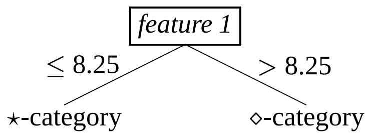


Figure 9.2: The decision tree induced by the ID3 algorithm.

It can be argued that the point estimation of the limits is a rather crude way of computing the acceptance regions. A more elaborate approach would be to compute confidence intervals for each limit and use these instead. This was actually the initial idea but it turned out that this only complicates the algorithm and does not increase the classification performance significantly.

9.5.3 A Simple Example

In this section a very simple example is presented to illustrate the suggested methods and compare them with the original ID3 algorithm. All instances are described by two numerical features, and the training instances belong to either of two categories: the $\star$-category or the $\diamond$-category. The system is given four training instances of each category.

The feature values of the training instances of the $\star$-category are: (3.0, 1.2), (3.5, 0.9), (4.5, 1.1), (5.0, 0.8) and the $\diamond$-category training instances are: (11.5, 1.8), (12.5, 1.5), (12.5, 1.8), (13.5, 2.1). If these training-instances are given to the ID3 algorithm, the output will be the decision tree shown in Figure 9.2.[16] This tree represents the decision rule: if *feature 1* $\leq$ 8.25 then the instance belongs to the $\star$-category, else it belongs to the $\diamond$-category. The classification boundary that follows from this rule is illustrated in Figure 9.3 by a vertical dashed line. If we now try to classify an instance of another category ($\bowtie$) with the feature values (9.0,0.5), it will be (mis)classified as an instance of the $\diamond$-category.

Let us apply the method based on the maximum specific description to the same training set. The result will be the augmented decision tree in Figure 9.4 which will reject all instances outside the dotted boxes in Figure 9.5. A classification of the $\bowtie$-category instance will proceed as follows: we first use the decision tree as before resulting in a preliminary classification which, still as before, suggests that it belongs to

[16] Or a similar one, depending on the cut-point selection strategy. In all examples presented here the cut-point is chosen by first sorting all values of the training instances belonging to the current node. The cut-point is then defined as the average of two consecutive values of the sorted list if they belong to instances of different classes.

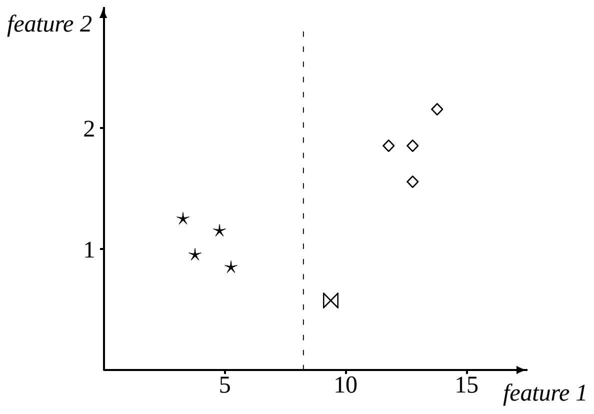


Figure 9.3: The feature space of the example. The boundary between the two categories ($\star$ and $\diamond$) induced by the ID3 algorithm is represented by the vertical dashed line.

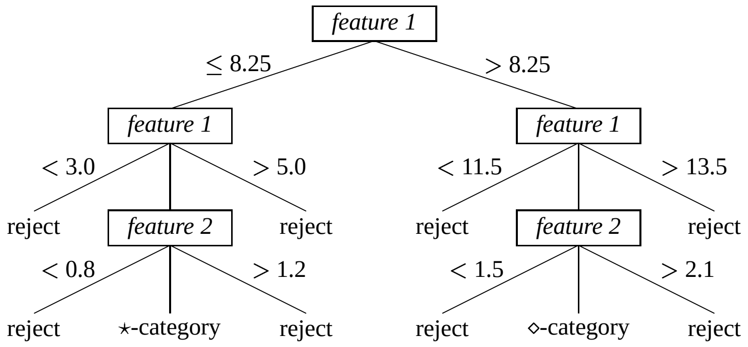


Figure 9.4: The decision tree induced by the ID3-Max algorithm.

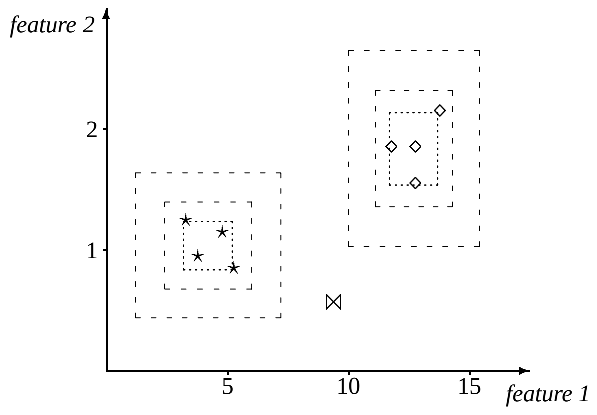


Figure 9.5: The acceptance regions of the maximum specific description (dotted boxes) and those resulting from the method based on statistical distribution with α = 0.05 (inner dashed boxes), and with α = 0.001 (outer dashed boxes).

the $\diamond$-category. However, as we proceed further down the tree into the appended subtree, we will eventually encounter a test that brings us to a reject-leaf, i.e., we check whether the new instance is inside the acceptance region (the dotted box), and find out that it is not. As a consequence, the instance is rejected and treated as an instance of a novel, or unknown, category.

If we apply the method based on statistical distribution with α = 0.05, the lower and upper limits will be as follows:

	feature 1	*feature 2*
$\star$-category	2.2 .. 5.8	0.6 .. 1.4
$\diamond$-category	10.9 .. 14.1	1.3 .. 2.3

These limits will yield a decision tree similar to that of the maximum specific method but with different values on the rejection branches, and will cover the acceptance region marked by the inner dashed boxes in Figure 9.5. In this case, a region can be interpreted as meaning that, if the assumptions mentioned above were correct and if the features are independent, 90.2% (0.95$\times$0.95) of the instances of the category will belong to the region. Just as with the maximum specific tree this tree will reject the $\bowtie$-category instance. We can also see that the lesser α-value that is chosen, the more will the algorithm generalize. The outer dashed boxes correspond to the acceptance regions for α = 0.001, i.e., the probability is 99.8% that an instance of the category is inside the region.

9.5.4 Empirical Evaluation

As we are interested in the behavior of the algorithms when confronted with unknown categories, not all of the categories present in the data sets were used in the training phase. This approach may at first sight seem somewhat strange as we actually know that there are, for instance, three categories of Irises in the Iris database. But, how can we be sure that there only exist three categories? It might exist some not yet discovered species of Iris. In fact, I believe that in most real world applications it is not reasonable to assume that all relevant categories are known in advance and can be represented in the training set.

To keep the presentation concise, only the results of one choice of categories in the training set will be presented here. The results of the remaining combinations can be found in Appendix A. Moreover, in these experiments I have used a basic ID3 algorithm for computing the initial decision tree.[17]

The Iris Database

The classic Iris database [10] contains 3 categories of 50 instances each, where a category refers to a type of Iris plant (Setosa, Versicolor or Virginica). All of the 4 attributes (sepal length, sepal width, petal length, and petal width) are numerical. In each experiment the data set was randomly divided in half, with one set used for training and the other for testing. Thus, 50 (2$\times$25) instances were used for training and 75 (3$\times$25) for testing. Each experiment was performed with the basic ID3 algorithm, the maximum specific tree algorithm (ID3-Max), and the algorithm based on statistical distribution (ID3-SD) for the α-values: 0.2, 0.1, 0.05, and 0.01.

Table 9.1 shows the classification results when the algorithms were trained on instances of Iris Setosa and Iris Virginica. ID3 misclassifies, of course, all the instances of Iris Versicolor, but more interesting is that ID3-SD (for α = 0.1) performs significantly better than the ID3-Max algorithm. It has a slightly higher rejection-rate, but misclassifies over 60% less instances than ID3-Max. We also notice that by varying the α-value it is possible to control the trade-off between the number of rejected and misclassified instances. It is possible to achieve almost zero misclassifications if we choose α = 0.2, but then we get a rejection rate of over 50% also for the two known categories. In fact, also the number of misclassifications of known categories is reduced by the algorithms learning characteristic descriptions. The decision trees induced by ID3 misclassifies

[17] All ID3-based algorithms have been implemented in Common Lisp on a Sun SparcStation running Solaris 2.3.

	Iris Setosa			Iris Versicolor			Iris Virginica		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
ID3	98.8	1.2	0.0	0.0	100.0	0.0	99.2	0.8	0.0
ID3-Max	68.8	0.0	31.2	0.0	14.8	85.2	74.0	0.0	26.0
ID3-SD 0.2	42.4	0.0	57.6	0.0	1.2	98.8	49.6	0.0	50.4
ID3-SD 0.1	62.4	0.0	37.6	0.0	5.6	94.4	70.4	0.0	29.6
ID3-SD 0.05	74.0	0.0	26.0	0.0	17.6	82.4	80.0	0.0	20.0
ID3-SD 0.01	84.0	0.0	16.0	0.0	47.2	52.8	91.2	0.0	8.8

Table 9.1: Results from training set containing instances of Iris Setosa and Iris Virginica (averages in percentages over 10 runs).

1.2% of the Iris Setosa and 0.8% of the Iris Virginica instances, whereas both ID3-Max and ID3-SD induce trees that do not misclassify any of these instances. The main reason for this seems to be that the characteristic decision trees check all features so that they do not take unreasonable values, whereas the discriminative trees only check one or two of the features.

Wine Recognition

This data set comes from the Institute of Pharmaceutical and Food Analysis and Technologies in Genoa, Italy.[18] It contains results of chemical analyses of wines grown in the same region of Italy, but are fermented using three different kinds of yeast. In the analyses the quantities of 13 different constituents were measured. The data set consists of 59 instances of wine of type 1, 71 of type 2, and 48 of type 3. This is a more difficult problem than the above since we here are dealing with many, potentially irrelevant, features. In each experiment 50 (2×25) instances were used for training and 60 (3×20) instances for testing. Each experiment was performed with the original ID3 algorithm, ID3-Max, and ID3-SD for the α-values: 0.1, 0.05, 0.01, 0.001 and 0.0001. Table 9.2 shows the classification results of these experiments.

We can see that for the right α-value (between 0.05 and 0.001 depending on the constraints of the application) the ID3-SD algorithm performs significantly better than the ID3-Max algorithm. ID3-Max has greater problems than ID3-SD when the number of features grows since each additional feature decreases the probability that every fea-

[18] It is available at the UCI Repository of Machine Learning databases (`ftp.ics.uci.edu`) by anonymous ftp in directory `/pub/machine-learning-databases`.

	Wine 1			Wine 2			Wine 3		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
ID3	93.0	7.0	0.0	90.0	10.0	0.0	0.0	100.0	0.0
ID3-Max	31.0	0.0	69.0	27.5	0.0	72.5	0.0	1.0	99.0
ID3-SD 0.1	19.0	0.0	81.0	22.5	0.0	77.5	0.0	0.0	100.0
ID3-SD 0.05	42.0	0.0	58.0	37.5	0.0	62.5	0.0	0.0	100.0
ID3-SD 0.01	67.5	3.0	29.5	63.5	0.0	36.5	0.0	1.5	98.5
ID3-SD 0.001	77.5	6.0	16.5	75.5	1.5	23.0	0.0	8.0	92.0
ID3-SD 0.0001	85.5	7.0	7.5	80.0	2.5	17.5	0.0	12.0	88.0

Table 9.2: Result from training set containing wine of type 1 and 2 (averages in percentages over 10 runs).

ture value of an instance is between the limits. In other words, the number of training instances needed by ID3-Max increases when the number of features increases.

At first sight, the classification performance of ID3-SD seems quite poor. Why would anyone want an algorithm that classifies less than 40% of unseen instances correctly? (As is the case for $\alpha = 0.05$.) Upon further reflection, however, it becomes clear that there are many applications where this behavior is fully acceptable. Take stock market prediction for example; if we were able to classify 10% of the stocks correctly into the categories “buy” and “sell” (and reject all others) we would be very pleased! Thus, being sure not to misclassify any instances is very valuable in some applications.

Coin Classification

This task corresponds to the problem of learning the decision mechanism in coin sorting machines described in the introduction. A preliminary study of this problem is described in a Master’s thesis by Mårtensson [167]. He tested a neural network approach, a statistical method based on a Bayesian classifier, and a decision tree induction method (ID3-Max) on the problem. Although these methods had approximately the same classification accuracy (on known types of coins), he concluded that ID3-Max was the most appropriate method for this application, mainly because it learns explicit rules and is fast both in the learning and in the classification phase.

Although the ID3-Max algorithm showed promising results, an improvement was desired for mainly two reasons: (i) robustness, e.g., an extremely low (or high) value of one parameter of one coin in the training set could ruin the whole decision mecha-

	Hong Kong Coins			Foreign Coins		
	correct	miss	reject	correct	miss	reject
ID3	98.3	1.7	0.0	0.0	100.0	0.0
ID3-Max	79.7	0.0	20.3	0.0	0.0	100.0
ID3-SD 0.05	74.8	0.0	25.2	0.0	0.0	100.0
ID3-SD 0.01	88.9	0.0	11.1	0.0	0.0	100.0
ID3-SD 0.001	95.1	0.3	4.6	0.0	0.0	100.0
ID3-SD 0.0001	96.3	0.5	3.2	0.0	0.0	100.0

Table 9.3: Results from training set containing Hong Kong coins (averages in percentages over 10 runs).

nism, and (2) dynamics (i.e., the ability to control the degree of generalization), e.g., too many coins were rejected as the ID3-Max algorithm did not generalize sufficiently. Both these reasons suggest that ID3-SD would be a promising candidate for this problem. This application is presented in detail in Paper X.

In the experiments two databases were used, one describing Canadian coins contains 7 categories (1, 5, 10, 25, 50 cent, 1 and 2 dollar), and one describing Hong Kong coins that also contains 7 categories (5, 10, 20, 50 cent, 1, 2, and 5 dollar). All of the 5 attributes (diameter, thickness, conductivity1, conductivity2, and permeability) are numerical. The Canada and Hong Kong databases were chosen because when using the manufacturer's current method for creating the rules of the decision mechanism (which is manual to a large extent), these coins have been causing problems. In each experiment 140 (7×20) instances were randomly chosen for training and 700 ($2\times7\times50$) for testing. This scenario is quite similar to the actual situation where you in the training phase expose the system only to the coins of one country, but in the classification phase also confront it with coins of other countries.

Table 9.3 shows the classification results when training on the Hong Kong coin database (the most difficult case). To begin with, we can see that all foreign coins (i.e., the Canadian coins) are rejected, except of course for the ID3 algorithm. However, there were some problems with misclassifications. In this particular application there are some demands that must be met by the learning system before it can be used in reality, namely, less than 5% rejects of known types of coins and very few misclassifications (not more than 0.5%). In our experiment, these requirements are met only by the ID3-SD algorithm with α = 0.001 and 0.0001, which illustrates the advantage of being able to control the degree of generalization.

9.5.5 On the Generality of the SD approach

The main limitation of the SD-method seems to be that it is only applicable to numerical properties. The maximum description method, on the other hand, requires only that the features can be ordered. Thus, one way of making the former method more general is to combine it with the latter method to form a hybrid approach that is able to handle all kinds of ordered features. We would then use the statistical method for numerical properties and the maximum description method for the rest of the properties. Moreover, nominal (and to some extent structural) properties could be handled by accepting those values present among the instances of the leaf and reject those that are not. In this way we get a method that learns characteristic descriptions using all kinds of properties. However, the degree of generalization can only be controlled for numeric features.

The SD approach for creating characteristic descriptions is a general method in the sense that we can take the output from any decision tree induction algorithm, compute a subtree for every leaf, and append them to their leaf. In fact, the approach can, in principle, be applied to any empirical learning method. The procedure for augmenting an arbitrary empirical learning algorithm X is as follows: apply X to the training set as usual, then compute the limits for every category (i.e., cluster) in the training set as described earlier. When a new instance is to be classified, first apply the classification mechanism associated to X as usual, then check that all features values of the new instance are larger than the lower limit and smaller than the upper limit. Thus, it is not necessary to represent the limits in the form of decision trees, the main point is that there should be a method for comparing the feature values of the instance to be classified with the limits.

IB1-SD

To illustrate the generality of the SD approach some experiments with its application to a nearest neighbor algorithm very similar to IB1 [5] will be described.[19] The distance between two instances was defined as the Euclidean distance (where each feature was normalized so that the lowest value of that feature in the training set corresponded to 0 and the highest to 1).

Table 9.4 shows the results when applied to the Iris database (compare to Table 9.1). Not surprisingly IB1-SD and ID3-SD provide very similar results. The reason is that both algorithms only form one cluster/leaf for each category; IB1 does it by definition and ID3 does it since the feature "petal length" linearly separates Iris Setosa and Iris Virginica. As a result, ID3-SD and IB1-SD will compute exactly the same acceptance regions. Thus, all differences between the results of the IB1- and ID3-based algorithms are due to the randomized selection of training and testing sets. (However, the same 40

[19] All IB1-based algorithms have been implemented in C++ on a Sun SparcStation running Solaris 2.3.

	Iris Setosa			Iris Versicolor			Iris Virginica		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
IB1	100.0	0.0	0.0	0.0	100.0	0.0	100.0	0.0	0.0
IB1-SD 0.2	44.3	0.0	55.7	0.0	2.1	97.9	49.9	0.0	50.1
IB1-SD 0.1	64.0	0.0	36.0	0.0	6.4	93.6	71.7	0.0	28.3
IB1-SD 0.05	75.5	0.0	24.5	0.0	17.0	83.0	80.4	0.0	19.6
IB1-SD 0.01	86.5	0.0	13.5	0.0	46.3	53.7	92.0	0.0	8.0

Table 9.4: Results from training set containing instances of Iris Setosa and Iris Virginica (averages in percentages over 40 runs).

	Wine 1			Wine 2			Wine 3		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
IB1	99.5	0.5	0.0	91.0	9.0	0.0	0.0	100.0	0.0
IB1-SD 0.1	30.0	0.0	70.0	28.3	0.0	71.7	0.0	0.0	100.0
IB1-SD 0.05	55.3	0.0	44.7	44.0	0.3	55.7	0.0	0.5	99.5
IB1-SD 0.01	75.9	0.1	24.0	69.0	2.1	28.9	0.0	8.9	91.1
IB1-SD 0.001	87.6	0.4	12.0	83.7	5.4	10.9	0.0	28.6	71.4
IB1-SD 0.0001	94.8	0.5	4.7	88.2	6.6	5.2	0.0	37.1	62.9

Table 9.5: Result from training set containing wine of type 1 and 2 (averages in percentages over 50 runs).

training/test sets were used for all IB1-based experiments. This is also true for all the ID3-based experiments.)

The wine database, on the other hand, gives rise to some more interesting observations. As can be seen in Table 9.5, IB1 performs significantly better than ID3 for Wine 1. Consequently, IB1-SD exhibits higher classification performance than ID3-SD with regard to this category. Note, however, that IB1-SD misclassifies more and rejects less instances of Wine 2 and 3. The reason for this is that the two categories in the training set cannot be linearly separated implying that ID3 will create a tree where at least one category is represented by more than one leaf. In this example ID3 will typically split the category Wine 2 into two leafs. Figure 9.6 illustrates what often happens when splitting a category in this way. Given the 20 instances in the figure, ID3 will construct the

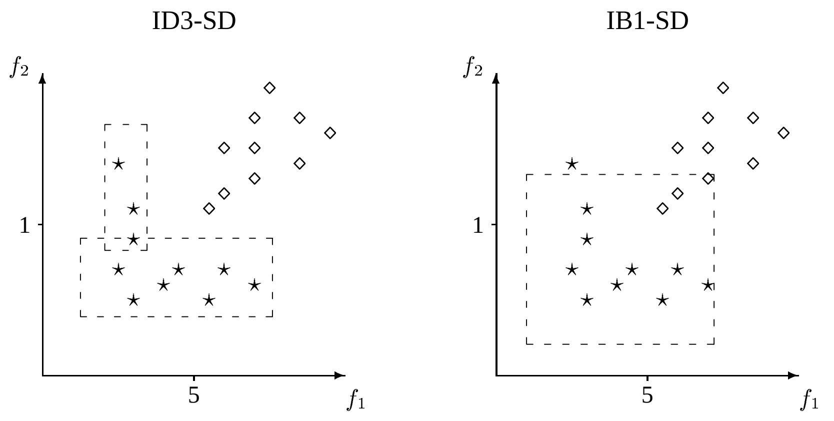


Figure 9.6: The acceptance regions computed by ID3-SD and IB1-SD for the $\star$-category (α = 0.05).

tree in Figure 9.7. ID3-SD will then, as illustrated to the left in Figure 9.6, compute two acceptance regions for the $\star$-category; one for the eight instances with a f_2-value below 1.0 and one for the remaining two instances. IB1-SD, on the other hand, will only compute one acceptance region. Thus, in this case IB1-SD will create larger acceptance regions than ID3-SD which implies that it will classify more instances correctly than ID3-SD (since it will not reject as many instances) but it will also misclassify more instances (of categories not belonging to the training set). In addition, the division into two acceptance regions done by ID3-SD makes the regions match the distribution of the training instances closer.

However, the situation gets even worse for IB1-SD when the instances of a category corresponds to two or more separate clusters in the feature space (cf. disjunctive concepts). The SD approach will work better for algorithms such as ID3 that explicitly separates the clusters, i.e., where it is possible to find out which cluster a particular instance belongs to. In this case, the acceptance regions can be computed separately for each cluster. Otherwise, we are forced to compute only one acceptance region for the whole category which in most cases will be too large (see Figure 9.8).

Also in the Coin domain IB1 performs better than ID3. If we compare Table 9.6 with Table 9.3, we see that IB1 misclassifies one percentage point fewer instances than ID3. In turn, this enables IB1-SD to increase the number of correctly classified instances compared to ID3-SD. More details about the experiments with the IB1-SD algorithm are presented in Appendix A.

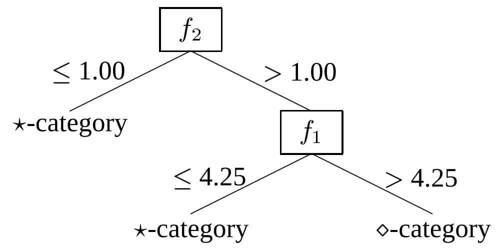


Figure 9.7: The decision tree constructed by ID3 from the instances in Figure 9.6.

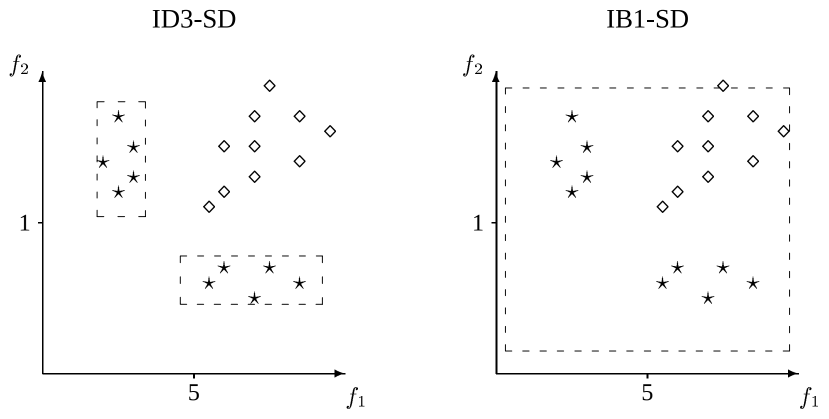


Figure 9.8: The acceptance regions computed by ID3-SD and IB1-SD for the $\star$-category (α = 0.05).

	Hong Kong Coins			Foreign Coins		
	correct	miss	reject	correct	miss	reject
IB1	99.3	0.7	0.0	0.0	100.0	0.0
IB1-SD 0.1	63.8	0.0	36.2	0.0	0.0	100.0
IB1-SD 0.05	76.9	0.0	23.1	0.0	0.0	100.0
IB1-SD 0.01	89.0	0.0	11.0	0.0	0.0	100.0
IB1-SD 0.001	94.2	0.1	5.7	0.0	0.0	100.0
IB1-SD 0.0001	95.7	0.1	4.2	0.0	0.0	100.0
IB1-SD 0.000001	98.2	0.1	1.7	0.0	0.0	100.0

Table 9.6: Result from training set containing Hong Kong coins (averages in percentages over 10 runs).

A Multivariate Version of IB1-SD

Since we are no longer restricted to representing concepts by decision trees, we are not forced to have separate and explicit limits for each feature. As a consequence, we do not have to assume that features are independent. If we are able to capture covariation among two or more features we would be able to create acceptance regions that closer match the distribution of feature values, i.e., regions that are smaller but still cover as many instances. This is illustrated by the example shown in Figure 9.9. The acceptance region for the $\star$-category computed by IB1-SD does not fit the training instances very well. Despite the fact that the members of the two categories form clearly separated clusters, the acceptance region actually covers all training instances of the $\diamond$-category. The acceptance region computed by IB1-SD-multi (an algorithm able to capture covariances between features and that will be described in detail below), on the other hand, do not cover any of the $\diamond$-instances.

To implement IB1-SD-multi we will make use of multivariate methods to compute a weighted distance from the instance to be classified to the “center” of the category/cluster. If this distance is larger than a critical value (dependent of α) the instance is rejected. Assuming that feature values are normally distributed within categories/clusters, we have that the solid ellipsoid of x values satisfying

$$(x-\mu)^T\Sigma^{-1}(x-\mu) \leq \chi_p^2(\alpha)$$

has probability 1 – α, where μ is the mean vector, Σ is the covariance matrix, χ^2 is the chi-square distribution and p is the number features (for more details, see for instance

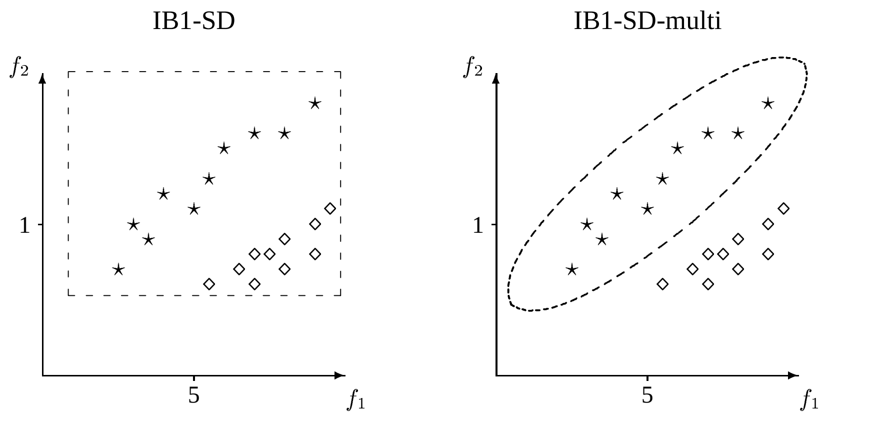


Figure 9.9: The acceptance regions computed by IB1-SD and IB1-SD-multi for the $\star$-category (α = 0.05).

Johnson and Wichern [133]). Thus, we have to estimate two parameters for each category/cluster: (i) μ, which contains the mean for each feature, and (ii) Σ, which represents the covariance for each pair of features. A simple way of estimating these parameters is to compute the observed mean vector and covariance matrix of the training instances $X = [X_1, ..., X_n]$ (a $p \times n$ matrix) belonging to the category/cluster. Thus, we have that:

$$\mu^* = \begin{bmatrix} \frac{\Sigma x_{i1}}{n} \\ \vdots \\ \frac{\Sigma x_{ip}}{n} \end{bmatrix} \qquad \Sigma^* = \frac{Xz^T Xz}{n-1} \quad \text{where} \quad Xz = X - [\mu^*, ..., \mu^*]$$

We can now use this result in the following way (assuming that instances are described by two features): let x_k = $[x_{k1}\ x_{k2}]^T$ be the instance to accept or reject and let α = 0.05. Since χ_2^2(0.05) = 5.99 we accept the instance if

$$(x_k - \mu^*)^T \Sigma^{*\,-1} (x_k - \mu^*) \leq 5.99$$

otherwise we reject it.

Table 9.7 shows some encouraging results when applying IB1-SD-multi to the Iris database. Compared to IB1-SD (for any α-value) we see that by selecting an appropriate α-value it is always possible to achieve both more correct classifications and less

	Iris Setosa			Iris Versicolor			Iris Virginica		
	corr	miss	reject	corr	miss	reject	corr	miss	reject
IB1-SD-multi 0.5	48.5	0.0	51.5	0.0	1.5	98.5	43.8	0.0	56.2
IB1-SD-multi 0.3	61.1	0.0	38.9	0.0	2.9	97.1	64.3	0.0	35.7
IB1-SD-multi 0.2	68.2	0.0	31.8	0.0	4.0	96.0	71.6	0.0	28.4
IB1-SD-multi 0.1	76.4	0.0	23.6	0.0	8.4	91.6	80.2	0.0	19.8
IB1-SD-multi 0.05	81.3	0.0	18.7	0.0	14.1	85.9	85.4	0.0	14.6
IB1-SD-multi 0.01	89.1	0.0	10.9	0.0	31.7	68.3	93.6	0.0	6.4

Table 9.7: Training on Setosa and Virginica (averages over 40 runs).

	Wine 1			Wine 2			Wine 3		
	corr	miss	reject	corr	miss	reject	corr	miss	reject
IB1-SD-multi 0.1	31.0	0.0	69.0	30.1	0.0	69.5	0.0	0.0	100.0
IB1-SD-multi 0.01	53.5	0.0	46.5	48.2	1.3	50.5	0.0	0.1	99.9
IB1-SD-multi 0.001	68.4	0.0	31.6	59.4	1.7	38.9	0.0	1.2	98.8
IB1-SD-multi 10^{-4}	77.3	0.3	22.4	66.4	2.1	31.5	0.0	3.0	97.0
IB1-SD-multi 10^{-6}	87.8	0.4	11.8	74.3	2.9	22.8	0.0	7.0	93.0
IB1-SD-multi 10^{-8}	92.2	0.4	7.4	80.2	3.4	16.4	0.0	11.9	88.1

Table 9.8: Training on wine of type 1 and 2 (averages over 50 runs).

misclassifications using the multivariate approach. (Compare, for instance, IB1-SD 0.05 with IB1-SD-multi 0.1.) Note, however, that the α-value of IB1-SD-multi regards the multivariate distribution whereas the α-value of IB1-SD regards only one variable. For example, as there are four features in the Iris database IB1-SD 0.05 should actually be compared with IB1-SD-multi 0.185 ($1 - (1 - 0.05)^4 = 0.18549$).

Also the experiments on the other two databases suggest that IB1-SD-multi is a promising approach. The results of the Wine classification summarized in Table 9.7.

As can be seen in Table 9.9, the method performs very good in the coin classification domain. In fact, it does not misclassify any instances! Note, however, that very small α-values must be used to achieve excellent classification behavior.

	Hong Kong Coins			Foreign Coins		
	correct	miss	reject	correct	miss	reject
IB1-SD-multi 0.1	74.3	0.0	25.7	0.0	0.0	100.0
IB1-SD-multi 0.01	87.4	0.0	12.6	0.0	0.0	100.0
IB1-SD-multi 0.001	92.1	0.0	7.9	0.0	0.0	100.0
IB1-SD-multi 10^{-4}	94.8	0.0	5.2	0.0	0.0	100.0
IB1-SD-multi 10^{-8}	98.0	0.0	2.0	0.0	0.0	100.0
IB1-SD-multi 10^{-12}	98.7	0.0	1.3	0.0	0.0	100.0

Table 9.9: Training on Hong Kong coins (averages in percentages over 10 runs).

9.5.6 Discussion of the SD approach

The rationale behind the methods presented in this section was to combine the advantages of characteristic descriptions with with those of discriminative descriptions. Of the methods presented, the maximum specific description method (Max) seems to work well in some domains, but often the methods based on statistical distribution (SD) give significantly better results. The main reasons for this seem to be that they are more robust than the former and that they make it possible to control the degree of generalization, which leads to another advantage of the statistical approach, namely, that the trade-off between the number of rejections and misclassifications can be balanced in accordance with the constraints of the application. In some applications the cost of a misclassification is very high and rejections are desirable in uncertain cases, whereas in others the number of rejected instances are to be kept low and a small number of misclassifications are accepted. Figure 9.10 provides a graphical illustration of the classification performance of ID3-SD that shows how this trade-off can be balanced by choosing an appropriate α-value. Graphical illustrations of all experiments are provided in Appendix A.

We should note that in all the experiments described above there is a discrepancy between the theoretical implications of the α-value and the actual results. For instance, the acceptance region for α = 0.01 does not accept 99% of the instances of the category (in the example illustrated in Figure 9.10, the average is only 65.5%). There are some obvious reasons for this. The two most important are: (i) the estimations of m, σ, μ, and Σ are not sufficiently close to the true values, and (ii) the feature values may, in fact, not be normally distributed. However, since the method seems to work so well in the practical applications, it is my opinion that this “anomaly” is of minor importance.

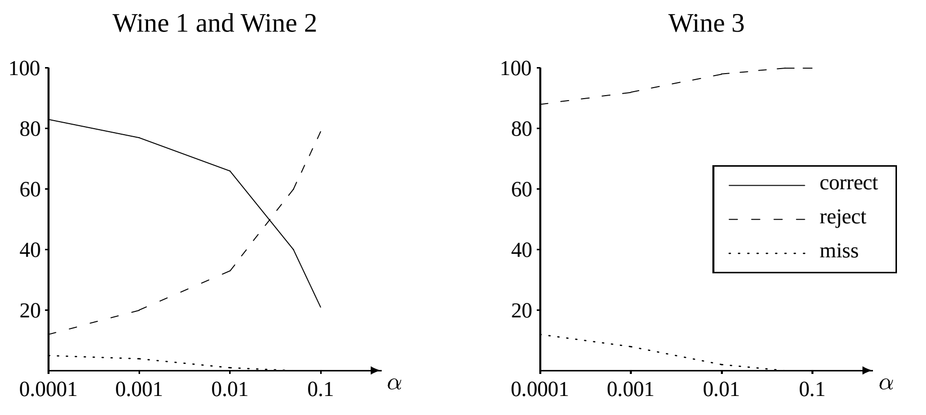


Figure 9.10: Classification performance of ID3-SD as a function of α. Performance on categories present in the training set are shown in the left diagram and categories not present in the training set in the diagram to the right. (Cf. Table 9.2.)

The expansion of an algorithm X to X-SD is carried out using only simple statistical methods. If n is the number of training instances (of a category/cluster) and m is the number of features, the algorithmic complexity of the computations associated with the acceptance regions is linear in the product of these, i.e. $O(nm)$, in the learning phase (which can be neglected when compared to the cost of computing the original decision tree), and linear in m, i.e. $O(m)$, in the classification phase. The algorithmic complexity of the SD-multi algorithm is somewhat higher; $O(nm^2)$ in the learning phase and $O(m^2)$ in the classification phase. However, as the number of features, m, typically is very small (less than 10), also the SD-multi algorithm is tractable for most practical applications.

Some Potential Limitations

As we observed in section 9.5.4, some instances that would have been correctly classified by the decision tree are rejected by the augmented tree, i.e., if any of its feature values is outside their interval. This is related to the trade-off between the number of rejections and misclassifications that can be controlled by selecting a proper α-value. Development of methods to automatically determine the appropriate degree of generalization belongs, however, to future research.

The original ID3-algorithm (in contrast to IB1) is quite good at handling the problem of irrelevant features, i.e., only features that are useful for discriminating between

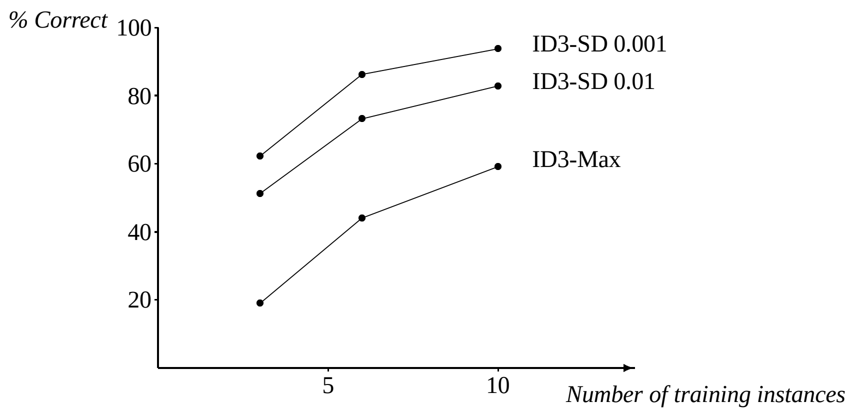


Figure 9.11: The percentage of correctly classified instances of known categories as a function of the number of instances of each category in small training sets (averages over 10 runs). The remaining instances were rejected.

the categories in the training set are selected. But since the suggested methods compute upper and lower limits for every feature and use these in the classification phase, also the irrelevant features will be subject for consideration. However, this potential problem will typically disappear when using the statistically based method for the following reason. An irrelevant feature is often defined as a feature which value is randomly selected according to a uniform distribution on the feature's value range (cf. Aha [4]). That is, the feature values have a large standard deviation, which will lead to a large gap between the lower and the upper limit. Thus, as the acceptance region will be very large with regard to this feature, the feature will still be irrelevant for the classification.

Another potential problem for ID3-SD is the problem of few training instances. One would think that when the number of training examples of a category decreases there is a risk that the estimates of the mean values and the standard deviations, or mean vector and covariance matrix (SD-multi), will not be sufficiently good. However, preliminary experiments in the coin classification domain indicates that the classification performance decreases only slowly when the training examples get fewer. As can be seen in figure 9.11, it handles the problem of few training instances better than the maximum specific description which, in fact, has been suggested as a solution to the related problem of small disjuncts (cf. Holte et al. [123]).

Finally, when using the SD approach (not SD-multi) and the number of features is large there is one thing that we should pay attention to. For example, if we choose α

= 0.01 and have 20 features, the probability that an instance is inside the acceptance region is just 81.8%[20] resulting in too many undesired rejected instances. However, a simple solution to this problem is to determine the α-value out of a desired total probability, P_{tot} (we have that $(1-\alpha)^n = P_{tot}$). For example, if there are 20 features and we want a total probability of 95%, we should choose α = 0.0025.

Noisy Data

Noisy data is a problem that has not been addressed in this chapter. Noise is an inescapable problem in most real-world applications and has been addressed by some learning-from-examples systems. The solutions are often based on the assumption that the members of a category are similar. An implication of this assumption is that descriptions of instances that differ significantly from descriptions of other members of the category should be regarded as noisy.[21]

Algorithms based on the SD approach are better at handling noisy data than algorithms based on maximum specific descriptions in the sense that an extreme feature value for one (or a few) instance(s) will not influence the positions of the limits of that feature in a SD-algorithm as much as it will in Max-algorithm. This is illustrated in the left part of Figure 9.12 where a single instance with a single noisy feature value corrupts the acceptance region of the Max-algorithm whereas the acceptance region of the SD-algorithm is affected to a lesser extent.

A method for further reducing the problem of noisy instances, would be to use the acceptance regions to remove instances that are (far) outside their acceptance region and then recalculate the region. For instance, if we remove the noisy instance in the figure and recalculate the acceptance region, we get the region shown in the right picture in the figure. However, this method for *outlier detection* has not yet been evaluated.

In this section we have based the SD-algorithms on algorithms that are not very good at handling noisy data in the first place (ID3 and IB1). Thus, there is another trivial solution to the problem with noisy data: use any noise tolerant algorithm. For inducing decision trees, take for instance Quinlan's C4.5 algorithm [218] and then compute the subtrees as before for the remaining leaves.

Some Final Remarks

At first glance, it now seems appropriate to ask the following questions: Why bother building a decision tree in the first place? Could we not just compute the lower and the

[20] Assuming that the estimations of the means and standard variations are the correct values.

[21] A problem with this assumption is that it is not compatible with the existence of atypical instances, in the sense that it becomes impossible to discriminate noise-laden instances from atypical ones (cf. the problem of small disjuncts [123]).

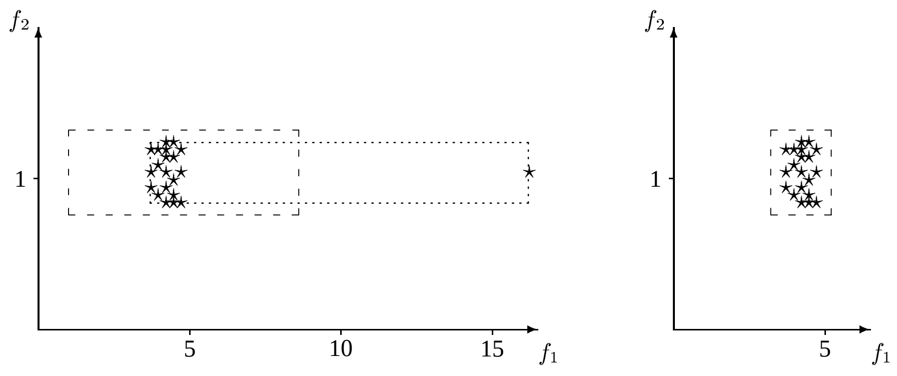


Figure 9.12: To the left: a category with twenty good and one noisy instance. The acceptance region computed by an algorithm based on the maximum specific description corresponds to the dotted box. The dashed box correspond to the acceptance region computed by an SD-algorithm with α = 0.1. To the right: the noisy instance has been removed and the acceptance region recomputed using the SD-algorithm

upper limits for every category and test unknown instances against these? The main problem with such an approach would be that when a new instance is to be classified, it might be assigned to two or more classes. The reason for this ambiguity is, of course, that the acceptance regions of two or more categories may overlap. Moreover, when dealing with disjunctive concepts, for the reason above, this would not be a good approach. In this case, we must have an algorithm that is able to find suitable disjuncts of the concept, a task that algorithms like ID3 normally are quite good at.[22]

Future work will further evaluate to what extent different empirical learning methods can be improved using the SD approach. In this perspective, we have here only described an application of the general method to the ID3 and IB1 algorithms. This is also the main reason why we have not compared the SD algorithms to other kinds of algorithms that learn characteristic descriptions.

To sum up, the SD approach divides the learning into two separate phases:

1. discrimination between all known categories
2. characterization of the known categories (against all possible categories).

[22]However, Van de Merckt [62] has suggested that for numerical attributes a similarity-based selection measure is more appropriate for finding the correct disjuncts than the original entropy-based measure that has been used in the empirical evaluations presented here.

(The classification is divided in a similar way.) It is the author's opinion that there is a tension between discrimination and characterization. However, whether these tasks really are "orthogonal" (in that they must be performed in two separate steps), or whether it is possible to optimize both of them in a one-step algorithm, is an interesting question that requires further deliberation.

9.6 Integrating Different Learning Strategies

Although the above mentioned requirements (incrementality, fastness, characteristicy, etc.) are very important, it is the requirement that several methods of learning must be applied simultaneously that indicates where the greatest need for more research can be found. It may be true that there already exist systems that integrate two or more learning methods. However, most of these systems integrate learning from examples and explanation-based learning (cf. Lebowitz [154]), or just different algorithms for learning from examples.

In particular, learning a composite category representation as outlined in the last chapter seems to demand a large variety of learning mechanisms. Let us discuss for each of the components how they could be learned by an autonomous agent. While it is clear that the agent does not learn all the components at the same time, the concept must be created by initially learning one, or possibly two, of the components. It seems natural to assume that the agent either learns the name of the category (i.e., the external designator) or the epistemological component first. Which of these is actually learned first by humans has been, and to a large extent still is, a controversial question within the field of *developmental psychology*. In most early research it was hypothesized that linguistic input was the fundamental source of information in the concept formation process (i.e., first the words are learned, then the other components). In contrast, Nelson [200] has argued that concepts originate from interaction with the physical world. Only after this, the words that fit to these concepts are learned. A more neutral position is taken by Bowerman [35] who suggests that: "there appears to be a complex interaction in word acquisition between children's own predispositions to categorize things in certain ways and their attention to the words of adult language as guides to concept formation." This view is in line with the approach that will be described below, which allows either of the components to be learned first. In the same article, Bowerman also argues that children's concept formation processes are quite similar to adults'. This is one of the reasons that there has not been put much weight on developmental psychology in this thesis.

Let us, however, begin with the epistemological component necessary for recognizing instances of the category. There seem to be two different ways of learning this component: (1) by unsupervised learning (i.e., through experience), or (2) by supervised learning (i.e., by being explicitly taught). The first of these is probably the most basic where the agent through perceptual interaction with the environment autonomously identifies a new category and creates an epistemological representation.

Supervised learning, on the other hand, requires that another agent (i.e., some kind of teacher) already knows the name of the category, knows how to recognize its instances, and is willing to communicate this knowledge. Since this kind of learning also involves another component, the external designator (i.e., the name of the category), there are two possible situations: (2a) the agent already knows the name, or (2b) it does not. If the name is known, the task will be to associate this already learned component with the new epistemological representation and make them parts of the same concept. If the name of the category is not known to the agent, it must learn both the external designator and the epistemological component. Whereas (1) and (2b) correspond to the traditional ML paradigms of learning by observation and learning from examples respectively, (2a) seems not to have received any attention within the ML community.

For learning the external designator we are restricted to supervised learning. Also in this case there are two possibilities: (1) the agent already has an epistemological component (e.g., learned by observation), or (2) it has not. The second case corresponds to (2b) above, whereas the first case bears some resemblance to (2a) in that what is actually to be learned is a connection between the epistemological component and the external designator. Some experiments in this direction have been carried out by Schyns [239]. He has constructed a modular neural network that uses an unsupervised method to form categories (and representations of them) (i.e., a self-organizing feature map) and a supervised method to learn their names (i.e., an auto-associator). A third way of learning the name is by direct implanting of knowledge or learning by being told. Regarding the epistemological representation, however, this approach may not be a very good idea for reasons associated with the grounding of symbols discussed in previous chapters, i.e., the grounding provided by a programmer is probably meaningless for the agent. The internal designator, on the other hand, need not to be learned, since it is the agent itself that decides what the category will be called internally. We here assume that the agent will invent this name when the concept is originally created.

It seems that in order to learn inferential knowledge through experience, the agent must already have an adequate epistemological component. That is, it must be able to recognize instances in order to detect more general rules regarding the category's members or their relations to members of other categories. On the other hand, to learn inferential knowledge at the linguistic level, for instance by being told, it would suffice to have the external designator. Neither of the concept learning algorithms presented above are suitable for learning this kind of knowledge. A potential candidate is the type of algorithms developed under the label of *Inductive Logic Programming* (cf. Muggleton [192]), which learn first-order theories from examples and background knowledge.

Thus, it would be possible to develop at least supervised approaches for learning the inferential component based on such algorithms.

As indicated above, which types of learning that are adequate to integrate also depends heavily on the scenario in which the agent works. As mentioned in Chapter 2, there are two possible scenarios for an autonomous agent. It can either be alone in its environment or be among other agents which it can communicate with. An agent that is alone seems to be limited to unsupervised learning, such as learning by observation. In a situation where other agents exist, there is the possibility of supervised learning, such as learning from examples, in addition to learning by observation and direct implanting of knowledge. Thus, for this kind of agent, an integration of learning from examples and learning from observation, possibly in ways suggested above, would be fruitful.

What about learning from discovery then? The experiments conducted so far have shown that such systems might work in small, well understood, and predictable domains, but that it is very hard to implement such systems able to function in real-world domains. The main reason being that they demand that the domain to discover is fully formalized. Thus, despite the fact that learning by discovery is a very powerful learning method, it seems that (at least at the present stage of research) autonomous agents will have to manage without it.

The algorithms that learn from examples and by observation seem more adequate for learning bottom-up than top-down categories, whereas explanation-based learning algorithms are, more or less, designed to learn top-down categories. Thus, a problem-solving agent may benefit from using techniques similar to those used in EBL. However, as pointed out earlier, EBL systems do not form any new categories. The actual category formation step is when the high-level description is created during problem solving. This is not a very well studied topic, deserving more attention. In addition, it may be the case that a new kind of high-level representation component (instead of the metaphysical) is needed for this task. EBL may then be used to infer a representation from this component that can support the epistemological function. However, as Lebowitz has pointed out [152], it is questionable whether there exist real-world situations where EBL can be applied, i.e., where the agent possesses all the background knowledge required to make the transformation into a low-level description. The problem is that the EBL-algorithm requires that the background knowledge is complete and consistent (in the more general senses of these terms). As pointed out by Honavar [126], an interesting approach to solving this problem would be to treat the background knowledge as though it was non-monotonic.

9.6.1 A Quite General Model of Concept Learning

Based on the discussion above, I will now sketch a general model of concept learning. It integrates supervised and unsupervised learning and is able to learn both the external designator and the epistemological component. The model is summarized in Figure 9.13.

The learner receives observations of three different kinds. In the first case, the input consists only of the name, $\mathcal{N}$, of a category (cf. learning by description). If the learner does not have any concept that has $\mathcal{N}$ as its external designator, then a new concept should be created with $\mathcal{N}$ as its external designator. (If such a concept is already known, there is nothing that can be learned from the observation.)

In the second case, the input consists only of a description, $\mathcal{D}$, of an object (cf. learning by observation). We should then check whether any of the epistemological components indicates that $\mathcal{D}$ probably is a description of an object belonging to an already known category, e.g., is sufficiently similar to the members of some category. If such a component is found, it should be updated using $\mathcal{D}$ (and the observation categorized as belonging to this category), else a new concept should be created and an epistemological representation constructed based on the observation $\mathcal{D}$.

Finally, in the third case where both a name ($\mathcal{N}$) of a category and a description ($\mathcal{D}$) are given (cf. learning from examples) there are two main alternatives: (i) The learner already knows a concept that has $\mathcal{N}$ as its external designator. If this concept has an epistemological representation, the representation should be updated, if it does not, the situation becomes more complicated for the following reason: the learner may have learned one external designator and one epistemological component separately (and, as a consequence, having formed two separate concepts) that, in fact, refers to the same category. Thus, it is necessary to check whether any of the concepts with an epistemological component, but without an external designator, indicates that $\mathcal{D}$ probably is a description of an object belonging to a known category. If such a concept is found, it should be merged with the first concept into a singular concept, else the learner should create a new epistemological representation based on $\mathcal{D}$. In the second alternative, (ii) there is not any concept that has $\mathcal{N}$ as its external designator. Also in this case the learner has to check whether any of the concepts with an epistemological component, but without an external designator, suggests that $\mathcal{D}$ probably is a description of an object belonging to a known category. If such a concept is found, $\mathcal{N}$ should be assigned to the external designator (and the epistemological component updated). If no such concept is found, a new concept should be created, its external designator assigned to $\mathcal{N}$, and its epistemological component constructed based on $\mathcal{D}$.

It should be noted that this learning model, in addition to integrating learning by being told, learning from examples and learning by observation, is incremental, and learns multiple concepts in parallel. The model does not specify what representation scheme should be used for the epistemological component. However, it is required that the learned descriptions are characteristic and satisfies thus all but one of the requirements of an autonomous learning system discussed above. The remaining require-

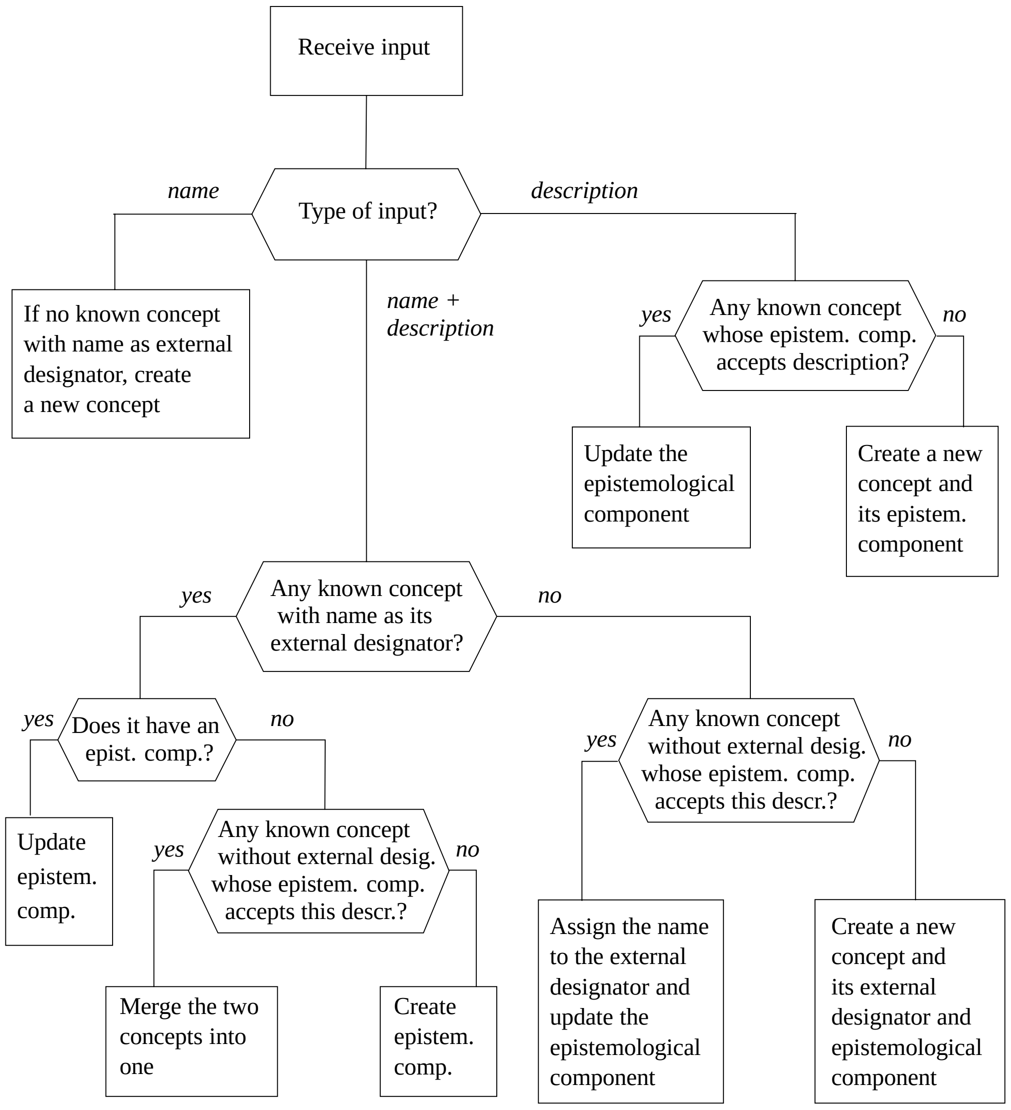

Receive input
Type of input?
name
description
name + description
If no known concept with name as external designator, create a new concept
yes
Any known concept whose epistem. comp. accepts description?
no
Update the epistemological component
Create a new concept and its epistem. component
yes
Any known concept with name as its external designator?
no
yes
Does it have an epist. comp.?
no
Update epistem. comp.
yes
Any known concept without external desig. whose epistem. comp. accepts this descr.?
no
Merge the two concepts into one
Create epistem. comp.
yes
Any known concept without external desig. whose epistem. comp. accepts this descr.?
no
Assign the name to the external designator and update the epistemological component
Create a new concept and its external designator and epistemological component

Figure 9.13: Flow chart describing a model of incremental similarity-based learning of external designators and epistemological components that integrates supervised and unsupervised learning.

ment, to make it learn fast (i.e., needing only a few observations to learn a fairly useful concept), is an implementational issue and will not be discussed here. However, an exemplar/template-based approach seems to be a promising alternative.

9.7 Conclusions

The issue of concept acquisition, in contrast to functional and representational issues, has been more intensively studied in AI than in the Cognitive Sciences. In fact, it is quite common that learning methods developed within AI are adopted by psychologists. This has been acknowledged by Taraban [264] who writes that: “Work in machine learning has produced as a by-product models or components of models that have been useful to cognitive/experimental psychology.” (p. 1) Moreover, we should notice that for all the psychological models presented in this chapter, there exist corresponding AI methods. For instance, one can compare the association theory with backpropagation learning (or any other learning algorithm described on a sufficiently high level of abstraction), the hypothesis testing theory with Gross’ system, and the exemplar strategy with Kibler and Aha’s experiments on instance-based learning. Thus, the most recognized of the existing theories about human concept acquisition have already been tested as AI methods, implying that there is not much we can gain by studying these psychological models. In addition, we presented a novel method for learning characteristic description that can be applied to almost every known concept learning algorithm. A general model for learning concepts (i.e., the epistemological component and external designator) that integrates supervised and unsupervised learning was also suggested.

However, the main result of the study is perhaps, as pointed out several times before, the insight that models of concept learning by autonomous agents are constrained by several demands. Table 9.10 summarizes to what extent some popular learning algorithms meet the requirements made upon an autonomous concept learning system.

In the table “Connell & Brady” refers to their learning vision system based on Winston’s ANALOGY program and “Murase & Nayar” to their approach based on appearance models. AQGMCL-IB1 is a (not yet implemented) version of my (quite) general model of concept learning based on a version of the IB1 algorithm that has been modified to learn characteristic descriptions according to the method described earlier in this chapter. The judgments presented in the table are, of course, only of a very rough nature (e.g., there are several different kinds of structural properties). Nevertheless, they should give an idea about the appropriateness of basing an autonomous concept learning system upon these algorithms.

It is primarily the basic versions of the algorithms that are evaluated. Improved or modified versions are only represented implicitly in the table by “–/+” (e.g., CLUS-

	pro	qual	quan	str	inc	mul	fast	char	sup	unsup
AQ-11	–	+	+	–/+	–	+	+	+	+	–
ART-2	+	+	+	–	+	+	+	+	–	+
ART-MAP	+	+	–/+	–	+	+	+	+	+	–
AUTOCLASS	+	+	+	–	–	+	+	–/+	–/+	+
Backpropagation	+	–/+	+	–	–	+	–	–/+	+	–
CLUSTER/2	–	+	+	–/+	–	+	+	+	–	+
COBWEB	+	+	–/+	–/+	+	+	+	+	–	+
IB1	+	–/+	+	–/+	+	+	+	–/+	+	–
ID3	–	+	–/+	–	–/+	+	+	–/+	+	–
Version Spaces	–	+	–	–	+	–	–	+	+	–
Connell & Brady	–	–	–	+	+	–	+	+	+	–
Murase & Nayar	+	–	–	+	–	–	–	+	+	–
AQGMCL-IB1-SD	+	–/+	+	–/+	+	+	+	+	+	+

Table 9.10: A table showing which requirements some popular learning systems meet. "+" means that the system do satisfy the requirement, "–" that the system does not satisfy the requirement, and "–/+" that the system can be modified to satisfy the requirement without making too much violence on the original idea. The requirements are defined as follows:

pro	not limited to classical definitions (consistent with idea of prototypes)
qual	able to represent qualitative properties
quan	able to represent quantitative properties
str	able to represent structural properties
inc	learns incrementally
mul	learns multiple concepts (i.e., uses knowledge about other concepts)
fast	need just a few instances to learn a reasonable description
char	learns characteristic descriptions
sup	able to perform supervised learning
unsup	able to perform unsupervised learning

TER/2 – CLUSTER/S, ID3 – ID3-SD). In other words, if a system do not satisfy a requirement but can be modified to do so without making too much violence on the original idea, it is marked in this way. In most of the occurrences of "–/+" in the "str"-column, we have assumed that it is possible to define an adequate similarity measure for structural properties.[23] Similarly, most of the occurrences of "–/+" in the "char"-column assumes that the system has been modified according to the SD-approach.

We should note that in some cases these requirements are not independent. For instance, an incremental unsupervised system have to learn characteristic descriptions (otherwise it could not decide when to form a new cluster, or category). Related to this, there is another requirement that one could add: when in the unsupervised mode, the system must by itself decide how many clusters (or categories) to form. In an incremental system this would correspond to one of the processes that Schank et al. argued must be specified in an autonomous concept learning system, namely, deciding when to create a new concept (see page 121). Yet another requirement that one might wish to add is that of being able to learn disjunctive concepts. However, all of the unsupervised systems evaluated in the table, except for the learning vision systems, satisfies this requirement. An unsupervised system can, of course, not realize that two separate clusters belong to the same category.

Note also that the learning computer vision systems do not meet many of the requirements. They are simply too specialized. Dealing with fuzzy perceptual data, however, the reduction of complexity by specialization might be the only way to make the problem manageable.

[23] It should also be noted that if the requirements "qual", "quan", and "str" is to be satisfied by a singular system, we must define a complex similarity measure that handles all the three kinds of properties.

Chapter 10

Conclusions and Further Research

The opening chapters of this thesis were devoted to different aspects of autonomous agents and provided the context for the remaining chapters. The goal of these has been, besides that of reviewing the research within the cognitive sciences and AI on different aspects of concepts, to investigate the issue whether psychological and philosophical theories of concept acquisition can help us in constructing algorithms for concept acquisition by computer-based autonomous agents. However, as is evident from the research reviewed in the previous chapters, it is from more fundamental aspects than acquisition that the influence from the cognitive sciences has the potential of being most fruitful.

In this chapter we will summarize the conclusions from the previous chapters and make some suggestions for further research based on these conclusions.

10.1 Conclusions

Based on the initial general discussion of autonomous agents, a categorization scheme was suggested that was based on the distinctions regarding whether the agents were situated or not, and whether they were embodied or not. It was concluded that when developing new approaches for physical autonomous agents that are supposed to work in real-world domains, it is important to regard the agent as being situated and embodied from start. The deliberative and the reactive approaches were then compared with the result that a hybrid approach is preferable since both high-level deliberative reasoning and low-level reaction on perceptual stimuli seems necessary. A presentation and discussion of several of the most acknowledged suggestions for a hybrid agent architecture confirmed, however, the assertion put forward by Wooldridge and Jennings [289] that these suggestions suffer from a lack of theoretical grounding. In order to improve this situation, a novel framework called anticipatory agents based on the theory of anticipatory systems was suggested in Chapter 3.

Initial results from simulations of a linear quasi-anticipatory autonomous agent architecture (ALQAAA), which correspond to a special case of the general framework of anticipatory agents, were presented. ALQAAA integrates low-level reaction with high-level deliberation by embedding an ordinary reactive system based on situation-action rules, called the Reactor, in an anticipatory agent forming a layered hybrid architecture. By treating all agents in the domain (itself included) as reactive agents, this approach drastically reduces the amount of search needed while at the same time requiring only a small amount of heuristic domain knowledge. Instead it relies on a linear anticipation mechanism, carried out by the Anticipator, to achieve complex behaviors. Results from both single- and multi-agent simulations indicate that the behavior of ALQAAA agents is superior to that of the corresponding reactive agents. Some promising results on cooperation and coordination of teams of agents were also presented. In particular, the linear anticipation mechanism was successfully used for conflict detection.

In the following chapter we discussed world modeling and concluded that, although explicit knowledge about the world is very difficult to acquire because of the necessary signal to symbol transformation, both world and environment models are vital for deliberative and hybrid agents. To bridge the gap between signals and symbols, closer integration of learning and vision systems is necessary. However, the environment model does not need to be at the detailed level often assumed by the proponents of the traditional deliberative view. Rather, it should be at a high level of abstraction, used only to guide the agent's behavior. We also identified concepts as one of the most important primitive entities of which world and environment models are built. Moreover, we concluded that although reactive agents do not have explicit knowledge about the world, they need to have concepts in one form or another.

In Chapter 5, we tried to make explicit what it actually means to have a concept, but suggested that a more appropriate question would be to ask which functions a concept should serve. While some cognitive science literature discusses this topic, it has hardly ever been discussed in the AI literature. In Chapter 6, six important classes of functions of human concepts were identified: stability, cognitive economical, linguistic, epistemological, metaphysical and inferential functions. All of these proved to be desirable also for autonomous agents (with a possible exception for the metaphysical function). In the following chapter, we discussed the nature of categories, the entities that concepts are supposed to represent. AI researchers have, or at least have had, a very simplified view of the nature of categories. An autonomous agent in a real-world environment has to deal with real categories, not artificial ones as most previous AI-systems have done. It is also important to make a distinction between natural and derived categories since they must be acquired in different ways. Natural categories, and natural kinds in particular, are typically formed by observing the external world and grouping similar objects together, whereas derived categories arise during internal problem solving activities. As a consequence, concepts corresponding to natural categories are probably best learned by similarity-based algorithms, whereas derived categories need top-down algorithms. Explanation-based learning is, in a sense, a top-down approach, but does not address the problem of *formation* of concepts. In this chapter we also discussed some aspects of the concept of properties, such as: where do the features used to describe objects actually come from; are they innate (cf. categorical perception) or are they learned? That is, should an autonomous agent's "feature detectors" be preprogrammed or learned? At the present stage of AI research, it is difficult to answer this question univocally. We also investigated what different kinds of properties there are, and concluded that it is relevant to distinguish between at least three kinds: qualitative (nominal), quantitative (numeric), and structured properties. All these was regarded as being of such importance that an agent must be able to deal with all of them. A related, and equally complicated issue, is the problem of how to deal with the concept of similarity. If we want to base the agent's concept formation on similarity, we must define a similarity measure that can handle all of the three kinds of properties.

As for the representation of categories, we concluded that a single and simple representation does not suffice to account for all the functions that we want concepts to serve. Instead, an autonomous agent must have a complex, or composite, category representation structured according to the desired conceptual functions. A suggestion for such a representation scheme was presented in Chapter 8. The suggested conceptual structure has an epistemological component for perceptual (i.e., normal) categorization and an optional metaphysical component for more "scientific" categorization. As we have seen, it seems that some kind of prototype-based representation also able to represent structural knowledge probably will be the best alternative for the epistemological component, whereas a logic-based classical representation seems to be the most appropriate for the metaphysical. To be able to reason and make predictions about the category and its members, the agent needs a large amount of encyclopedic knowledge. This is stored in the inferential component. How this should be represented has not been discussed in detail, but in the light of past (and most current) AI-research some kind of logic-based, possibly probabilistic, representation language seems a natural choice. Finally, to support stability and linguistic functions, the concept structure should also include an internal and an external designator. It was also argued that the problem of symbol grounding becomes easier to resolve if it is viewed in terms of this composite concept framework. In this approach, it is essentially the vision system, through its use of epistemological representations that are parts of the same structure as the corresponding symbols, which facilitates grounding, or the connection between symbols (i.e., internal designators) and their referents (i.e., objects in the world).

Regarding the actual acquisition of the different parts of this structure, it seems that the agent has to rely on learning from examples (if there is some kind of teacher available) and learning by observation. In addition, some method for forming derived (top-down) categories is probably needed. It was also concluded that an autonomous concept learning system must be incremental, reasonably fast, and it must not concern only one concept at a time. However, the most urgent topic for research seems to be to develop systems that integrate different acquisition methods. The most interesting combination is probably learning from examples and learning from observation. A model for this called AQGMCL was sketched in Section 9.6.1 but has not been implemented. Another important requirement is that autonomous learning algorithms should learn characteristic, not discriminative, category representations. This demand disqualifies several popular learning algorithms such as ID3, nearest neighbor, and the backpropagation algorithm. To solve this problem, a general method for learning characteristic representation, the SD method, was presented in Section 9.4.4 and applied to the ID3 and the IB1 algorithms. A number of experiments suggested that this method is superior to previous methods, mainly because of its ability to control the degree of generalization. A multivariate version of this method, SD-multi, was also presented and evaluated. Chapter 8 was concluded with a table summarizing to what extent some of the most popular learning systems meet the requirements made upon an autonomous concept learning system. The only system of these that have the potential of meeting all the requirements was based on AQGMCL using the SD method to learn characteristic descriptions. It was also pointed out that the input to the learner in present AI-systems is usually *descriptions* of instances; consequently they deal with linguistic descriptions of the real world. Thus, the observations are on the linguistic level. Autonomous agents, on the other hand, have to deal with reality itself, making observations on the perceptual level as well. In particular, agents that are alone rely heavily on such observations, whereas communicating agents also make observations on the linguistic level. However, as we have seen there is a growing interest in developing vision systems able to learn category representations. Finally, we noted that concept learning has been limited to the learning of epistemological (and metaphysical) components and to some extent the external designator, ignoring the inferential component.

In short, besides the surveys of autonomous agents and of different aspects of concepts, the following main contributions can be identified: (i) the framework for anticipatory agents, in particular the model (and simulations) of linearly quasi-anticipatory agents, (ii) the composite concept representation scheme, and (iii) the general method for (and thorough empirical evaluation of) learning characteristic concept descriptions.

10.2 Suggestions for Further Research

This thesis will certainly not be the last word in the research on neither anticipatory agents nor the concept of concepts in the context of autonomous agents. There are many examples of interesting topics that are suitable subjects for further research In my opinion, the following are especially worth pursuing:

- Further developing the general framework of anticipatory agents, e.g., introduce different kinds of learning and methods for dealing with anticipation failures.
- Applying ALQAAA to more realistic domains.
- Trying out the framework of composite category representation in realistic settings.
- Investigating whether it is necessary to expand this framework, which is now limited to the perception of and reasoning about objects, making it to cover *actions* related to the objects as well.
- Studying concepts representing non-object categories, such as event and situation categories, to see whether the framework must be modified to cover also this kind of categories.
- Examining the actual relevance of the metaphysical function and its relation to categorization by core, i.e., whether there actually are two fundamentally different ways of categorizing objects.
- Applying and evaluating the SD methods for learning characteristic descriptions to other learning algorithms than ID3 and IB1.
- Implement the outlined general model of concept learning AQGMCL.

While these topics are closely related to my own research, there are also several more general issues that need to be addressed in the future. For instance:

- The relationship between properties, similarity, and categorical perception.
- Representation schemes and languages that can handle all the relevant kinds of properties.
- Models of similarity for such representations.
- Top-down category formation.
- Learning inferential knowledge.

However, although it has begun to receive considerable attention, the most fundamental problem to be solved regarding concept learning by autonomous agent is how to integrate learning algorithms with perception systems successfully.

Appendix A

Empirical Results — SD approach

This appendix provides a summary of the empirical evaluation of the SD approach. Three data sets were used: Iris, Wine, and Coin. Three different algorithms, ID3-SD, IB1-SD, and IB1-SD-multi, were tested on each of the data sets (no IB1 on the Coin data set yet). As we here are interested in the behaviour of the algorithms when confronted with unknown categories, not all of the categories present in the data sets were used in the training phase. For each experiment, training and test sets were randomly chosen without replacement. Below are some data set specific details of the experiments:

Iris database contains 3 categories of 50 instances each, where a category refers to a type of Iris plant (Setosa, Versicolor or Virginica). 4 numerical attributes (sepal length, sepal width, petal length, and petal width) are numerical. 50 (2$\times$25) instances were used for training and 75 (3$\times$25) for testing. Note that Versicolor and Virginica are hard to separate whereas Setosa is easily separated from the other two.

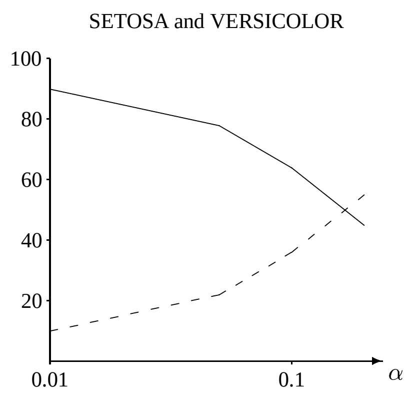
Wine database contains results of chemical analyses of wines grown in the same region of Italy, but are fermented using three different kinds of yeast. 13 numerical attributes (different constituents). The data set consists of 59 instances of wine of type 1, 71 of type 2, and 48 of type 3. 50 (2$\times$25) instances were used for training and 60 (3$\times$20) instances for testing.

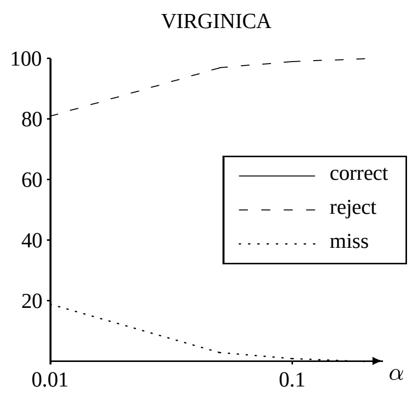
Coin databases contains measurements of the coins of two countries, Canadian coins contains 7 categories (1, 5, 10, 25, 50 cent, 1 and 2 dollar), and Hong Kong coins that also contains 7 categories (5, 10, 20, 50 cent, 1, 2, and 5 dollar). 5 numerical attributes (diameter, thickness, conductivity1, conductivity2, and permeability). 140 (7$\times$20) instances were used for training and 700 (2$\times$7$\times$50) for testing.

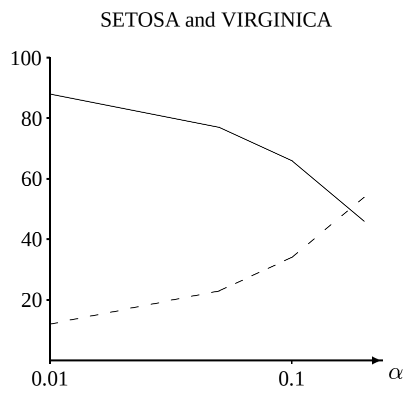
	SETOSA			VERSICOLOR			VIRGINICA		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
ID3	100.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0	0.0
ID3-Max	68.8	0.0	31.2	76.4	0.0	23.6	0.0	0.4	99.6
ID3-SD 0.2	42.4	0.0	57.6	47.2	0.0	52.8	0.0	0.0	100.0
ID3-SD 0.1	62.8	0.0	37.2	65.2	0.0	34.8	0.0	1.2	98.8
ID3-SD 0.05	73.2	0.0	26.8	82.0	0.0	18.0	0.0	2.8	97.2
ID3-SD 0.01	84.4	0.0	15.6	95.2	0.0	4.8	0.0	18.8	81.2

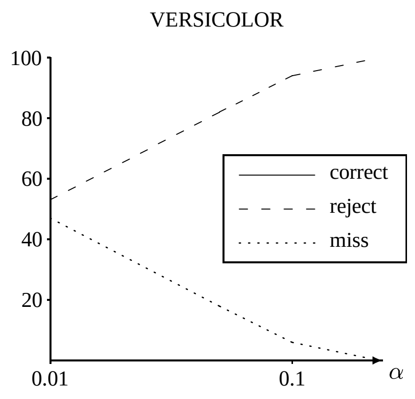
	SETOSA			VERSICOLOR			VIRGINICA		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
ID3	98.8	1.2	0.0	0.0	100.0	0.0	99.2	0.8	0.0
ID3-Max	68.8	0.0	31.2	0.0	14.8	85.2	74.0	0.0	26.0
ID3-SD 0.2	42.4	0.0	57.6	0.0	1.2	98.8	49.6	0.0	50.4
ID3-SD 0.1	62.4	0.0	37.6	0.0	5.6	94.4	70.4	0.0	29.6
ID3-SD 0.05	74.0	0.0	26.0	0.0	17.6	82.4	80.0	0.0	20.0
ID3-SD 0.01	84.0	0.0	16.0	0.0	47.2	52.8	91.2	0.0	8.8

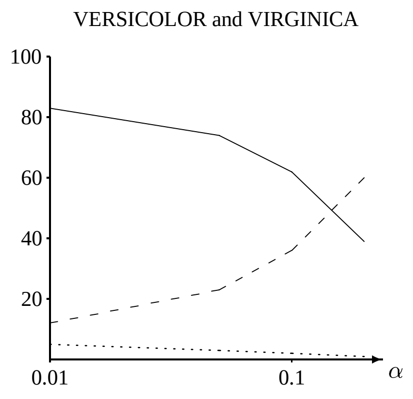
	SETOSA			VERSICOLOR			VIRGINICA		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
ID3	0.0	100.0	0.0	91.6	8.4	0.0	90.8	9.2	0.0
ID3-max	0.0	0.0	100.0	66.0	2.0	32.0	65.2	0.0	34.8
ID3-SD 0.2	0.0	0.0	100.0	40.0	1.6	58.4	38.0	0.0	62.0
ID3-SD 0.1	0.0	0.0	100.0	60.0	3.2	36.8	64.4	0.0	35.6
ID3-SD 0.05	0.0	0.0	100.0	74.0	4.8	21.2	74.4	1.6	24.0
ID3-SD 0.01	0.0	0.0	100.0	84.4	4.8	10.8	82.8	4.8	12.4

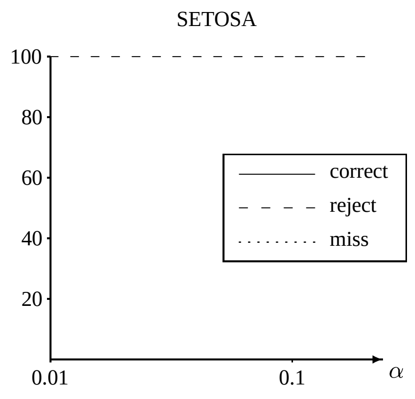

SETOSA and VERSICOLOR
100
80
60
40
20
0.01
0.1
α


VIRGINICA
100
80
60
40
20
0.01
0.1
α
correct
reject
miss


SETOSA and VIRGINICA
100
80
60
40
20
0.01
0.1
α


VERSICOLOR
100
80
60
40
20
0.01
0.1
α
correct
reject
miss

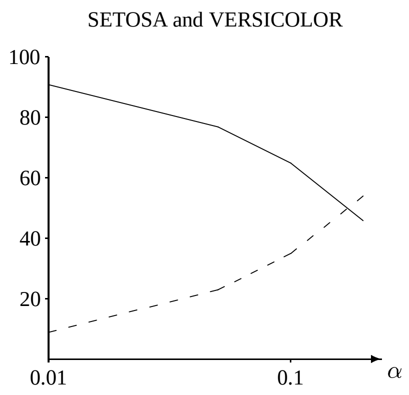

VERSICOLOR and VIRGINICA
100
80
60
40
20
0.01
0.1
α

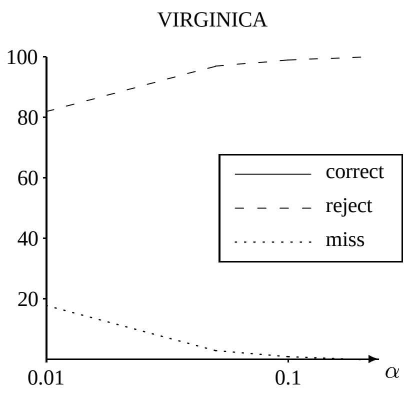

SETOSA
100
80
60
40
20
0.01
0.1
α
correct
reject
miss

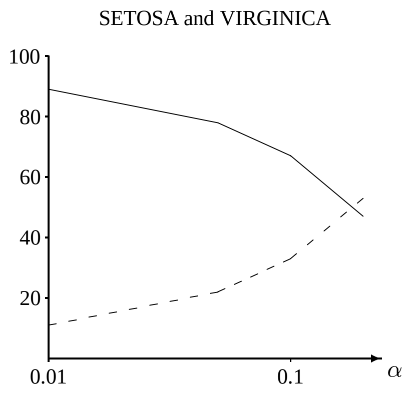
	SETOSA			VERSICOLOR			VIRGINICA		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
IB1	100.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0	0.0
IB1-SD 0.2	46.3	0.0	53.7	46.7	0.0	53.3	0.0	0.0	100.0
IB1-SD 0.1	65.0	0.0	35.0	66.2	0.0	33.8	0.0	0.8	99.2
IB1-SD 0.05	74.5	0.0	25.5	80.1	0.0	19.9	0.0	3.2	96.8
IB1-SD 0.01	88.5	0.0	11.5	93.7	0.0	6.3	0.0	17.7	82.3

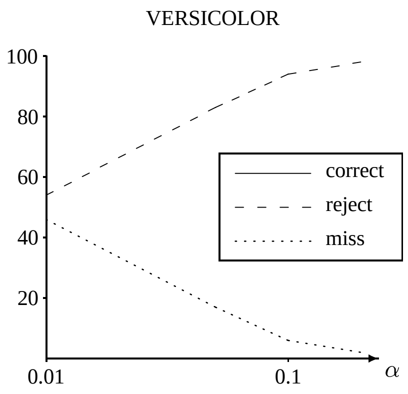
	SETOSA			VERSICOLOR			VIRGINICA		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
IB1	100.0	0.0	0.0	0.0	100.0	0.0	100.0	0.0	0.0
IB1-SD 0.2	47.3	0.0	52.7	0.0	2.1	97.9	49.9	0.0	50.1
IB1-SD 0.1	66.0	0.0	34.0	0.0	7.4	92.6	71.7	0.0	28.3
IB1-SD 0.05	75.5	0.0	24.0	0.0	17.0	83.0	80.4	0.0	19.6
IB1-SD 0.01	89.5	0.0	10.5	0.0	46.3	53.7	92.0	0.0	8.0

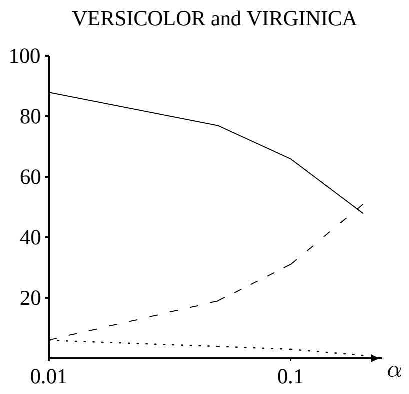
	SETOSA			VERSICOLOR			VIRGINICA		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
IB1	0.0	100.0	0.0	92.1	7.9	0.0	93.5	6.5	0.0
IB1-SD 0.2	0.0	0.0	100.0	46.2	1.4	52.4	49.6	0.0	50.4
IB1-SD 0.1	0.0	0.0	100.0	62.0	4.8	33.2	71.0	0.3	28.7
IB1-SD 0.05	0.0	0.0	100.0	74.7	5.6	19.7	78.9	2.1	19.0
IB1-SD 0.01	0.0	0.0	100.0	88.6	7.6	3.8	87.5	5.1	7.4

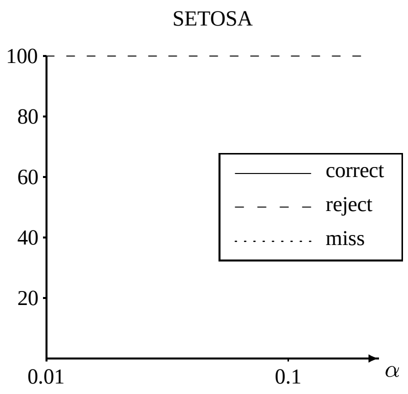

SETOSA and VERSICOLOR
100
80
60
40
20
0.01
0.1
α


VIRGINICA
100
80
60
40
20
0.01
0.1
α
correct
reject
miss


SETOSA and VIRGINICA
100
80
60
40
20
0.01
0.1
α


VERSICOLOR
100
80
60
40
20
0.01
0.1
α
correct
reject
miss

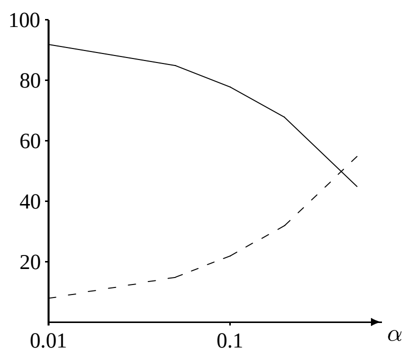

VERSICOLOR and VIRGINICA
100
80
60
40
20
0.01
0.1
α

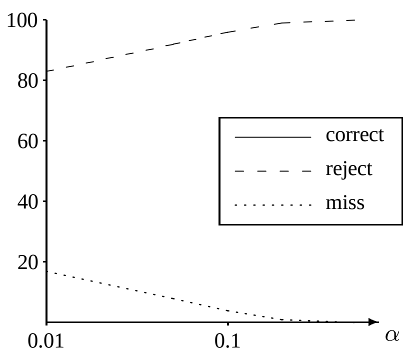

SETOSA
100
80
60
40
20
0.01
0.1
α
correct
reject
miss

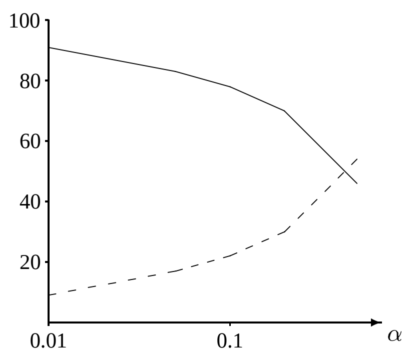
	SETOSA			VERSICOLOR			VIRGINICA		
	corr.	miss	reject	corr.	miss	reject	corr.	miss	reject
IB1-SD-multi 0.5	48.5	0.0	51.5	40.5	0.0	59.5	0.0	0.1	99.9
IB1-SD-multi 0.4	54.7	0.0	45.3	48.0	0.0	52.0	0.0	0.4	99.6
IB1-SD-multi 0.3	61.1	0.0	38.9	56.3	0.0	43.7	0.0	0.8	99.2
IB1-SD-multi 0.2	68.2	0.0	31.8	67.3	0.0	32.7	0.0	1.4	98.6
IB1-SD-multi 0.1	76.4	0.0	23.6	79.3	0.0	20.7	0.0	4.2	95.8
IB1-SD-multi 0.05	81.3	0.0	18.7	88.5	0.0	11.5	0.0	7.9	92.1
IB1-SD-multi 0.01	89.1	0.0	10.9	95.7	0.0	4.3	0.0	17.1	82.3

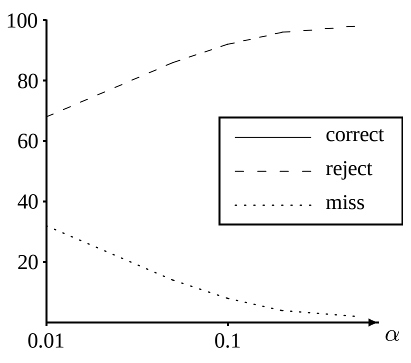
	SETOSA			VERSICOLOR			VIRGINICA		
	corr.	miss	reject	corr.	miss	reject	corr.	miss	reject
IB1-SD-multi 0.5	48.5	0.0	51.5	0.0	1.5	98.5	43.8	0.0	56.2
IB1-SD-multi 0.4	54.7	0.0	45.3	0.0	2.0	98.0	54.3	0.0	45.7
IB1-SD-multi 0.3	61.1	0.0	38.9	0.0	2.9	97.1	64.3	0.0	35.7
IB1-SD-multi 0.2	68.2	0.0	31.8	0.0	4.0	96.0	71.6	0.0	28.4
IB1-SD-multi 0.1	76.4	0.0	23.6	0.0	8.4	91.6	80.2	0.0	19.8
IB1-SD-multi 0.05	81.3	0.0	18.7	0.0	14.1	85.9	85.4	0.0	14.6
IB1-SD-multi 0.01	89.1	0.0	10.9	0.0	31.7	68.3	93.6	0.0	6.4

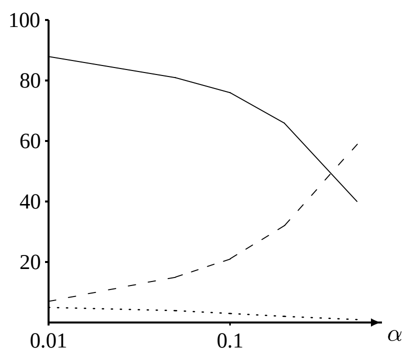
	SETOSA			VERSICOLOR			VIRGINICA		
	corr.	miss	reject	corr.	miss	reject	corr.	miss	reject
IB1-SD-multi 0.5	0.0	0.0	100.0	36.4	1.3	62.3	43.7	0.0	56.3
IB1-SD-multi 0.4	0.0	0.0	100.0	44.0	2.0	54.0	53.8	0.0	46.2
IB1-SD-multi 0.3	0.0	0.0	100.0	51.6	3.0	45.4	62.8	0.4	36.8
IB1-SD-multi 0.2	0.0	0.0	100.0	62.5	3.8	33.7	69.4	0.9	29.7
IB1-SD-multi 0.1	0.0	0.0	100.0	74.4	5.0	20.6	77.0	1.3	21.7
IB1-SD-multi 0.05	0.0	0.0	100.0	81.2	6.0	12.8	80.7	1.9	17.4
IB1-SD-multi 0.01	0.0	0.0	100.0	88.7	6.8	4.5	87.4	2.6	10.0

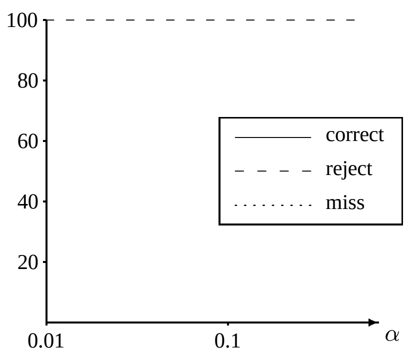

SETOSA and VERSICOLOR


VIRGINICA


SETOSA and VIRGINICA


VERSICOLOR


VERSICOLOR and VIRGINICA

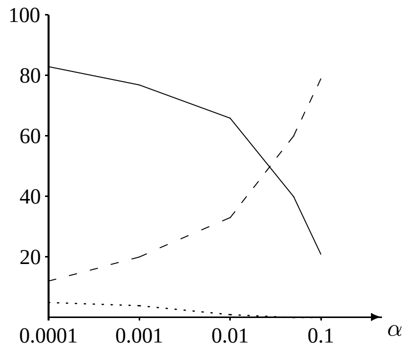

SETOSA

	WINE I			WINE II			WINE III		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
ID3	93.0	7.0	0.0	90.0	10.0	0.0	0.0	100.0	0.0
ID3-Max	31.0	0.0	69.0	27.5	0.0	72.5	0.0	1.0	99.0
ID3-SD 0.1	19.0	0.0	81.0	22.5	0.0	77.5	0.0	0.0	100.0
ID3-SD 0.05	42.0	0.0	58.0	37.5	0.0	62.5	0.0	0.0	100.0
ID3-SD 0.01	67.5	3.0	29.5	63.5	0.0	36.5	0.0	1.5	98.5
ID3-SD 0.001	77.5	6.0	16.5	75.5	1.5	23.0	0.0	8.0	92.0
ID3-SD 0.0001	85.5	7.0	7.5	80.0	2.5	17.5	0.0	12.0	88.0

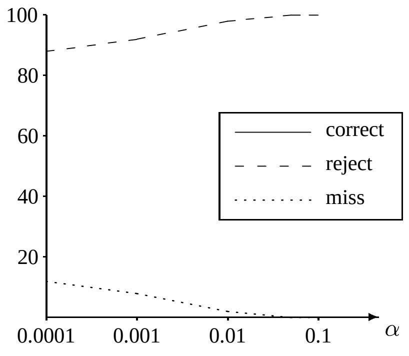
	WINE I			WINE II			WINE III		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
ID3	99.5	0.5	0.0	0.0	100.0	0.0	99.0	1.0	0.0
ID3-Max	33.0	0.0	67.0	0.0	0.0	100.0	35.0	0.0	65.0
ID3-SD 0.1	20.0	0.0	80.0	0.0	0.0	100.0	26.5	0.0	73.5
ID3-SD 0.05	46.0	0.0	54.0	0.0	0.5	99.5	44.0	0.0	56.0
ID3-SD 0.01	74.0	0.0	26.0	0.0	5.0	95.0	78.5	0.0	21.5
ID3-SD 0.001	86.5	0.0	13.5	0.0	40.5	59.5	91.0	0.0	9.0
ID3-SD 0.0001	94.5	0.0	5.5	0.0	59.0	31.0	95.0	0.0	5.0

	WINE I			WINE II			WINE III		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
ID3	0.0	100.0	0.0	91.5	8.5	0.0	84.5	15.5	0.0
ID3-Max	0.0	2.5	97.5	31.0	0.0	69.0	32.0	0.0	68.0
ID3-SD 0.1	0.0	0.0	100.0	25.5	0.0	74.5	24.0	0.0	76.0
ID3-SD 0.05	0.0	0.0	100.0	43.0	0.0	57.0	40.5	0.0	59.5
ID3-SD 0.01	0.0	10.0	90.0	65.5	0.5	34.0	64.5	2.0	33.5
ID3-SD 0.001	0.0	22.0	78.0	77.0	3.0	20.0	73.0	6.0	21.0
ID3-SD 0.0001	0.0	29.5	70.5	83.0	4.0	13.0	77.0	10.5	12.5

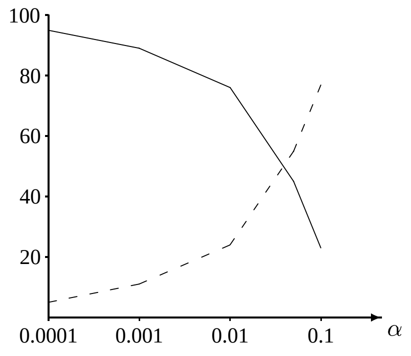
WINE I and WINE II



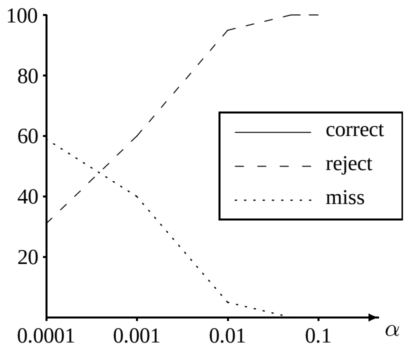
WINE III



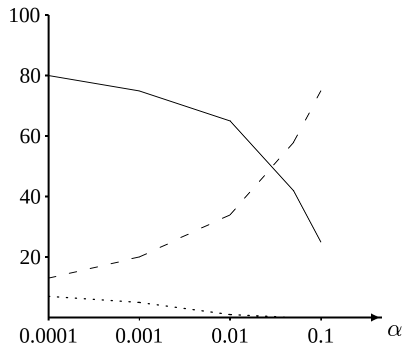
WINE I and WINE III



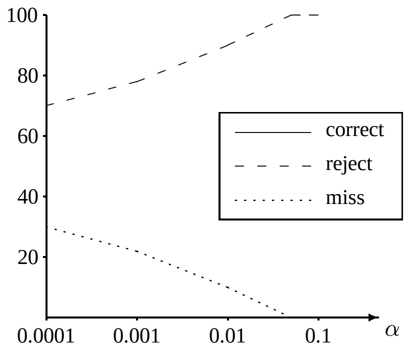
WINE II



WINE II and WINE III



WINE I

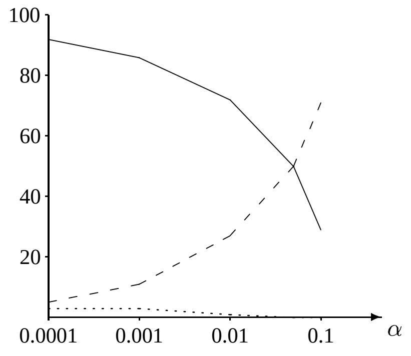


	WINE I			WINE II			WINE III		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
IB1	99.5	0.5	0.0	91.0	9.0	0.0	0.0	100.0	0.0
IB1-SD 0.1	30.0	0.0	70.0	28.3	0.0	71.7	0.0	0.0	100.0
IB1-SD 0.05	55.3	0.0	44.7	44.0	0.3	55.7	0.0	0.5	99.5
IB1-SD 0.01	75.9	0.1	24.0	69.0	2.1	28.9	0.0	8.9	91.1
IB1-SD 0.001	87.6	0.4	12.0	83.7	5.4	10.9	0.0	28.6	71.4
IB1-SD 0.0001	94.8	0.5	4.7	88.2	6.6	5.2	0.0	37.1	62.9

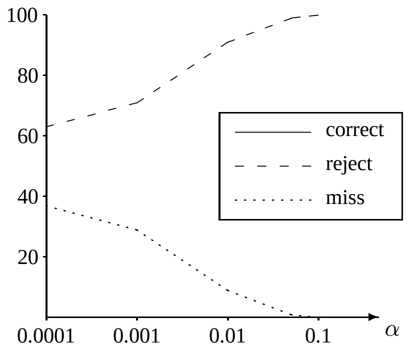
	WINE I			WINE II			WINE III		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
IB1	100.0	0.0	0.0	0.0	100.0	0.0	100.0	0.0	0.0
IB1-SD 0.1	30.1	0.0	69.9	0.0	0.0	100.0	30.7	0.0	69.3
IB1-SD 0.05	55.6	0.0	44.4	0.0	0.4	99.6	51.3	0.0	48.7
IB1-SD 0.01	76.2	0.0	23.8	0.0	9.5	90.5	83.8	0.0	16.2
IB1-SD 0.001	87.9	0.0	12.1	0.0	45.1	54.9	95.6	0.0	4.4
IB1-SD 0.0001	95.1	0.0	4.9	0.0	71.4	28.6	97.8	0.0	2.2

	WINE I			WINE II			WINE III		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
IB1	0.0	100.0	0.0	92.5	7.5	0.0	99.4	0.6	0.0
IB1-SD 0.1	0.0	0.0	100.0	29.4	0.0	70.6	30.7	0.0	69.3
IB1-SD 0.05	0.0	0.6	99.4	48.1	0.0	51.9	51.3	0.0	48.7
IB1-SD 0.01	0.0	13.2	86.8	71.6	2.6	25.8	83.2	0.4	16.4
IB1-SD 0.001	0.0	39.1	60.9	83.3	5.9	10.8	95.0	0.5	4.5
IB1-SD 0.0001	0.0	56.9	43.1	89.3	7.0	3.7	97.2	0.5	2.3

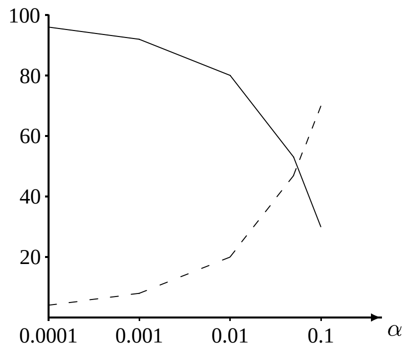
WINE I and WINE II



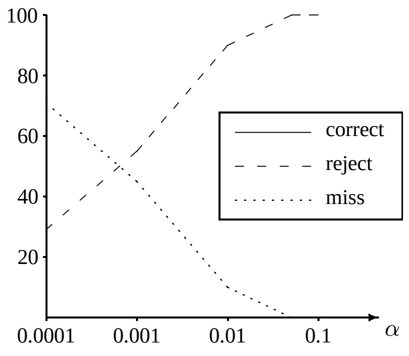
WINE III



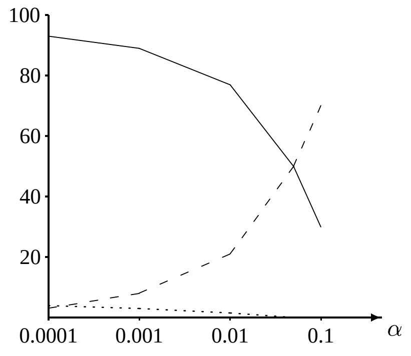
WINE I and WINE III



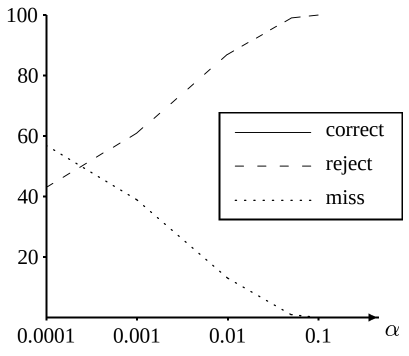
WINE II



WINE II and WINE III



WINE I

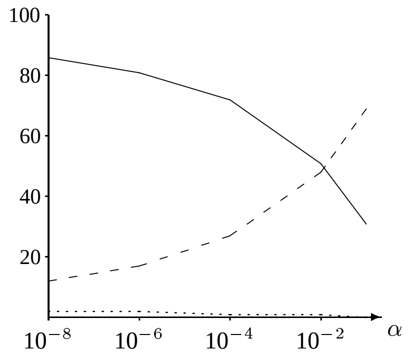


	WINE I			WINE II			WINE III		
	corr.	miss	reject	corr.	miss	reject	corr.	miss	reject
IB1-SD-multi 0.1	31.0	0.0	69.0	30.1	0.0	69.5	0.0	0.0	100.0
IB1-SD-multi 0.01	53.5	0.0	46.5	48.2	1.3	50.5	0.0	0.1	99.9
IB1-SD-multi 0.001	68.4	0.0	31.6	59.4	1.7	38.9	0.0	1.2	98.8
IB1-SD-multi 10^{-4}	77.3	0.3	22.4	66.4	2.1	31.5	0.0	3.0	97.0
IB1-SD-multi 10^{-5}	83.3	0.3	16.4	70.8	2.6	26.6	0.0	5.0	95.0
IB1-SD-multi 10^{-6}	87.8	0.4	11.8	74.3	2.9	22.8	0.0	7.0	93.0
IB1-SD-multi 10^{-8}	92.2	0.4	7.4	80.2	3.4	16.4	0.0	11.9	88.1

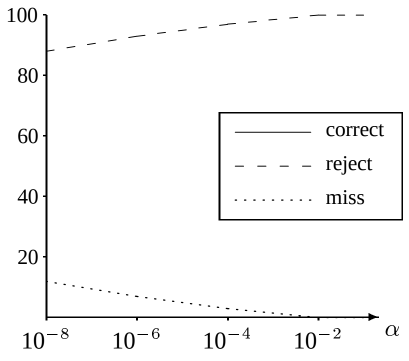
	WINE I			WINE II			WINE III		
	corr.	miss	reject	corr.	miss	reject	corr.	miss	reject
IB1-SD-multi 0.1	31.0	0.0	69.0	0.0	0.5	99.5	28.3	0.0	71.7
IB1-SD-multi 0.01	53.6	0.0	46.4	0.0	2.3	97.7	49.5	0.0	50.5
IB1-SD-multi 0.001	68.6	0.0	31.4	0.0	3.9	96.1	64.6	0.0	35.4
IB1-SD-multi 10^{-4}	77.5	0.0	22.5	0.0	5.5	94.5	73.7	0.0	26.3
IB1-SD-multi 10^{-5}	83.5	0.0	16.5	0.0	8.3	91.7	81.2	0.0	18.8
IB1-SD-multi 10^{-6}	88.1	0.0	11.9	0.0	10.3	89.7	85.8	0.0	14.2
IB1-SD-multi 10^{-8}	92.5	0.0	7.5	0.0	18.7	81.3	92.9	0.0	7.1

	WINE I			WINE II			WINE III		
	corr.	miss	reject	corr.	miss	reject	corr.	miss	reject
IB1-SD-multi 0.1	0.0	0.1	99.9	35.5	0.0	64.5	28.3	0.0	71.7
IB1-SD-multi 0.01	0.0	2.5	97.5	54.2	0.0	45.8	49.5	0.0	50.5
IB1-SD-multi 0.001	0.0	7.0	93.0	63.9	0.0	36.1	64.5	0.0	35.5
IB1-SD-multi 10^{-4}	0.0	11.3	88.7	71.9	0.0	28.1	73.4	0.0	26.6
IB1-SD-multi 10^{-5}	0.0	15.6	84.4	76.5	0.0	23.5	80.8	0.0	19.2
IB1-SD-multi 10^{-6}	0.0	19.9	80.1	80.1	0.0	19.9	85.4	0.0	14.6
IB1-SD-multi 10^{-8}	0.0	30.1	69.9	84.4	0.1	15.5	92.4	0.2	7.4

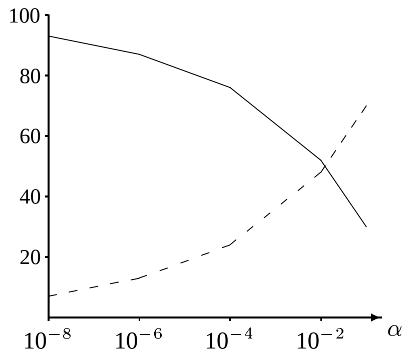
WINE I and WINE II



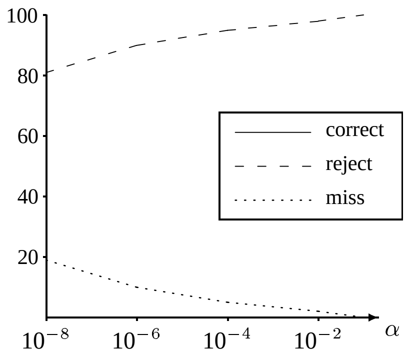
WINE III



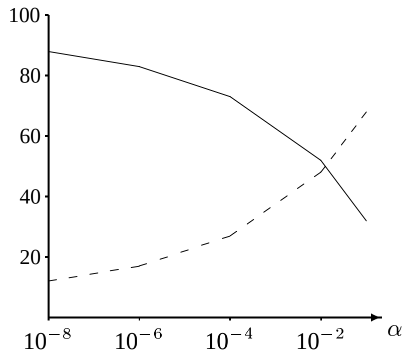
WINE I and WINE III



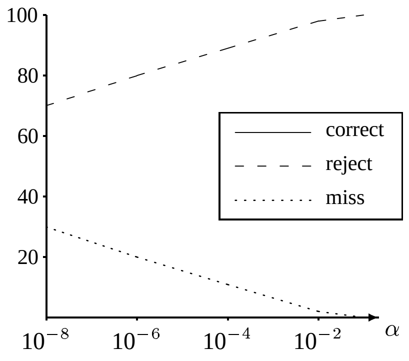
WINE II



WINE II and WINE III

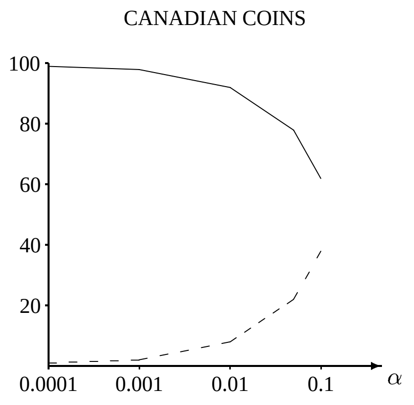


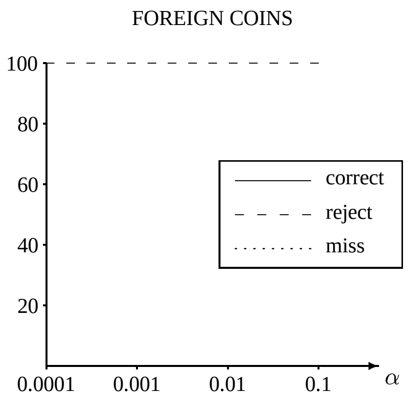
WINE I

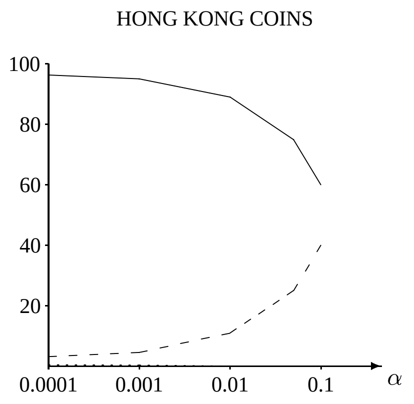


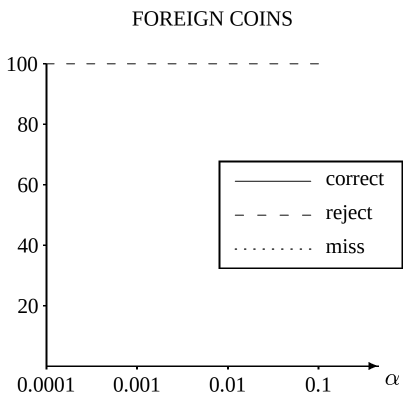
	CANADIAN COINS			FOREIGN COINS		
	correct	miss	reject	correct	miss	reject
ID3	99.7	0.3	0.0	0.0	100.0	0.0
ID3-Max	83.7	0.0	16.3	0.0	0.0	100.0
ID3-SD 0.1	62.1	0.0	37.9	0.0	0.0	100.0
ID3-SD 0.05	77.5	0.0	22.5	0.0	0.0	100.0
ID3-SD 0.01	92.2	0.0	7.8	0.0	0.0	100.0
ID3-SD 0.001	97.9	0.0	2.1	0.0	0.0	100.0
ID3-SD 0.0001	98.9	0.0	1.1	0.0	0.0	100.0

	HONG KONG COINS			FOREIGN COINS		
	correct	miss	reject	correct	miss	reject
ID3	98.3	1.7	0.0	0.0	100.0	0.0
ID3-Max	79.7	0.0	20.3	0.0	0.0	100.0
ID3-SD 0.1	60.0	0.0	40.0	0.0	0.0	100.0
ID3-SD 0.05	74.8	0.0	25.2	0.0	0.0	100.0
ID3-SD 0.01	88.9	0.0	11.1	0.0	0.0	100.0
ID3-SD 0.001	95.1	0.3	4.6	0.0	0.0	100.0
ID3-SD 0.0001	96.3	0.5	3.2	0.0	0.0	100.0


CANADIAN COINS
100
80
60
40
20
0.0001
0.001
0.01
0.1
α

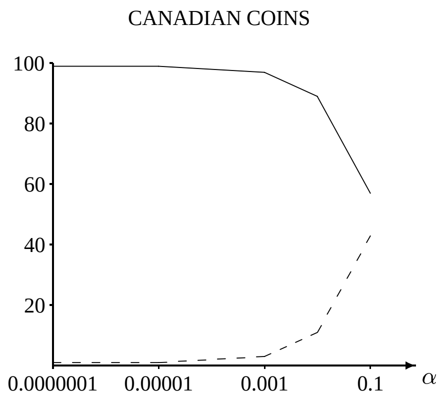

FOREIGN COINS
100
80
60
40
20
0.0001
0.001
0.01
0.1
α
correct
reject
miss

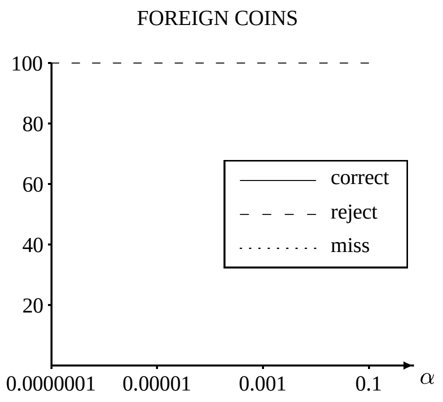

HONG KONG COINS
100
80
60
40
20
0.0001
0.001
0.01
0.1
α

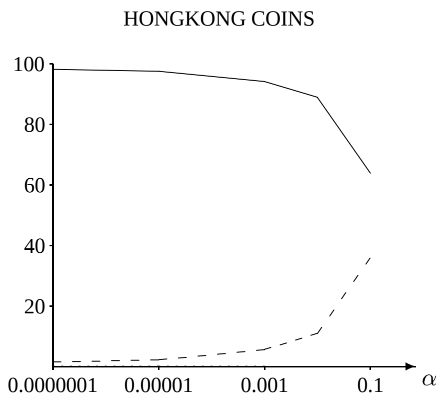

FOREIGN COINS
100
80
60
40
20
0.0001
0.001
0.01
0.1
α
correct
reject
miss

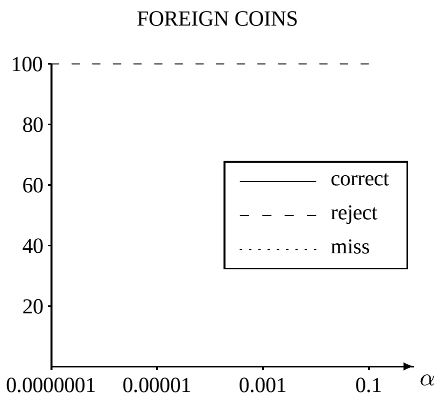
	CANADIAN COINS			FOREIGN COINS		
	correct	miss	reject	correct	miss	reject
IB1	100.0	0.0	0.0	0.0	100.0	0.0
IB1-SD 0.1	56.9	0.0	43.1	0.0	0.0	100.0
IB1-SD 0.05	72.1	0.0	27.9	0.0	0.0	100.0
IB1-SD 0.01	89.3	0.0	10.7	0.0	0.0	100.0
IB1-SD 0.001	96.8	0.0	3.2	0.0	0.0	100.0
IB1-SD 0.0001	98.4	0.0	1.6	0.0	0.0	100.0
IB1-SD 0.00001	98.9	0.0	1.1	0.0	0.0	100.0
IB1-SD 0.000001	99.5	0.0	0.5	0.0	0.0	100.0
IB1-SD 0.0000001	99.6	0.0	0.4	0.0	0.0	100.0

	HONG KONG COINS			FOREIGN COINS		
	correct	miss	reject	correct	miss	reject
IB1	99.3	0.7	0.0	0.0	100.0	0.0
IB1-SD 0.1	63.8	0.0	36.2	0.0	0.0	100.0
IB1-SD 0.05	76.9	0.0	23.1	0.0	0.0	100.0
IB1-SD 0.01	89.0	0.0	11.0	0.0	0.0	100.0
IB1-SD 0.001	94.2	0.1	5.7	0.0	0.0	100.0
IB1-SD 0.0001	95.7	0.1	4.2	0.0	0.0	100.0
IB1-SD 0.00001	97.6	0.1	2.3	0.0	0.0	100.0
IB1-SD 0.000001	98.2	0.1	1.7	0.0	0.0	100.0
IB1-SD 0.0000001	98.3	0.1	1.6	0.0	0.0	100.0


CANADIAN COINS
100
80
60
40
20
0.0000001
0.00001
0.001
0.1
α

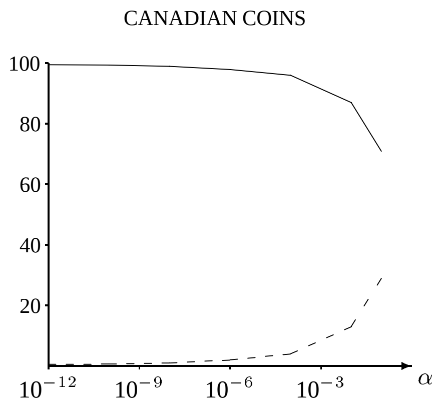

FOREIGN COINS
100
80
60
40
20
correct
reject
miss
0.0000001
0.00001
0.001
0.1
α

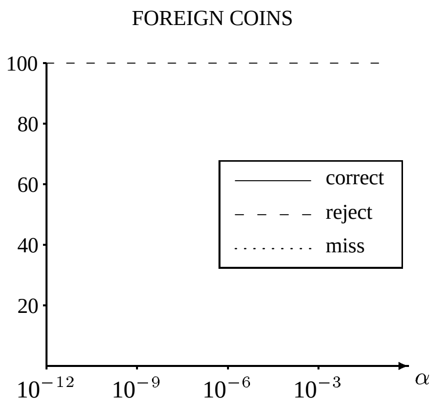

HONGKONG COINS
100
80
60
40
20
0.0000001
0.00001
0.001
0.1
α

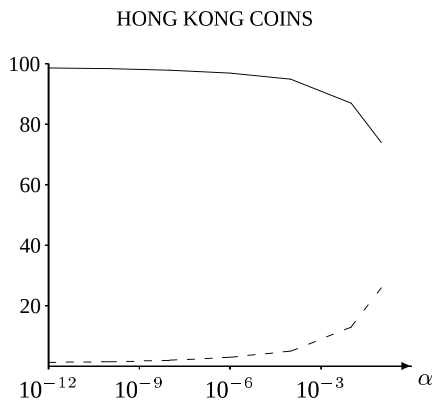

FOREIGN COINS
100
80
60
40
20
correct
reject
miss
0.0000001
0.00001
0.001
0.1
α

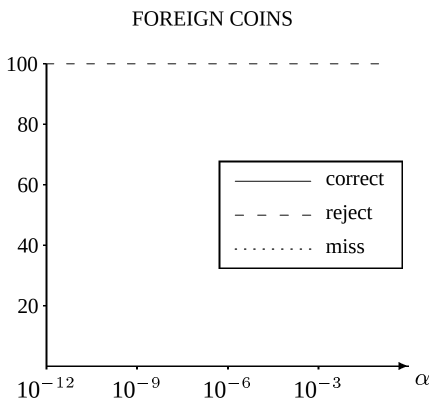
	CANADIAN COINS			FOREIGN COINS		
	correct	miss	reject	correct	miss	reject
IB1-SD-multi 0.1	70.9	0.0	29.1	0.0	0.0	100.0
IB1-SD-multi 0.01	87.3	0.0	12.7	0.0	0.0	100.0
IB1-SD-multi 0.001	93.5	0.0	6.5	0.0	0.0	100.0
IB1-SD-multi 10^{-4}	95.8	0.0	4.2	0.0	0.0	100.0
IB1-SD-multi 10^{-6}	98.1	0.0	1.9	0.0	0.0	100.0
IB1-SD-multi 10^{-8}	99.1	0.0	0.9	0.0	0.0	100.0
IB1-SD-multi 10^{-10}	99.4	0.0	0.6	0.0	0.0	100.0
IB1-SD-multi 10^{-12}	99.5	0.0	0.5	0.0	0.0	100.0

	HONG KONG COINS			FOREIGN COINS		
	correct	miss	reject	correct	miss	reject
IB1-SD-multi 0.1	74.3	0.0	25.7	0.0	0.0	100.0
IB1-SD-multi 0.01	87.4	0.0	12.6	0.0	0.0	100.0
IB1-SD-multi 0.001	92.1	0.0	7.9	0.0	0.0	100.0
IB1-SD-multi 10^{-4}	94.8	0.0	5.2	0.0	0.0	100.0
IB1-SD-multi 10^{-6}	97.0	0.0	3.0	0.0	0.0	100.0
IB1-SD-multi 10^{-8}	98.0	0.0	2.0	0.0	0.0	100.0
IB1-SD-multi 10^{-10}	98.5	0.0	1.5	0.0	0.0	100.0
IB1-SD-multi 10^{-12}	98.7	0.0	1.3	0.0	0.0	100.0


CANADIAN COINS
100
80
60
40
20
10^{-12}
10^{-9}
10^{-6}
10^{-3}
α


FOREIGN COINS
100
80
60
40
20
correct
reject
miss
10^{-12}
10^{-9}
10^{-6}
10^{-3}
α


HONG KONG COINS
100
80
60
40
20
10^{-12}
10^{-9}
10^{-6}
10^{-3}
α


FOREIGN COINS
100
80
60
40
20
correct
reject
miss
10^{-12}
10^{-9}
10^{-6}
10^{-3}
α

Bibliography

[1] *AAAI Fall Symposium Series, Instantiating Real-World Agents, (FS–93–03)*. AAAI Press, 1993.

[2] *AAAI Fall Symposium Series, Machine Learning in Computer Vision, (FS–93–04)*. AAAI Press, 1993.

[3] P.E. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *AAAI-87*, pages 268–272. Morgan Kaufmann, 1987.

[4] D.W. Aha. Generalizing from case studies: A case study. In *Ninth International Workshop on Machine Learning*, pages 1–10. Morgan Kaufmann, 1992.

[5] D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[6] J.S. Albus. Outline for a theory of intelligence. *IEEE Trans. on Systems, Man, and Cybernetics*, 21(3):473–509, 1991.

[7] Y. Aloimonos and A. Rosenfeld. Computer vision. *Science*, 253:1249–1254, 1991.

[8] J.A. Ambros-Ingerson and S. Steel. Integrating planning, execution and monitoring. In *AAAI-88*, pages 607–611. Morgan Kaufmann, 1988.

[9] J. Amsterdam. Some philosophical problems with formal learning theory. In *AAAI-88*, pages 580–584. Morgan Kaufmann, 1988.

[10] E. Anderson. The Irises of the Gaspe Peninsula. *Bulletin of the American Iris Society*, 59:2–5, 1935.

[11] J.R. Anderson and M. Matessa. An incremental bayesian algorithm for categorization. In D.H. Fischer, M.J. Pazzani, and P. Langley, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, pages 45–70. Morgan Kaufmann, 1991.

[12] D. Angluin and C.H. Smith. Inductive inference: Theory and methods. *Computing Surveys*, 15:237–268, 1983.

[13] M. Asada. Map building for a mobile robot from sensory data. *IEEE Trans. on Systems, Man, and Cybernetics*, 20(6):1326–1336, 1990.

[14] K.J. Åström and B. Wittenmark. *Computer Controlled Systems: Theory and Design*. Prentice-Hall, 1984.

[15] R.C. Atkinson and W.K. Estes. Stimulus sampling theory. In R.D. Luce, R.R. Bush, and E. Galanter, editors, *Handbook of Mathematical Psychology*. Wiley, 1963.

[16] R.L. Atkinson, R.C. Atkinson, E.E. Smith, and E.R. Hilgard. *Introduction to Psychology, ninth edition*. Harcourt Brace Jovanovic Publishers, 1987.

[17] G.P. Baker and P.M.S. Hacker. *Language, Sense & Nonsense*. Basil Blackwell, 1984.

[18] C. Balkenius. Neural mechanisms for self-organization of emergent schemata, dynamical schema processing, and semantic constraint satisfaction. Lund University Cognitive Studies 14, ISSN 1101–8453, Lund University, Sweden, 1993.

[19] C. Balkenius and P. Gärdenfors. Nonmonotonic inferences in neural networks. Lund University Cognitive Studies 3, ISSN 1101–8453, Lund University, Sweden, 1991.

[20] D. H. Ballard. Animate vision. *Artificial Intelligence*, 48(1):57–86, 1991.

[21] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W. Whittaker. Ambler: An autonomous rover for planetary exploration. *IEEE Computer*, 22(6):18–26, 1989.

[22] A.H. Barr. Superquadrics and angle-preserving transformations. *IEEE Comput. Graph. Appl.*, 1(1):11–23, 1981.

[23] L.W. Barsalou. Are there static category representations in long-term memory? *Behavioral and Brain Sciences*, 9:651–652, 1986.

[24] J. Barwise and J. Etchemendy. Model-theoretic semantics. In M.L. Posner, editor, *Foundations of Cognitive Science*, pages 207–243. MIT Press, 1989.

[25] O. Belo and J. Neves. Cooperation among opportunistic agents in a distributed environment. In C. Zhang and D. Lukose, editors, *Distributed Artificial Intelligence (preliminary title)*. Springer Verlag, 1996. To be published in the Lecture Notes in Artificial Intelligence series.

[26] F. Bergadano, S. Matwin, R.S. Michalski, and J. Zhang. Learning two-tiered descriptions of flexible concepts, part I: Principles and methodology. Technical Report MLI 88–6 TR–14–88, Machine Learning & Inference Laboratory, Center for Artificial Intelligence, George Mason University, 1988.

[27] F. Bergadano, S. Matwin, R.S. Michalski, and J. Zhang. Learning two-tiered descriptions of flexible concepts: The POSEIDON system. *Machine Learning*, 8(1):5–43, 1992.

[28] M.K. Bhandaru, B.A. Draper, and V.R. Lesser. Learning image to symbol conversion. In *AAAI Fall Symposium on Machine Learning and Computer Vision, (FS–93–04)*. AAAI Press, 1993.

[29] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

[30] L. Birnbaum. Rigor mortis: A response to Nilsson's "Logic and artificial intelligence". *Artificial Intelligence*, 47(1):57–77, 1991.

[31] M.J. Black et al. Action, representation, and purpose: Re-evaluating the foundations of computational vision. In *IJCAI-93*, pages 1661–1666. Morgan Kaufmann, 1993.

[32] D.R. Blidberg. Autonomous underwater vehicles: Current activities and research opportunities. In T. Kanade, F.C.A. Groen, and L.O. Hertzberger, editors, *Intelligent Autonomous Systems 2*, 1989.

[33] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[34] R.M. Bolle and B.C. Vemuri. On three-dimensional surface reconstruction methods. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(1):1–13, 1991.

[35] M. Bowerman. The structure and origin of semantic categories in the language learning child. In M. Foster, editor, *Symbol as Sense: New Approaches to the Analysis of Meaning*, pages 277–299. Academic Press, 1980.

[36] M.E. Bratman, D.J. Israel, and M.E. Pollack. Plans and resource-bounded preactical reasoning. *Computational Intelligence*, 4:349–355, 1988.

[37] J. Bresina and M. Drummond. Integrating planning and reaction: A preliminary report. In J. Hendler, editor, *AAAI Spring Symposium on Planning in Uncertain, Unpredictable or Changing Environments*, 1990.

[38] R.A. Brooks. Symbolic reasoning among 3-d models and 2-d images. *Artificial Intelligence*, 17(1–3):285–348, 1981.

[39] R.A. Brooks. Elephants don't play chess. In P. Maes, editor, *Designing Autonomous Agents*, pages 3–15. MIT Press, 1990.

[40] R.A. Brooks. Intelligence without reason. In *IJCAI-91*, pages 569–595. Morgan Kaufmann, 1991.

[41] R.A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1):139–159, 1991.

[42] R.A. Brooks. New approaches to robotics. *Science*, 253:1227–1232, 1991.

[43] R.A. Brooks. The role of learning in autonomous robots. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 5–10. Morgan Kaufmann, 1991.

[44] J.S. Bruner, J.J. Goodnow, and G.A. Austin. *A Study of Thinking*. Wiley, 1956.

[45] W. Buntine. A critique of the Valiant model. In *IJCAI-89*, pages 837–842. Morgan Kaufmann, 1989.

[46] J.M. Burgers. Causality and anticipation. *Science*, 189:194–198, 1975.

[47] G.A. Carpenter and S. Grossberg. Neural dynamics of category learning and recognition: Attention, memory consolidation, and amnesia. In J. Davis, R. Newburgh, and E. Wegman, editors, *Brain Structure, Learning, and Memory, AAAS Symposium Series*, pages 239–286, 1986.

[48] G.A. Carpenter, S. Grossberg, and J.H. Reynolds. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, 4, 1991.

[49] B. Chandrasekaran. Coming out from under a cloud: AAAI and IAAI '93. *IEEE Expert*, 8(5):78–81, 1993.

[50] B. Chandrasekaran, N.H. Narayanan, and Y. Iwasaki. Reasoning with diagrammatic representation. *AI Magazine*, 14(2):49–56, 1993.

[51] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(2):333–378, 1987.

[52] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. AutoClass: A bayesian classification system. In *Fifth International Conference on Machine Learning*, pages 54–64. Morgan Kaufmann, 1988.

[53] P. Cheeseman, M. Self, J. Kelly, W. Taylor, D. Freeman, and J. Stutz. Bayesian classification. In *AAAI-88*, pages 607–611. Morgan Kaufmann, 1988.

[54] R.T. Chin and C.R. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, 18(1):67–108, 1986.

[55] J.H. Connell. *Minimalist Mobile Robots: A Colony-style Architecture for an Artificial Creature*. Academic Press, 1990.

[56] J.H. Connell and M. Brady. Generating and generalizing models of visual objects. *Artificial Intelligence*, 31(2):159–183, 1987.

[57] J.E. Corter. Relevant features and statistical models of generalization. *Behavioral and Brain Sciences*, 9:653–654, 1986.

[58] K. Craik. *The Nature of Explanation*. Camebridge University Press, 1943.

[59] P. Davidsson. On reactive planning. LU–CS–IR: 90–5, Department of Computer Science, Lund University, Lund, Sweden, 1990.

[60] R. Davis, H. Shrobe, and P. Szolovits. What is a knowledge representation? *AI Magazine*, 14(1):17–33, 1993.

[61] M. de la Maza. A prototype based symbolic concept learning system. In *Eighth International Workshop on Machine Learning*, pages 41–45. Morgan Kaufmann, 1991.

[62] T. Van de Merckt. Decision trees in numerical attribute spaces. In *IJCAI-93*, pages 1016–1021. Morgan Kaufmann, 1993.

[63] T. Dean and M. Boddy. An analysis of time-dependent planning. In *AAAI-88*, pages 49–54. Morgan Kaufmann, 1988.

[64] G. DeJong and R. Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176, 1986.

[65] D.C. Dennett. *The Intentional Stance*. MIT Press, 1987.

[66] K. Devlin. *Logic and Information*. Cambridge University Press, 1991.

[67] L. Dey, P.P. Das, and S. Chaudhury. Recognition and learning of unknown objects in a hierarchical knowledge-base. In *AAAI Fall Symposium on Machine Learning and Computer Vision, (FS–93–04)*. AAAI Press, 1993.

[68] S.J. Dickinson et al. The use of geons for generic 3-d object recognition. In *IJCAI-93*, pages 1693–1699. Morgan Kaufmann, 1993.

[69] T.G. Dietterich and R.S. Michalski. Inductive learning of structural descriptions. *Artificial Intelligence*, 16(3):257–294, 1981.

[70] G. Dorffner and E. Prem. Connectionism, symbol grounding and autonomous agents. Technical Report No. 17, Austrian Research Institute for Artificial Intelligence, Vienna, Austria, 1993.

[71] M. Drummond and J. Bresina. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *AAAI-90*, pages 138–144. MIT Press, 1990.

[72] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[73] M.A.E. Dummett. What is a theory of meaning? In S. Guttenplan, editor, *Mind and Language*, pages 97–138. Clarendon Press, 1975.

[74] M.A.E. Dummett. What is a theory of meaning? II. In Gareth Evans and John McDowell, editors, *Truth and Meaning: Essays in Semantics*, pages 67–137. Clarendon Press, 1976.

[75] M.A.E. Dummett. *Truth and Other Enigmas*. Duckworth, 1978.

[76] G.M. Edelman. *The Remembered Present: A Biological Theory of Consciousness*. Basic Books, 1989.

[77] B. Ekdahl. *Reflection and Computability in Computerized Agents (preliminary title)*. PhD thesis, Department of Computer Science, Lund University, Lund, Sweden, 1996. In preparation.

[78] S.L. Epstein. The role of memory and concepts in learning. *Minds and Machines*, 2(3):239–265, 1992.

[79] W.K. Estes. Models of categorization and category learning. *The Psychology of Learning and Motivation*, 29:15–56, 1993.

[80] O. Etzioni. Intelligence without robots: A reply to Brooks. *AI Magazine*, 14(4):7–13, 1993.

[81] B. Falkenheimer, K.D. Forbus, and D. Gentner. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41(1):1–63, 1989.

[82] I.A. Ferguson. *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents*. PhD thesis, University of Cambridge, 1992.

[83] I.A. Ferguson. TouringMachines: Autonomous agents with attitudes. *IEEE Computer*, 25(5):51–55, 1992.

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

[85] M.A. Fischler and O. Firschein, editors. *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*. Morgan Kaufmann, 1987.

[86] D. Fisher and P. Langley. Approaches to conceptual clustering. In *IJCAI-85*, pages 691–697. Morgan Kaufmann, 1985.

[87] D.H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[88] D.H. Fisher. A computational account of basic level and typicality effects. In *AAAI-88*, pages 233–238. Morgan Kaufmann, 1988.

[89] D.H. Fisher and K.B. McKusick. An empirical comparison of ID3 and back-propagation. In *IJCAI-89*, pages 788–793. Morgan Kaufmann, 1989.

[90] D.H. Fisher and M.J. Pazzani. Computational models of concept learning. In D.H. Fisher, M.J. Pazzani, and P. Langley, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, pages 3–43. Morgan Kaufmann, 1991.

[91] N.S. Flann and T.G. Dietterich. Selecting appropriate representations for learning from examples. In *AAAI-86*, pages 460–466. Morgan Kaufmann, 1986.

[92] J.A. Fodor. *The Language of Thought*. Thomas Y. Crowell, 1975.

[93] J.A. Fodor. *The Modularity of Mind*. Bradford Books, MIT Press, 1983.

[94] P. Gärdenfors. *Knowledge in Flux*. MIT Press, 1988.

[95] P. Gärdenfors. Frameworks for properties: Possible worlds vs. conceptual spaces. *Language, Knowledge, and Intentionality, Acta Philosophica Fennica*, 49:383–407, 1990.

[96] P. Gärdenfors. Three levels of inductive inference. Lund University Cognitive Studies 9, ISSN 1101–8453, Lund University, Sweden, 1992.

[97] E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *AAAI-92*, pages 809–815. MIT Press, 1992.

[98] E. Gat. On the role of stored internal state in the control of autonomous mobile robots. *AI Magazine*, 14(1):64–73, 1993.

[99] M.S. Gazzaniga. Organization of the human brain. *Science*, 245:947–952, 1989.

[100] M.R. Genesereth et al. Knowledge interchange format, version 3.0 reference manual. Technical Report Logic–92–1, Department of Computer Science, Stanford University, 1992.

[101] M.P. Georgeff and F.F. Ingrand. Decision-making in an embedded reasoning system. In *IJCAI-89*, pages 972–978. Morgan Kaufmann, 1989.

[102] M. Ginsberg. *Readings in Nonmonotonic Reasoning*. Morgan Kaufmann, 1987.

[103] M. Ginsberg. Universal planning: An (almost) universally bad idea. *AI Magazine*, 10(4):40–44, 1989.

[104] M.A. Gluck and G.H. Bower. From conditioning to category learning: An adaptive network model. *Journal of Experimental Psychology: General*, 117:225–244, 1988.

[105] M.A. Gluck and J.E. Corter. Information, uncertainty, and the utility of categories. In *Seventh Annual Conference of the Cognitive Science Society*, pages 283–287. Lawrence Erlbaum Associates, 1985.

[106] P.J. Gmytrasiewicz and E.H. Durfee. Rational coordination in multiagent environments through recursive modeling. *(submitted for publication)*, 1995.

[107] M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[108] R.L. Goldstone. The role of similarity in categorization: Providing a groundwork. *Cognition*, 52:125–157, 1994.

[109] R.L. Goldstone. Three tests of an alignment-based model of similarity. Technical Report 114, Department of Cognitive Science, Indiana University, 1994.

[110] K.P. Gross. Incremental multiple concept learning using experiments. In *Fifth International Conference on Machine Learning*, pages 65–72. Morgan Kaufmann, 1988.

[111] T.R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. Technical Report KSL–93–4, Knowledge Systems Laboratory, Stanford University, 1993.

[112] R. Gustavsson and S. Hägg. Societies of computation: A framework for computing & communication. University College of Karlskrona/Ronneby, Research Report 1/94, ISSN 1101–8453, 1994.

[113] S. Hägg and F. Ygge. Agent-oriented programming in power distribution automation. LUNFD6/(NFCS–3094/1–183/(1995), Department of Computer Science, Lund University, Sweden, 1995.

[114] S. Hanks and R.J. Firby. Issues and architectures for planning and execution. In *DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 59–70, San Mateo, CA, 1990. Morgan Kaufmann.

[115] S. Hanks, M. Pollack, and P. Cohen. Benchmarks, testbeds, controlled experimentation, and the design of agent architectures. Technical Report 93–06–05, Department of Computer Science and Engineering, University of Washington, 1993.

[116] S.J. Hanson and M. Bauer. Conceptual clustering, categorization, and polymorphy. *Machine Learning*, 3(4):343–372, 1989.

[117] S. Harnad. Category induction and representation. In S. Harnad, editor, *Categorical Perception*, pages 535–565. Cambridge University Press, 1987.

[118] S. Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

[119] J. Haugeland. *Artificial Intelligence – The Very Idea*. MIT Press, 1985.

[120] B. Hayes-Roth. An integrated architecture for intelligent agents. *SIGART*, 2(4):79–81, 1991.

[121] P.L. Heath. Concept. In P. Edwards, editor, *Encyclopedia of Philosophy*. Macmillan, 1967.

[122] H.H. Hexmoor, J.M. Lammens, and S.C. Shapiro. An autonomous agents architecture for integrating "unconscious" ans "conscious", reasoned behaviors. In *Computer Architectures for Computer Vision – 93*, 1993.

[123] R.C. Holte, L.E. Acker, and B.W. Porter. Concept learning and the problem of small disjuncts. In *IJCAI-89*, pages 813–818. Morgan Kaufmann, 1989.

[124] K.J. Holyoak and R.E. Nisbett. Induction. In R.J. Sternberg and E.E. Smith, editors, *The Psychology of Human Thought*. Cambridge University Press, 1988.

[125] V. Honavar. Inductive learning using generalized distance measures. In *SIPE Conference on Adaptive and Learning Systems*, Orlando, FL, 1992.

[126] V. Honavar. Toward learning systems that integrate different strategies and representations. Technical Report TR: 93–22, Department of Computer Science, Iowa State University of Science and Technology, IA, 1993. Also in: *Artificial Intelligence and Neural Networks: Steps toward principled integration*, V. Honavar and L. Uhr (ed), pages 615–644, Academic Press, 1994.

[127] A.E. Howe and P.R. Cohen. Failure recovery: A model and experiments. In *AAAI-91*, pages 801–808. MIT Press, 1991.

[128] E.B. Hunt, J. Marin, and P.J. Stone. *Experiments in Induction*. Academic Press, New York, 1966.

[129] A. Hutchinson. *Algorithmic Learning*. Clarendon Press, 1994.

[130] F.F. Ingrand, M.P. Georgeff, and A.S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6):34–44, 1992.

[131] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[132] C.G. Jansson. *Taxonomic Representation*. PhD thesis, The Royal Institute of Technology, Stockholm, Sweden, 1987.

[133] R.A. Johnson and D.W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice-Hall, 1992.

[134] P. Johnson-Laird. Procedural semantics. *Cognition*, 5:189–214, 1977.

[135] P. Johnson-Laird. *Mental Models*. Harvard University Press, 1983.

[136] L.P. Kaelbling. Goals as parallel program specifications. In *AAAI-88*, pages 60–65. Morgan Kaufmann, 1988.

[137] L. Kanal and T. Tsao. Artificial intelligence and natural perception. In L.O. Hertzberger and F.C.A. Groen, editors, *Intelligent Autonomous Systems*, pages 60–70. North-Holland, 1987.

[138] M. Kearns, M. Li, L. Pitt, and L.G. Valiant. Recent results on boolean concept learning. In *Fourth International Workshop on Machine Learning*, pages 337–352. Morgan Kaufmann, 1987.

[139] D. Kibler and D. Aha. Learning representative exemplars of concepts. In *Fourth International Workshop on Machine Learning*, pages 24–30. Morgan Kaufmann, 1987.

[140] D.N. Kinny and M.P. Georgeff. Commitment and effectiveness of situated agents. In *IJCAI-91*, pages 82–88. Morgan Kaufmann, 1991.

[141] D. Kirsh. Second-generation AI theories of learning. *Behavioral and Brain Sciences*, 9:658–659, 1986.

[142] D. Kirsh. Foundations of AI: The big issues. *Artificial Intelligence*, 47(1):3–30, 1991.

[143] D. Kirsh. Today the earwig, tomorrow man? *Artificial Intelligence*, 47(1):161–184, 1991.

[144] T. Kohonen. The “neural” phonetic typewriter. *IEEE Computer*, 27(3):11–22, 1988.

[145] L. Kohout. *A Perspective on Intelligent Systems*. Chapman and Hall Computing, 1990.

[146] L.K. Komatsu. Recent views of conceptual structure. *Psychological Bulletin*, 112(3):500–526, 1992.

[147] J.B. Kruskal and M. Wish. *Multidimensional Scaling*. Sage, 1978.

[148] D.R. Kuokka. MAX: A meta-reasoning architecture for “X”. *SIGART*, 2(4):93–97, 1991.

[149] G. Lakoff. *Women, Fire, and Dangerous Things: What categories reveal about the mind.* The University of Chicago Press, 1987.

[150] J.M. Lammens, H.H. Hexmoor, and S.C. Shapiro. Of elephants and men. In *NATO-ASI on the Biology and Technology of Intelligent Autonomous Agents*, Trento, Italy, 1993.

[151] P. Langley. Machine learning and grammar induction. *Machine Learning*, 2(1):5–8, 1987.

[152] M. Lebowitz. Concept learning in a rich input domain: Generalization-based memory. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An AI Approach, Volume II*, pages 193–214. Morgan Kaufmann, 1986.

[153] M. Lebowitz. Experiments with incremental concept formation: UNIMEM. *Machine Learning*, 2(2):103–138, 1987.

[154] M. Lebowitz. The utility of similarity-based learning in a world needing explanation. In R.S. Michalski and Y. Kodratoff, editors, *Machine Learning: An AI Approach, Volume III*, pages 399–422. Morgan Kaufmann, 1990.

[155] D.B. Lenat. *AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search*. PhD thesis, Stanford University, 1976.

[156] D.B. Lenat. Automated theory formation in mathematics. In *IJCAI-77*, pages 833–842. Morgan Kaufmann, 1977.

[157] D.B. Lenat and J.S. Brown. Why AM and Eurisko appear to work. *Artificial Intelligence*, 23(3):269–294, 1984.

[158] D.B. Lenat and R.V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley, 1990.

[159] R.C. Lou and M.G. Kay. Multisensor integration and fusion in intelligent systems. *IEEE Trans. on Systems, Man, and Cybernetics*, 19(5):901–931, 1989.

[160] D.M. Lyons and M.A. Arbib. A formal model of computation for sensory-based robotics. *IEEE Trans. on Robotics Automation*, 5(3):280–293, 1989.

[161] D.M. Lyons and A.J. Hendriks. Reactive planning. In S.C. Shapiro, editor, *Encyclopedia of Artificial Intelligence, 2nd edition*, pages 1171–1181. John Wiley and Sons, 1992.

[162] D.M. Lyons, A.J. Hendriks, and S. Mehta. Achieving robustness by casting planning as adaption of a reactive system. In *IEEE International Conference on Robotics and Automation*, 1991.

[163] R.L. Madarasz, L.C. Heiny, R.F. Cromp, and N.M. Mazur. The design of an autonomous vehicle for the disabled. *IEEE Journal of Robotics and Automation*, 2(3):117–126, 1986.

[164] P. Maes. The agent network architecture (ANA). *SIGART*, 2(4):115–120, 1991.

[165] P. Maes and D. Nardi. *Meta-Level Architectures and Reflection*. Elsevier Science, 1988.

[166] D. Marr. *Vision*. Freeman, 1982.

[167] E. Martensson. Improved coin classification, (Master's thesis). LU–CS–EX: 94–2, Department of Computer Science, Lund University, Sweden, 1994. (In Swedish).

[168] C.J. Matheus. Conceptual purpose: Implications for representation and learning in machines and humans. Technical Report UIUCDCS–R–87–1370, Department of Computer Science, University of Illinois at Urbana-Champaign, 1987.

[169] C.J. Matheus, L.R. Rendell, D.L. Medin, and R.L. Goldstone. Purpose and conceptual functions: A framework for concept representation and learning in humans and machines. In A. G. Cohn, editor, *Proc. of the Seventh Conference of the Society for the Study of Artificial Intelligence and Simulation of Behaviour*, pages 13–20. Pitman and Kaufmann, 1989.

[170] M.W. Matlin. Concept learning. In R.J. Corsini, editor, *Encyclopedia of Psychology*, pages 266–267. John Wiley & Sons, 1984.

[171] G. Matthews and J. Hearne. Clustering without a metric. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(2):175–184, 1991.

[172] F.G. McCabe and K.L. Clark. April – agent process interaction language. In M. Wooldridge and N.R. Jennings, editors, *Intelligent Agents — Theories, Architectures, and Languages*, pages 324–340. Springer Verlag, 1995.

[173] M. McCloskey and S. Glucksberg. Natural categories: Well defined or fuzzy-sets? *Memory and Cognition*, 6:462–472, 1978.

[174] D.L. Medin and M.M. Schaffer. Context theory of classification learning. *Psychological Review*, 85(3):207–238, 1978.

[175] D.L. Medin and E.E. Smith. Concepts and concept formation. *Annual Review of Psychology*, 35:113–138, 1984.

[176] R.S. Michalski. Pattern recognition as rule-guided inductive inference. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2(4):349–361, 1980.

[177] R.S. Michalski. Concept learning. In S.C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 185–194. John Wiley and Sons, 1987.

[178] R.S. Michalski. How to learn imprecise concepts: A method for employing a two-tiered knowledge representation in learning. In *Fourth International Workshop on Machine Learning*, pages 50–58. Morgan Kaufmann, 1987.

[179] R.S. Michalski and R.L. Chilausky. Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing expert systems for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems*, 4(2), 1980.

[180] R.S Michalski and Y. Kodratoff. Research in machine learning: Recent progress, classification of methods, and future directions. In Y. Kodratoff and R.S. Michalski, editors, *Machine Learning: An AI Approach, Volume III*, pages 3–30. Morgan Kaufmann, 1990.

[181] R.S. Michalski and J.B. Larson. Selection of most representative training examples and incremental generation of VL hypotheses: The underlying methodology and the description of programs ESEL and AQ11. Technical Report 877, Computer Science Department, University of Illinois, Urbana, 1978.

[182] R.S. Michalski and R.E. Stepp. Learning from observation: Conceptual clustering. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An AI Approach*, pages 331–363. Springer-Verlag, 1983.

[183] L. Miclet. *Structural Methods in Pattern Recognition*. North Oxford Academic, 1986.

[184] M. Minsky. A framework for representing knowledge. In P.H. Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, 1975.

[185] M. Minsky. *The Society of Mind*. Simon and Schuster, 1986.

[186] M. Minsky and S. Papert. *Perceptrons*. MIT Press, 1969.

[187] S. Minton. On modularity in integrated architectures. *SIGART*, 2(4):134–135, 1991.

[188] T.M. Mitchell. Version spaces: A candidate elimination approach to rule learning. In *IJCAI-77*, pages 305–310. Morgan Kaufmann, 1977.

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

[190] R. Mooney, J. Shavlik, G. Towell, and A. Gove. An experimental comparison of symbolic and connectionist learning algorithms. In *IJCAI-89*, pages 775–780. Morgan Kaufmann, 1989.

[191] S. Moscatelli and Y. Kodratoff. Advanced machine learning techniques for computer vision. In *Advanced Topics in Artificial Intelligence, Lecture Notes in Artificial Intelligence 617*, pages 161–197. Springer Verlag, 1992.

[192] S. Muggleton, editor. *Inductive Logic Programming.* Academic Press, 1992.

[193] H. Murase and S.K. Nayar. Learning object models from appearance. In *AAAI-93*, pages 836–843. MIT Press, 1993.

[194] G.L. Murphy and D.L. Medin. The role of theories in conceptual coherence. *Psychological Review*, 92(3):289–316, 1985.

[195] N. Muscettola, S.F. Smith, A. Cesta, and D. D'Aloisi. Coordinating space telescope operations in an integrated planning and scheduling architecture. *IEEE Control Systems Magazine*, 12(1):28–37, 1992.

[196] P.B. Musgrove and R.I. Phelps. An automatic system for acquisition of natural concepts. In *ECAI-90*, pages 455–460. Pitman, 1990.

[197] D.J. Nagel. Learning concepts with a prototype-based model for concept representation. In T.M. Mitchell, J.G. Carbonell, and R.S. Michalski, editors, *Machine Learning: A Guide to Current Research*, pages 233–236. Kluwer Academic Press, 1986.

[198] D.J. Nagel. *Learning Concepts with a Prototype-based Model for Concept Representation*. PhD thesis, Department of Computer Science, Rutgers, The State University of New Jersey, 1987.

[199] U. Neisser, editor. *Concepts and Conceptual Development: Ecological and intellectual factors in categorization*. Cambridge University Press, 1987.

[200] K. Nelson. Concept, word, and sentence: Interrelations in acquisition and development. *Psychological Review*, 81(4):267–285, 1974.

[201] A. Newell. *Unified Theories of Cognition*. Harvard University Press, 1990.

[202] N.J. Nilsson. A mobile automaton: An application of artificial intelligence techniques. In *IJCAI-69*, pages 509–520. Morgan Kaufmann, 1969.

[203] R.M. Nosofsky. Exemplar-based approach to relating categorization, identification, and recognition. In F.G. Ashby, editor, *Multidimensional Models of Perception and Cognition*. Lawrence Erlbaum Associates, 1992.

[204] D. Osherson, M. Stob, and S. Weinstein. *Systems that Learn*. A Bradford Book, MIT Press, 1986.

[205] P.W. Pachowicz. Integration of machine learning and vision into an active agent paradigm. In *AAAI Fall Symposium on Machine Learning and Computer Vision: What, Why, and How?, (FS–93–04)*. AAAI Press, 1993.

[206] K. Pahlavan. *Active Robot Vision and Primary Ocular Processes*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 1993.

[207] S.E. Palmer. Fundamental aspects of cognitive representation. In E. Rosch and B.B. Lloyd, editors, *Cognition and Categorization*. Erlbaum, 1978.

[208] R.S. Patil et al. The DARPA knowledge sharing effort: Progress report. In *Knowledge Representation and Reasoning (KR&R-92)*, pages 777–788. Morgan Kaufmann, 1992.

[209] L. Pitt. Introduction: Special issue on computational learning theory. *Machine Learning*, 5(2):117–120, 1990.

[210] L. Pitt and R.E. Reinke. Criteria for polynomial-time (conceptual) clustering. *Machine Learning*, 2(4):371–396, 1988.

[211] T. Poggio and S. Edelman. A network that learns to recognize three-dimensional objects. *Nature*, 343:263–266, 1990.

[212] M.E. Pollack and M. Ringuette. Introducing the Tileworld: Experimentally evaluating agent architectures. In *AAAI-90*. MIT Press, 1990.

[213] A.R. Pope and D.G. Lowe. Learning 3-d object recognition models from 2-d images. In *AAAI Fall Symposium on Machine Learning and Computer Vision: What, Why, and How?, (FS–93–04)*. AAAI Press, 1993.

[214] Z.W. Pylyshyn. *Computation and Cognition: Toward a Foundation for Cognitive Science*. MIT Press, 1984.

[215] M.R. Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing*, pages 227–270. MIT Press, 1968.

[216] W.V.O. Quine. *Ontological Relativity and other Essays*. Columbia University Press, 1969.

[217] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[218] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[219] L. Rendell. A new basis for state-space learning systems and a successful implementation. *Artificial Intelligence*, 20(4):369–392, 1983.

[220] L. Rendell. A general framework for induction and a study of selective induction. *Machine Learning*, 1(2):177–226, 1986.

[221] L. Rendell. Learning hard concepts. In *Third European Working Session on Learning*, pages 177–200, 1988.

[222] L. Rendell. Comparing systems and analyzing functions to improve constructive induction. In *Sixth International Workshop on Machine Learning*, pages 461–464. Morgan Kaufmann, 1989.

[223] G. Rey. Concepts and stereotypes. *Cognition*, 15:237–262, 1983.

[224] G. Rey. Concepts and conceptions: A reply to Smith, Medin and Rips. *Cognition*, 19:297–303, 1985.

[225] E. Rosch. Principles of categorization. In E. Rosch and B.B. Lloyd, editors, *Cognition and Categorization*, pages 28–49. Erlbaum, 1978.

[226] E. Rosch and C.B. Mervis. Family resemblances. studies in the internal structure of categories. *Cognitive Psychology*, 7:573–605, 1975.

[227] E. Rosch, C.B. Mervis, W. Gray, D. Johnson, and P. Boyes-Braem. Basic objects in natural categories. *Cognitive Psychology*, 8:382–439, 1976.

[228] R. Rosen. Planning, management, policies and strategies: Four fuzzy concepts. *International Journal of General Systems*, 1:245–252, 1974.

[229] R. Rosen. *Anticipatory Systems – Philosophical, Mathematical and Methodological Foundations*. Pergamon Press, 1985.

[230] S. Rosenschein and L. Kaelbling. The synthesis of digital machines with provable epistemic properties. In J.F. Halpern, editor, *The 1986 Conference on Theoretical Aspects of Reasoning and Knowledge*, 1987.

[231] Y. Roth-Tabak and R. Jain. Building an environment model using depth information. *IEEE Computer*, 22(6):85–90, 1989.

[232] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol.1: Foundations*. MIT Press, 1986.

[233] B. Russell. *The Problems of Philosophy*. Oxford University Press, 1912.

[234] E. Sacerdoti. The nonlinear nature of plans. In *IJCAI-75*, pages 205–214. Morgan Kaufmann, 1975.

[235] C. Schaffer. A conservation law for generalization performance. In *Eleventh International Conference on Machine Learning*, pages 259–265. Morgan Kaufmann, 1994.

[236] R.C. Schank and R.P. Abelson. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum, 1977.

[237] R.C. Schank, G.C. Collins, and L.E. Hunter. Transcending inductive category formation in learning. *Behavioral and Brain Sciences*, 9:639–686, 1986.

[238] J.C. Schlimmer and P. Langley. Machine learning. In S.C. Shapiro, editor, *Encyclopedia of Artificial Intelligence, 2nd edition,* pages 785–805. John Wiley and Sons, 1992.

[239] P.G. Schyns. A modular neural network model of concept acquisition. *Cognitive Science*, 15:461–508, 1991.

[240] K. Sengupta and K.L. Boyer. Information theoretic clustering of large structural modelbases. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 174–179, 1993.

[241] R.N. Shepard. A taxonomy of some principal types of data and of multidimensional methods for their analysis. In R.N. Shepard, A.K. Romney, and S.B. Nerlove, editors, *Multidimensional Scaling: Theory and Applications in the Behavioral Sciences, vol. I*, pages 21–47. Seminar Press, 1972.

[242] R.N. Shepard. Representation of structure in similarity data: Problems and prospects. *Psychometrika*, 39:373–421, 1974.

[243] M. Shneier. A compact relational structure representation. In *IJCAI-79*, pages 818–826. Morgan Kaufmann, 1979.

[244] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[245] S. Sjölander. Some cognitive break-throughs in the evolution of cognition and consciousness, and their impact on the biology of language. *Evolution and Cognition*, 3(1):1–10, 1993.

[246] A. Sloman. Why we need many knowledge representation formalisms. In M. Bramer, editor, *Research and Development in Expert Systems: Proc. of the BCS Expert Systems Conference 1984*. Cambridge University Press, 1985.

[247] A. Sloman. Reference without causal links. In J.B.H. du Boulay, D. Hogg, and L. Steels, editors, *Advances in Artificial Intelligence – II*, pages 369–381. North Holland, 1986.

[248] E.E. Smith. Category differences/automaticity. *Behavioral and Brain Sciences*, 9:667, 1986.

[249] E.E. Smith. Concepts and thought. In R.J. Sternberg and E.E. Smith, editors, *The Psychology of Human Thought*. Cambridge University Press, 1988.

[250] E.E. Smith. Concepts and induction. In M.L. Posner, editor, *Foundations of Cognitive Science*, pages 501–526. MIT Press, 1989.

[251] E.E. Smith and D.L. Medin. *Categories and Concepts*. Harvard University Press, 1981.

[252] E.E. Smith, D.L. Medin, and L.J. Rips. A psychological approach to concepts: Comments on Rey's "Concepts and stereotypes". *Cognition*, 17:265–274, 1984.

[253] P. Smolensky. On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1–74, 1988.

[254] P. Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2):159–216, 1990.

[255] P. Smyth and J. Mellstrom. Detecting novel classes with applications to fault diagnosis. In *Ninth International Workshop on Machine Learning*, pages 416–425. Morgan Kaufmann, 1992.

[256] R.J. Solomonoff. A formal theory of inductive inference. *Information and Control*, 7(1,2):1–22, 224–254, 1964. (Parts I and II).

[257] R.L. Solso. *Cognitive Psychology, third edition*. Allyn and Bacon, 1991.

[258] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

[259] R.E. Stepp. Concepts in conceptual clustering. In *IJCAI-87*, pages 211–213. Morgan Kaufmann, 1987.

[260] R.E. Stepp and R.S. Michalski. Conceptual clustering: Inventing goal-oriented classifications of structured objects. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An AI Approach, Volume II*, pages 471–498. Morgan Kaufmann, 1986.

[261] R.E. Stepp and R.S. Michalski. Conceptual clustering of structured objects: A goal-oriented approach. *Artificial Intelligence*, 28(1):43–69, 1986.

[262] R.S. Sutton. First results with Dyna, an integrated architecture for learning, planning and reacting. In W.T. Miller, R.S. Sutton, and P.J. Werbos, editors, *Neural Networks for Control*, pages 179–189. MIT Press, 1990.

[263] S. Tachi and K. Komoriya. Guide dog robot. In S.S. Iyengar and A. Elfes, editors, *Autonomous Mobile Robots: Control, Planning and Architecture*, pages 360–367. IEEE Computer Society Press, 1991.

[264] R. Taraban. Introduction: A coupling of disciplines in categorization research. *The Psychology of Learning and Motivation*, 29:1–12, 1993.

[265] R. Taraban and J.M. Palacios. Exemplar models and weighted cue models in category learning. *The Psychology of Learning and Motivation*, 29:91–128, 1993.

[266] A. Tarski. *Logic, Semantics, Metamathematics*. Oxford, 1956. second edition, J. Corcoran (ed.), Hacklett, 1983.

[267] K. Thompson and P. Langley. Incremental concept formation with composite objects. In *Sixth International Workshop on Machine Learning*, pages 371–374. Morgan Kaufmann, 1989.

[268] K. Thompson and P. Langley. Concept formation in structured domains. In D.H. Fischer, M.J. Pazzani, and P. Langley, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, pages 127–161. Morgan Kaufmann, 1991.

[269] G.G. Towell, J.W. Shavlik, and M.O. Noordewier. Refinement of approximate domain theories by knowledge-based neural networks. In *AAAI-90*, pages 861–866. MIT Press, 1990.

[270] M.M. Trivedi, C. Chen, and S.B. Marapane. A vision system for robotic inspection and manipulation. *IEEE Computer*, 22(6):91–97, 1989.

[271] J.K. Tsotsos. Behaviorist intelligence and the scaling problem. *Artificial Intelligence*, 75(2):135–160, 1995.

[272] L.H. Tsoukalas. *Anticipatory Systems Using a Probabilistic-Possibilistic Formalism*. PhD thesis, Department of Nuclear Engineering, University of Illinois at Urbana-Champaign, 1989.

[273] A. Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977.

[274] S. Ullman. Visual routines. *Cognition*, 18:97–106, 1984.

[275] P.E. Utgoff. Shift of bias for inductive concept learning. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An AI Approach, Volume II*. Morgan Kaufmann, 1986.

[276] L.M. Vaina and M-C. Jaulent. Object structure and action requirements: A compatibility model for functional recognition. *International Journal of Intelligent Systems*, 6:313–336, 1991.

[277] L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[278] S. Watanabe. *Knowing and Guessing: A Quantitative Study of Inference and Information*. John Wiley and Sons, 1969.

[279] G.I. Webb. Generality is more significant then complexity: Toward an alternative to Occam's razor. In *Seventh Australian Joint Conference on AI*, pages 60–67. World Scientific, 1994.

[280] S.M. Weiss and I. Kapouleas. An empirical comparison of pattern recognition, neural nets and machine learning classification methods. In *IJCAI-89*, pages 781–787. Morgan Kaufmann, 1989.

[281] S.M. Weiss and C.A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann, 1991.

[282] P.H. Winston. Learning structural descriptions from examples. In P.H. Winston, editor, *The Psychology of Computer Vision*, pages 157–209. McGraw-Hill, 1975.

[283] P.H. Winston. Learning new principles from precedents and exercises. *Artificial Intelligence*, 19(3):321–350, 1982.

[284] E.J. Wisniewski and D.L. Medin. The fiction and nonfiction of features. In R. Michalski and G. Tecuci, editors, *Machine Learning: A Multistrategy Approach, Volume IV*, pages 63–84. Morgan Kaufmann, 1994.

[285] L. Wittgenstein. *Tractatus Logico-Philosophicus*. Routledge and Kegan Paul, 1922.

[286] L. Wittgenstein. *Philosophical Investigations*. Basil Blackwell, 1953.

[287] M.J. Wooldridge. *The Logical Modelling of Computational Multi-Agent Systems*. PhD thesis, Department of Computation, University of Manchester, 1992.

[288] M.J. Wooldridge and N.R. Jennings. Agent theories, architectures, and languages: A survey. In M. Wooldridge and N.R. Jennings, editors, *Intelligent Agents — Theories, Architectures, and Languages, Lecture Notes in Artificial Intelligence 890*, pages 1–39. Springer Verlag, 1995.

[289] M.J. Wooldridge and N.R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[290] A.D. Woozley. Universals. In P. Edwards, editor, *Encyclopedia of Philosophy*. Macmillan, 1967.

[291] S. Wrobel. Towards a model of grounded concept formation. In *IJCAI-91*, pages 712–717. Morgan Kaufmann, 1991.

[292] S. Wrobel. *Concept Formation and Knowledge Revision*. Kluwer Academic Publishers, 1994.

[293] B.P. Zeigler. *Object-Oriented Simulation with Hierarchical, Modular Models – Intelligent Agents and Endomorphic Systems*. Academic Press, 1990.

[294] S. Zilberstein and S.J. Russell. Anytime sensing, planning and action: A practical model for robot control. In *IJCAI-93*, pages 1402–1407. Morgan Kaufmann, 1993.

[295] G. Zlotkin and J.S. Rosenschein. Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domain. In *AAAI-94*, pages 432–437. MIT Press, 1994.

Author Index

Subject Index