Undergraduate Topics in Computer Science

Wolfgang Ertel

Introduction to Artificial Intelligence

Second Edition

UTiCS

Springer

Wolfgang Ertel

Introduction to Artificial Intelligence

Second Edition

Translated by Nathanael Black
With illustrations by Florian Mast

Springer

Wolfgang Ertel
Hochschule Ravensburg-Weingarten
Weingarten
Germany

ISSN 1863-7310 ISSN 2197-1781 (electronic)
Undergraduate Topics in Computer Science
ISBN 978-3-319-58486-7 ISBN 978-3-319-58487-4 (eBook)
DOI 10.1007/978-3-319-58487-4

Library of Congress Control Number: 2017943187

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Contents

Introduction 1

1.1 What Is Artificial Intelligence?

The term *artificial intelligence* stirs emotions. For one thing there is our fascination with *intelligence*, which seemingly imparts to us humans a special place among life forms. Questions arise such as "*What is intelligence*?", "*How can one measure intelligence*?" or "*How does the brain work*?". All these questions are meaningful when trying to understand artificial intelligence. However, the central question for the engineer, especially for the computer scientist, is the question of the intelligent machine that behaves like a person, showing intelligent behavior.

The attribute *artificial* might awaken much different associations. It brings up fears of intelligent cyborgs. It recalls images from science fiction novels. It raises the question of whether our highest good, the soul, is something we should try to understand, model, or even reconstruct.

With such different offhand interpretations, it becomes difficult to define the term *artificial intelligence* or *AI* simply and robustly. Nevertheless I would like to try, using examples and historical definitions, to characterize the field of AI. In 1955, John McCarthy, one of the pioneers of AI, was the first to define the term *artificial intelligence*, roughly as follows:

> The goal of AI is to develop machines that behave as though they were intelligent.

To test this definition, the reader might imagine the following scenario. Fifteen or so small robotic vehicles are moving on an enclosed four by four meter square surface. One can observe various behavior patterns. Some vehicles form small groups with relatively little movement. Others move peacefully through the space and gracefully avoid any collision. Still others appear to follow a leader. Aggressive behaviors are also observable. Is what we are seeing intelligent behavior?

According to McCarthy's definition the aforementioned robots can be described as intelligent. The psychologist Valentin Braitenberg has shown that this seemingly

W. Ertel, *Introduction to Artificial Intelligence*, Undergraduate Topics in Computer Science, DOI 10.1007/978-3-319-58487-4_1

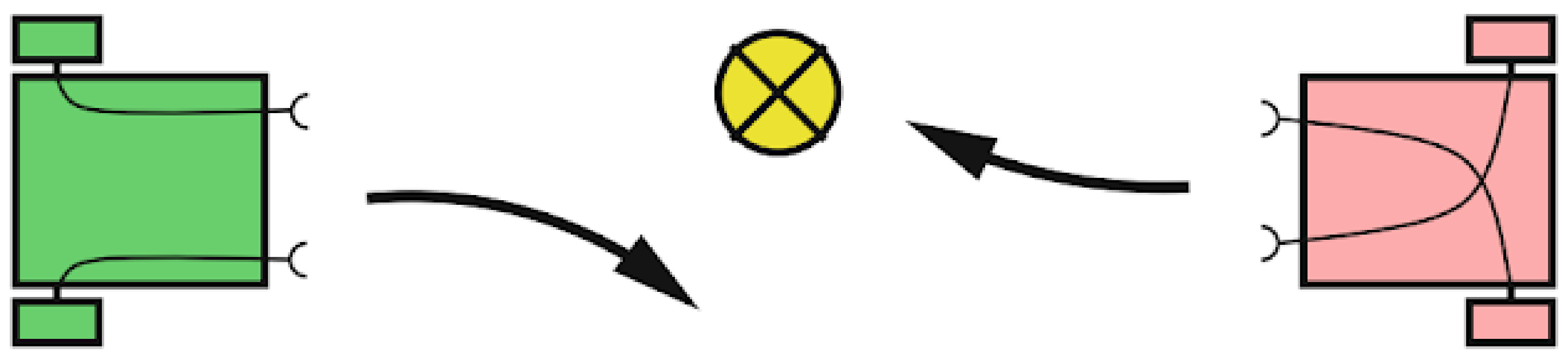

Fig. 1.1 Two very simple Braitenberg vehicles and their reactions to a light source

complex behavior can be produced by very simple electrical circuits [Bra84]. So-called Braitenberg vehicles have two wheels, each of which is driven by an independent electric motor. The speed of each motor is influenced by a light sensor on the front of the vehicle as shown in Fig. 1.1. The more light that hits the sensor, the faster the motor runs. Vehicle 1 in the left part of the figure, according to its configuration, moves away from a point light source. Vehicle 2 on the other hand moves toward the light source. Further small modifications can create other behavior patterns, such that with these very simple vehicles we can realize the impressive behavior described above.

Clearly the above definition is insufficient because AI has the goal of solving difficult practical problems which are surely too demanding for the Braitenberg vehicle. In the Encyclopedia Britannica [Bri91] one finds a Definition that goes like:

> AI is the ability of digital computers or computer controlled robots to solve problems that are normally associated with the higher intellectual processing capabilities of humans …

But this definition also has weaknesses. It would admit for example that a computer with large memory that can save a long text and retrieve it on demand displays intelligent capabilities, for memorization of long texts can certainly be considered a *higher intellectual processing capability* of humans, as can for example the quick multiplication of two 20-digit numbers. According to this definition, then, every computer is an AI system. This dilemma is solved elegantly by the following definition by Elaine Rich [Ric83]:

> Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better.

Rich, tersely and concisely, characterizes what AI researchers have been doing for the last 50 years. Even in the year 2050, this definition will be up to date.

Tasks such as the execution of many computations in a short amount of time are the strong points of digital computers. In this regard they outperform humans by many multiples. In many other areas, however, humans are far superior to machines. For instance, a person entering an unfamiliar room will recognize the surroundings within fractions of a second and, if necessary, just as swiftly make decisions and plan actions. To date, this task is too demanding for autonomous[1]

[1]An autonomous robot works independently, without manual support, in particular without remote control.

robots. According to Rich's definition, this is therefore a task for AI. In fact, research on autonomous robots is an important, current theme in AI. Construction of chess computers, on the other hand, has lost relevance because they already play at or above the level of grandmasters.

It would be dangerous, however, to conclude from Rich's definition that AI is only concerned with the pragmatic implementation of intelligent processes. Intelligent systems, in the sense of Rich's definition, cannot be built without a deep understanding of human reasoning and intelligent action in general, because of which neuroscience (see Sect. 1.1.1) is of great importance to AI. This also shows that the other cited definitions reflect important aspects of AI.

A particular strength of human intelligence is adaptivity. We are capable of adjusting to various environmental conditions and change our behavior accordingly through *learning*. Precisely because our learning ability is so vastly superior to that of computers, *machine learning* is, according to Rich's definition, a central subfield of AI.

1.1.1 Brain Science and Problem Solving

Through research of intelligent systems we can try to understand how the human brain works and then model or simulate it on the computer. Many ideas and principles in the field of neural networks (see Chap. 9) stem from brain science with the related field of neuroscience.

A very different approach results from taking a goal-oriented line of action, starting from a problem and trying to find the most optimal solution. How humans solve the problem is treated as unimportant here. The method, in this approach, is secondary. First and foremost is the optimal intelligent solution to the problem. Rather than employing a fixed method (such as, for example, predicate logic) AI has as its constant goal the creation of intelligent agents for as many different tasks as possible. Because the tasks may be very different, it is unsurprising that the methods currently employed in AI are often also quite different. Similar to medicine, which encompasses many different, often life-saving diagnostic and therapy procedures, AI also offers a broad palette of effective solutions for widely varying applications. For mental inspiration, consider Fig. 1.2 on page 4. Just as in medicine, there is no universal method for all application areas of AI, rather a great number of possible solutions for the great number of various everyday problems, big and small.

Cognitive science is devoted to research into human thinking at a somewhat higher level. Similarly to brain science, this field furnishes practical AI with many important ideas. On the other hand, algorithms and implementations lead to further important conclusions about how human reasoning functions. Thus these three fields benefit from a fruitful interdisciplinary exchange. The subject of this book, however, is primarily problem-oriented AI as a subdiscipline of computer science.

There are many interesting philosophical questions surrounding intelligence and artificial intelligence. We humans have consciousness; that is, we can think about

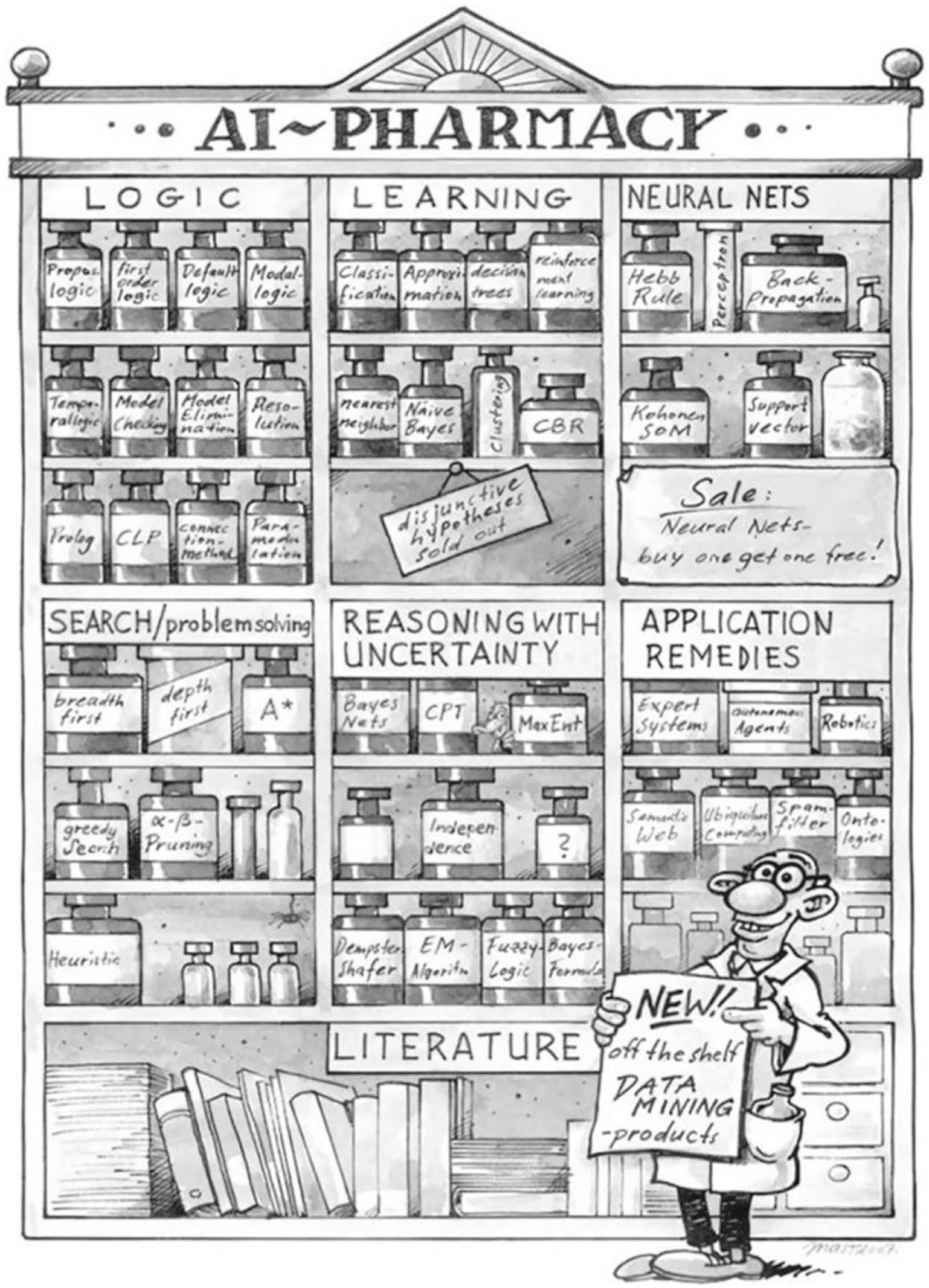

Fig. 1.2 A small sample of the solutions offered by AI

ourselves and even ponder that we are able to think about ourselves. How does consciousness come to be? Many philosophers and neurologists now believe that the mind and consciousness are linked with matter, that is, with the brain. The question of whether machines could one day have a mind or consciousness could at some point in the future become relevant. The mind-body problem in particular concerns whether or not the mind is bound to the body. We will not discuss these questions here. The interested reader may consult [Spe98, Spe97] and is invited, in the course of AI technology studies, to form a personal opinion about these questions.

1.1.2 The Turing Test and Chatterbots

Alan Turing made a name for himself as an early pioneer of AI with his definition of an intelligent machine, in which the machine in question must pass the following test. The test person Alice sits in a locked room with two computer terminals. One terminal is connected to a machine, the other with a non-malicious person Bob. Alice can type questions into both terminals. She is given the task of deciding, after five minutes, which terminal belongs to the machine. The machine passes the test if it can trick Alice at least 30% of the time [Tur50].

While the test is very interesting philosophically, for practical AI, which deals with problem solving, it is not a very relevant test. The reasons for this are similar to those mentioned above related to Braitenberg vehicles (see Exercise 1.3 on page 21).

The AI pioneer and social critic Joseph Weizenbaum developed a program named *Eliza*, which is meant to answer a test subject's questions like a human psychologist [Wei66]. He was in fact able to demonstrate success in many cases. Supposedly his secretary often had long discussions with the program. Today in the internet there are many so-called *chatterbots*, some of whose initial responses are quite impressive. After a certain amount of time, however, their artificial nature becomes apparent. Some of these programs are actually capable of learning, while others possess extraordinary knowledge of various subjects, for example geography or software development. There are already commercial applications for chatterbots in online customer support and there may be others in the field of e-learning. It is conceivable that the learner and the e-learning system could communicate through a chatterbot. The reader may wish to compare several chatterbots and evaluate their intelligence in Exercise 1.1 on page 20.

1.2 The History of AI

AI draws upon many past scientific achievements which are not mentioned here, for AI as a science in its own right has only existed since the middle of the Twentieth Century. Table 1.1 on page 6, with the most important AI milestones, and a graphical representation of the main movements of AI in Fig. 1.3 on page 8 complement the following text.

Table 1.1 Milestones in the development of AI from Gödel to today

1931	The Austrian Kurt Gödel shows that in first-order *predicate logic* all true statements are derivable [Göd31a]. In higher-order logics, on the other hand, there are true statements that are unprovable [Göd31b]. (In [Göd31b] Gödel showed that predicate logic extended with the axioms of arithmetic is incomplete.)
1937	Alan Turing points out the limits of intelligent machines with the halting problem [Tur37].
1943	McCulloch and Pitts model *neural networks* and make the connection to propositional logic.
1950	Alan Turing defines machine intelligence with the *Turing test* and writes about learning machines and genetic algorithms [Tur50].
1951	Marvin Minsky develops a neural network machine. With 3000 vacuum tubes he simulates 40 neurons.
1955	Arthur Samuel (IBM) builds a learning checkers program that plays better than its developer [Sam59].
1956	McCarthy organizes a conference in Dartmouth College. Here the name *Artificial Intelligence* was first introduced.
	Newell and Simon of Carnegie Mellon University (CMU) present the *Logic Theorist*, the first symbol-processing computer program [NSS83].
1958	McCarthy invents at MIT (Massachusetts Institute of Technology) the high-level language *LISP*. He writes programs that are capable of modifying themselves.
1959	Gelernter (IBM) builds the Geometry Theorem Prover.
1961	The General Problem Solver (GPS) by Newell and Simon imitates human thought [NS61].
1963	McCarthy founds the AI Lab at Stanford University.
1965	Robinson invents the *resolution calculus* for predicate logic [Rob65] (Sect. 3.5).
1966	Weizenbaum's program Eliza carries out dialog with people in natural language [Wei66] (Sect. 1.1.2).
1969	Minsky and Papert show in their book *Perceptrons* that the perceptron, a very simple neural network, can only represent linear functions [MP69] (Sect. 1.1.2).
1972	French scientist Alain Colmerauer invents the logic programming language *PROLOG* (Chap. 5).
	British physician de Dombal develops an *expert system* for diagnosis of acute abdominal pain [dDLS+72]. It goes unnoticed in the mainstream AI community of the time (Sect. 7.3).
1976	Shortliffe and Buchanan develop MYCIN, an expert system for diagnosis of infectious diseases, which is capable of dealing with uncertainty (Chap. 7).
1981	Japan begins, at great expense, the "Fifth Generation Project" with the goal of building a powerful PROLOG machine.
1982	R1, the expert system for configuring computers, saves Digital Equipment Corporation 40 million dollars per year [McD82].
1986	Renaissance of neural networks through, among others, Rumelhart, Hinton and Sejnowski [RM86]. The system Nettalk learns to read texts aloud [SR86] (Chap. 9).
1990	Pearl [Pea88], Cheeseman [Che85], Whittaker, Spiegelhalter bring probability theory into AI with *Bayesian networks* (Sect. 7.4). Multi-agent systems become popular.

(continued)

Table 1.1 (continued)

1992	Tesauros TD-gammon program demonstrates the advantages of reinforcement learning.
1993	Worldwide *RoboCup* initiative to build soccer-playing autonomous robots [Roba].
1995	From statistical learning theory, Vapnik develops support vector machines, which are very important today.
1997	IBM's chess computer Deep Blue defeats the chess world champion Gary Kasparov.
	First international RoboCup competition in Japan.
2003	The robots in RoboCup demonstrate impressively what AI and robotics are capable of achieving.
2006	Service robotics becomes a major AI research area.
2009	First Google self-driving car drives on the California freeway.
2010	Autonomous robots begin to improve their behavior through learning.
2011	IBM's "Watson" beats two human champions on the television game show "Jeopardy!". Watson understands natural language and can answer difficult questions very quickly (Sect. 1.4).
2015	Daimler premiers the first autonomous truck on the Autobahn.
	Google self-driving cars have driven over one million miles and operate within cities.
	Deep learning (Sect. 11.9) enables very good image classification.
	Paintings in the style of the Old Masters can be automatically generated with deep learning. AI becomes creative!
2016	The Go program AlphaGo by Google DeepMind [SHM+16] beats the European champion 5:0 in January and Korean Lee Sedol, one of the world's best Go players, 4:1 in March. Deep learning techniques applied to pattern recognition, as well as reinforcement learning and Monte Carlo tree search lead to this success.

1.2.1 The First Beginnings

In the 1930s Kurt Gödel, Alonso Church, and Alan Turing laid important foundations for logic and theoretical computer science. Of particular interest for AI are Gödel's theorems. The completeness theorem states that first-order predicate logic is complete. This means that every true statement that can be formulated in predicate logic is provable using the rules of a formal calculus. On this basis, automatic theorem provers could later be constructed as implementations of formal calculi. With the incompleteness theorem, Gödel showed that in higher-order logics there exist true statements that are unprovable.[2] With this he uncovered painful limits of formal systems.

Alan Turing's proof of the undecidability of the halting problem also falls into this time period. He showed that there is no program that can decide whether a given arbitrary program (and its respective input) will run in an infinite loop. With

[2]Higher-order logics are extensions of predicate logic, in which not only variables, but also function symbols or predicates can appear as terms in a quantification. Indeed, Gödel only showed that any system that is based on predicate logic and can formulate Peano arithmetic is incomplete.

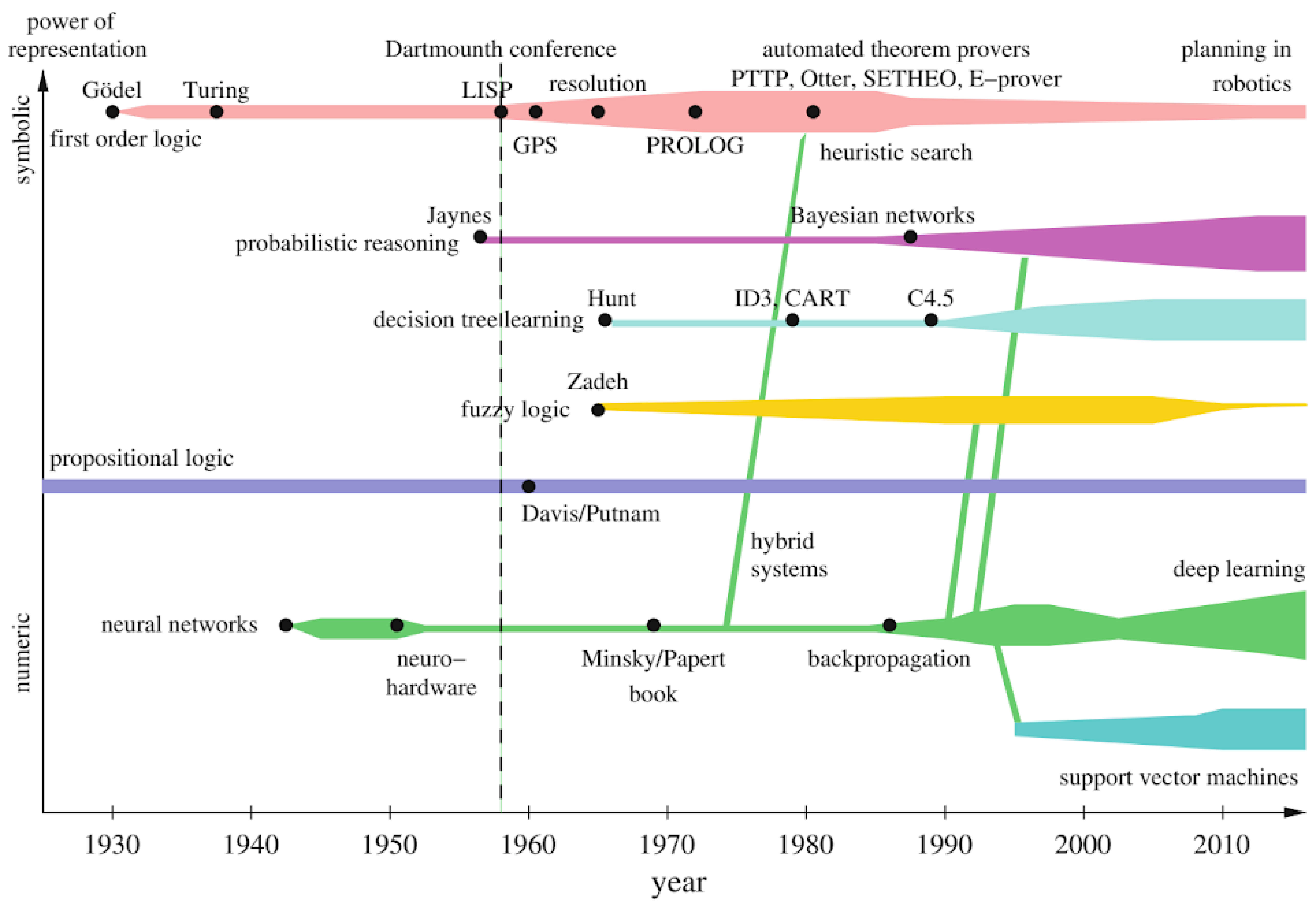

Fig. 1.3 History of the various AI areas. The width of the *bars* indicates prevalence of the method's use

this Turing also identified a limit for intelligent programs. It follows, for example, that there will never be a universal program verification system.[3]

In the 1940s, based on results from neuroscience, McCulloch, Pitts and Hebb designed the first mathematical models of neural networks. However, computers at that time lacked sufficient power to simulate simple brains.

1.2.2 Logic Solves (Almost) All Problems

AI as a practical science of thought mechanization could of course only begin once there were programmable computers. This was the case in the 1950s. Newell and Simon introduced Logic Theorist, the first automatic theorem prover, and thus also showed that with computers, which actually only work with numbers, one can also process symbols. At the same time McCarthy introduced, with the language LISP, a programming language specially created for the processing of symbolic structures. Both of these systems were introduced in 1956 at the historic Dartmouth Conference, which is considered the birthday of AI.

In the US, LISP developed into the most important tool for the implementation of symbol-processing AI systems. Thereafter the logical inference rule known as resolution developed into a complete calculus for predicate logic.

[3]This statement applies to "total correctness", which implies a proof of correct execution as well as a proof of termination for every valid input.

In the 1970s the logic programming language PROLOG was introduced as the European counterpart to LISP. PROLOG offers the advantage of allowing direct programming using Horn clauses, a subset of predicate logic. Like LISP, PROLOG has data types for convenient processing of lists.

Until well into the 1980s, a breakthrough spirit dominated AI, especially among many logicians. The reason for this was the string of impressive achievements in symbol processing. With the Fifth Generation Computer Systems project in Japan and the ESPRIT program in Europe, heavy investment went into the construction of intelligent computers.

For small problems, automatic provers and other symbol-processing systems sometimes worked very well. The combinatorial explosion of the search space, however, defined a very narrow window for these successes. This phase of AI was described in [RN10] as the “Look, Ma, no hands!” era.

Because the economic success of AI systems fell short of expectations, funding for logic-based AI research in the United States fell dramatically during the 1980s.

1.2.3 The New Connectionism

During this phase of disillusionment, computer scientists, physicists, and Cognitive scientists were able to show, using computers which were now sufficiently powerful, that mathematically modeled neural networks are capable of learning using training examples, to perform tasks which previously required costly programming. Because of the fault-tolerance of such systems and their ability to recognize patterns, considerable successes became possible, especially in pattern recognition. Facial recognition in photos and handwriting recognition are two example applications. The system Nettalk was able to learn speech from example texts [SR86]. Under the name *connectionism*, a new subdiscipline of AI was born.

Connectionism boomed and the subsidies flowed. But soon even here feasibility limits became obvious. The neural networks could acquire impressive capabilities, but it was usually not possible to capture the learned concept in simple formulas or logical rules. Attempts to combine neural nets with logical rules or the knowledge of human experts met with great difficulties. Additionally, no satisfactory solution to the structuring and modularization of the networks was found.

1.2.4 Reasoning Under Uncertainty

AI as a practical, goal-driven science searched for a way out of this crisis. One wished to unite logic’s ability to explicitly represent knowledge with neural networks’ strength in handling uncertainty. Several alternatives were suggested.

The most promising, *probabilistic reasoning*, works with conditional probabilities for propositional calculus formulas. Since then many diagnostic and expert systems have been built for problems of everyday reasoning using *Bayesian* *networks*. The success of Bayesian networks stems from their intuitive comprehensibility, the clean semantics of conditional probability, and from the centuries-old, mathematically grounded probability theory.

The weaknesses of logic, which can only work with two truth values, can be solved by *fuzzy logic*, which pragmatically introduces infinitely many values between zero and one. Though even today its theoretical foundation is not totally firm, it is being successfully utilized, especially in control engineering.

A much different path led to the successful synthesis of logic and neural networks under the name *hybrid systems*. For example, neural networks were employed to learn heuristics for reduction of the huge combinatorial search space in proof discovery [SE90].

Methods of decision tree learning from data also work with probabilities. Systems like CART, ID3 and C4.5 can quickly and automatically build very accurate decision trees which can represent propositional logic concepts and then be used as expert systems. Today they are a favorite among machine learning techniques (Sect. 8.4).

Since about 1990, *data mining* has developed as a subdiscipline of AI in the area of statistical data analysis for extraction of knowledge from large databases. Data mining brings no new techniques to AI, rather it introduces the requirement of using large databases to gain explicit knowledge. One application with great market potential is steering ad campaigns of big businesses based on analysis of many millions of purchases by their customers. Typically, machine learning techniques such as decision tree learning come into play here.

1.2.5 Distributed, Autonomous and Learning Agents

Distributed artificial intelligence, DAI, has been an active area research since about 1985. One of its goals is the use of parallel computers to increase the efficiency of problem solvers. It turned out, however, that because of the high computational complexity of most problems, the use of "intelligent" systems is more beneficial than parallelization itself.

A very different conceptual approach results from the development of autonomous software agents and robots that are meant to cooperate like human teams. As with the aforementioned Braitenberg vehicles, there are many cases in which an individual agent is not capable of solving a problem, even with unlimited resources. Only the cooperation of many agents leads to the intelligent behavior or to the solution of a problem. An ant colony or a termite colony is capable of erecting buildings of very high architectural complexity, despite the fact that no single ant comprehends how the whole thing fits together. This is similar to the situation of provisioning bread for a large city like New York [RN10]. There is no central planning agency for bread, rather there are hundreds of bakers that know their respective areas of the city and bake the appropriate amount of bread at those locations.

Active skill acquisition by robots is an exciting area of current research. There are robots today, for example, that independently learn to walk or to perform various motorskills related to soccer (Chap. 10). Cooperative learning of multiple robots to solve problems together is still in its infancy.

1.2.6 AI Grows Up

The above systems offered by AI today are not a universal recipe, but a workshop with a manageable number of tools for very different tasks. Most of these tools are well-developed and are available as finished software libraries, often with convenient user interfaces. The selection of the right tool and its sensible use in each individual case is left to the AI developer or knowledge engineer. Like any other artisanship, this requires a solid education, which this book is meant to promote.

More than nearly any other science, AI is interdisciplinary, for it draws upon interesting discoveries from such diverse fields as logic, operations research, statistics, control engineering, image processing, linguistics, philosophy, psychology, and neurobiology. On top of that, there is the subject area of the particular application. To successfully develop an AI project is therefore not always so simple, but almost always extremely exciting.

1.2.7 The AI Revolution

Around the year 2010 after about 25 years of research on neural networks, scientists could start harvesting the fruits of their research. The very powerful *deep learning networks* can for example learn to classify images with very high arruracy. Since image classification is of crucial importance for all types of smart robots, this initiated the *AI revolution* which in turn leads to smart self-driving cars and service robots.

1.3 AI and Society

There have been many scientific books and science fiction novels written on all aspects of this subject. Due to great advances in AI research, we have been on the brink of the age of autonomous robots and the Internet of Things since roughly 2005. Thus we are increasingly confronted with AI in everyday life. The reader, who may soon be working as an AI developer, must also deal with the social impact of this work. As an author of a book on AI techniques, I have the crucial task of examining this topic. I would like to deal with some particularly important aspects of AI which are of great practical relevance for our lives.

1.3.1 Does AI Destroy Jobs?

In January 2016, the World Econonic Forum published a study [SS16], frequently cited by the German press, predicting that "industry 4.0 " would destroy over five million jobs in the next five years. This forecast is hardly surprising because automation in factories, offices, administration, transportation, in the home and in many other areas has led to continually more work being done by computers, machines and robots. AI has been one of the most important factors in this trend since about 2010.

Presumably, the majority of people would gladly leave physically hard, dirty and unhealthy jobs and tasks to machines. Thus automation is a complete blessing for humanity, assuming it does not result in negative side effects, such as harm to the environment. Many of the aforementioned unpleasant jobs can be done faster, more precisely, and above all cheaper by machines. This seems almost like a trend towards paradise on Earth, where human beings do less and less unpleasant work and have correspondingly more time for the good things in life. This seems almost like a trend towards paradise on earth. We have to do less and less unpleasant work and in turn have more time for the good things in life.[4] All the while, we would enjoy the same (or potentially even increasing) prosperity, for the economy would not employ these machines if they did not markedly raise productivity.

Unfortunately we are not on the road to paradise. For several decades, we have worked more than 40 hours per week, have been stressed, complained of burnout and other sicknesses, and suffered a decline in real wages. How can this be, if productivity is continually increasing? Many economists say that the reason for this is competitive pressure. In an effort to compete and deliver the lowest priced goods to market, companies need to lower production costs and thus lay off workers. This results in the aforementioned unemployment. In order to avoid a drop in sales volume due to reduced prices, more products need to be manufactured and sold. The economy must grow!

If the economy continues to grow in a country in which the population is no longer growing (as is the case in most modern industrialized countries), each citizen must necessarily consume more. For that to happen, new markets must be created,[5] and marketing has the task of convincing us that we want the new products. This is—allegedly—the only way to "sustainably" ensure prosperity. Apparently there seems to be no escape from this growth/consumption spiral. This has two fatal consequences. For one thing, this increase in consumption should make people happier, but it is having quite the opposite effect: mental illness is increasing.

Even more obvious and, above all, fatal, are economic growth's effects on our living conditions. It is no secret that the earth's growth limit has long been exceeded [MMZM72, Ran12], and that we are overexploiting nature's nonrenewable resources. We are therefore living at the expense of our children and grandchildren, who consequently will have poorer living conditions than we have today. It is also known that every additional dollar of economic growth is an additional burden on the environment—for example through additional CO_2 concentration in the atmosphere and the resulting climate change [Pae16]. We are destroying our own basis of

[4]Those of us, such as scientists, computer scientists and engineers, who enjoy it may of course continue our work.

[5]Many EU and German Ministry of Education and Research funding programs for example require that scientists who submit proposals show evidence that their research will open up new markets.

existence. Thus it is obvious that we should abandon this path of growth for the sake of a livable future. But how?

Let's think back to the road to paradise that AI is supposedly preparing for us. Apparently, as we practice it, it does not lead to paradise. Understanding this problem and finding the right path is one of the central tasks of today. Because of inherent complexities, this problem can not be fully dealt with in an introductory AI textbook. However, I would like to provide the reader with a little food for thought.

Although productivity is growing steadily in almost all areas of the economy, workers are required to work as hard as ever. They do not benefit from the increase in productivity. So, we must ask, where do the profits go? Evidently not to the people to whom they are owed, i.e. the workers. Instead, part of the profits is spent on investment and thus on further growth and the rest is taken by the capital owners, while employees work the same hours for declining real wages [Pik14]. This leads to ever-increasing capital concentration among a few rich individuals and private banks, while on the other hand increasing poverty around the world is creating political tensions that result in war, expulsion and flight.

What is missing is a fair and just distribution of profits. How can this be achieved? Politicians and economists are continually trying to optimize our economic system, but politics has not offered a sustainable solution, and too few economists are investigating this highly exciting economic question. Obviously the attempt to optimize the parameters of our current capitalist economic system has not lead to a more equitable distribution of wealth, but to the opposite.

This is why economists and financial scientists must begin to question the system and look for alternatives. We should ask ourselves how to change the rules and laws of the economy so that all people profit from increased productivity. A growing community of economists and sustainability scientists have offered interesting solutions, a few of which I will briefly describe here.

Problem Number One is the creation of fiat money by the banks. New money—which is needed, among other things, to keep our growing economy going—is now being created by private banks. This is made possible by the fact that banks have to own only a small part, namely the minimum cash reserve ratio, of the money they give as loans. In the EU in 2016, the minimum cash reserve ratio is one percent.

States then borrow this money from private banks in the form of government bonds and thus fall into debt. This is how our current government debt crises have developed. This problem can be solved easily by prohibiting creation of money by the banks by increasing the minimum cash reserve ratio to 100%. State central banks will then get back the monopoly on creating money, and the newly created money can be used directly by the state for the purposes of social welfare. It should be evident that this simple measure would significantly ease the problem of public debt.

Further interesting components of such an economic reform could be the conversion of the current interest rate system to the so-called natural economic order [GP58], and the introduction of the "economy for the common good" [Fel14] and the biophysical economy [GK09, Küm11]. The practical implementation of the economy for the common good would involve a tax reform, the most important elements of which would be the abolition of the income tax and substantially increased value added tax on energy and resource consumption. We would thus arrive at a highly prosperous, more sustainable human world with less environmental damage and more local trade. The reader may study the literature and assess whether the ideas quoted here are interesting and, if necessary, help to make the required changes.

To conclude this section, I would like to quote the famous physicist Stephen Hawking. In a community-driven interview on www.reddit.com he gave the following answer to whether he had any thoughts about unemployment caused by automation:

> If machines produce everything we need, the outcome will depend on how things are distributed. **Everyone can enjoy a life of luxurious leisure if the machine-produced wealth is shared, or most people can end up miserably poor if the machine-owners successfully lobby against wealth redistribution.** So far, the trend seems to be toward the second option, with technology driving ever-increasing inequality.

Another Hawking quotation is also fitting. During the same interview,[6] to an AI professor's question about which moral ideas he should impart to his students, Hawking answered:

> … Please encourage your students to think not only about how to create AI, but also about how to ensure its beneficial use.

As a consequence we should question the reasonableness of AI applications such as the export of intelligent cruise missiles to "allied" Arab states, the deployment of humanoid combat robots, etc.

1.3.2 AI and Transportation

In the past 130 years, automotive industry engineers have made great strides. In Germany, one out of every two people owns their own car. These cars are highly reliable. This makes us very mobile and we use this very convenient mobility in work, everyday life and leisure. Moreover, we are dependent on it. Today, we can not get by without a motor vehicle, especially in rural areas with weak public transportation infrastructure, as for instance in Upper Swabia, where the author and his students live.

The next stage of increased convenience in road transportation is now imminent. In a few years, we will be able to buy electric self-driving cars, i.e. robotic cars, which will autonomously bring us to almost any destination. All passengers in the robotic car would be able to read, work or sleep during the trip. This is possible on public transit already, but passengers in a robotic car would be able to do this at any time and on any route.

Autonomous vehicles that can operate independently could also travel without passengers. This will lead to yet another increase in convenience: robotic taxis. Via a smartphone app, we will be able to order the optimal taxi, in terms of size and equipment, for any conceivable transportation purpose. We will be able to choose whether we want to travel alone in the taxi or whether we are willing to share a ride with

[6]https://www.reddit.com/user/Prof-Stephen-Hawking.

other passengers. We will not need our own car anymore. All associated responsibilities and expenses, such as refueling, technical service, cleaning, searching for parking, buying and selling, garage rent, etc. are void, which saves us money and effort.

Besides the immediate gains in comfort and convenience, robotic cars will offer other significant advantages. For example, according to a McKinsey study [GHZ14], we will need far fewer cars and, above all, far fewer parking places in the era of self-driving cars, which will lead to an immense reduction in resource consumption. According to a Lawrence Berkeley National Laboratory study [GS15], electric self-driving cars will cause a 90% reduction in green house emissions per passenger mile due to the vehicles' energy efficiency and the optimized fit between the vehicle and its purpose. Due to their optimal resource utilization, robotic taxis will be much more environmentally friendly than, for example, heavy buses, which often run at low capacity, especially in rural areas. Overall, robot taxis will contribute dramatically to energy savings and thus, among other things, to a significant improvement in CO_2 and climate problems.

Passenger safety will be much higher than it is today. Experts currently estimate future accident rates between zero and ten percent compared to today. Emotional driving ("road rage"), distracted driving and driving under the influence of drugs and alcohol will no longer exist.

Taxi drivers losing their jobs is often cited as a disadvantage of robotic cars. It is almost certain that there will no longer be taxi drivers from about 2030 onwards, but that is not necessarily a problem. As explained in the previous section, our society just needs to deal with the newly gained productivity properly.

In addition to the many advantages mentioned above, robotic cars have two critical problems. Firstly, the so-called rebound effect will nullify at least some of the gains in resource, energy and time savings. Shorter driving times as well as more comfortable and cheaper driving will tempt us to drive more. We can only deal with this problem by rethinking our attitude towards consumption and quality of life. Do we have to use the entire time saved for more activities? Here we are all invited to critical reflection.

Another problem we should take seriously is that the robotic cars will need to be networked. In principle, this gives hackers and terrorists the ability to access and manipulate the vehicles' controls through security holes in their network protocols. If a hacker manages to do this once, he could repeat the attack on a grand scale, potentially bringing entire vehicle fleets to a halt, causing accidents, spying on vehicle occupants, or initiating other criminal actions. Here, as in other areas such as home automation and the Internet of Things, IT security experts will be needed to ensure the highest possible security guarantees using tools of the trade such as cryptographic methods. By the way, improved machine learning algorithms will be useful in detecting hacking attacks.

1.3.3 Service Robotics

In a few years, shortly after self-driving cars, the next bit of consumption bait on the shelves of electronics stores will be service robots. Recently the Google subsidiary

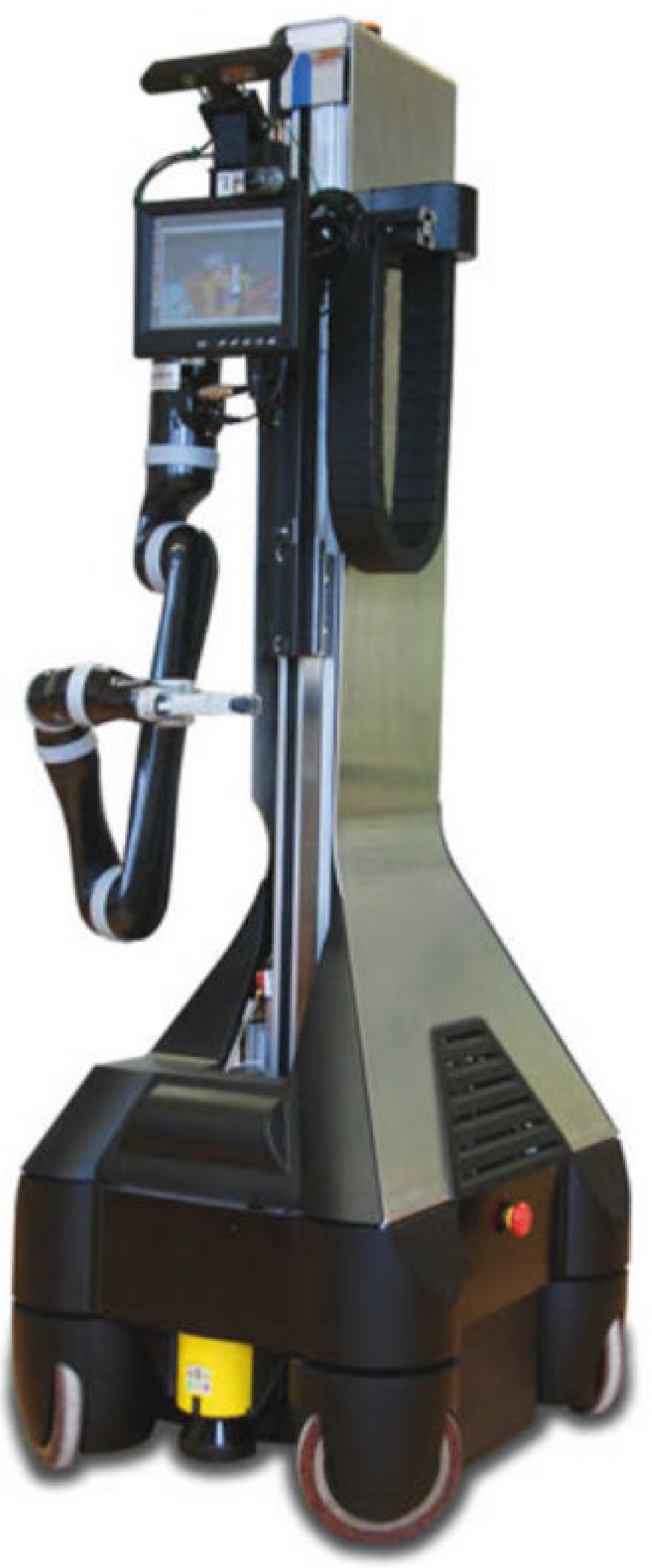
Fig. 1.4 The assistance robot Marvin, deployed in the AsRoBe research project

Boston Dynamics provided an impressive example in its humanoid robot Atlas.[7] Like the new cars, service robots offer a large gain in comfort and convenience which we would probably like to enjoy. One need only imagine such a robot dutifully cleaning and scrubbing after a party from night until morning without a grumble. Or think of the help that an assistance robot like Marvin, shown in Fig. 1.4, could provide to the elderly[8] or to people with disabilities [SPR+16].

In contrast to the robotic cars, however, these benefits come with costlier trade-offs. Completely new markets would be created, more natural resources and more energy would be consumed, and it is not even certain that people's lives would be simplified by the use of service robots in all areas. One of the first applications for robots like Atlas, developed by Boston Dynamics in contract with Google, will probably be military combat.

It is therefore all the more important that, before these robots come to market, we engage in social discourse on this topic. Science fiction films, such as "Ex Machina" (2015) with its female androids, the chilling "I, Robot" (2004) or the humorous "Robot and Frank" (2012), which depicts the pleasant side of a service robot as an old man's helper, can also contribute to such a discussion.

[7] https://youtu.be/rVlhMGQgDkY.

[8] In the coming demographic shift, assistance robots could become important for the elderly and thus for our whole society.

1.4 Agents

Although the term *intelligent* agents is not new to AI, only in recent years has it gained prominence through [RN10], among others. *Agent* denotes rather generally a system that processes information and produces an output from an input. These agents may be classified in many different ways.

In classical computer science, *software agents* are primarily employed (Fig. 1.5). In this case the agent consists of a program that calculates a result from user input.

In robotics, on the other hand, *hardware agents* (also called autonomous robots) are employed, which additionally have sensors and actuators at their disposal (Fig. 1.6). The agent can perceive its environment with the sensors. With the actuators it carries out actions and changes its environment.

With respect to the intelligence of the agent, there is a distinction between *reflex agents*, which only react to input, and *agents with memory*, which can also include the past in their decisions. For example, a driving robot that through its sensors knows its exact position (and the time) has no way, as a reflex agent, of determining its velocity. If, however, it saves the position, at short, discrete time steps, it can thus easily calculate its average velocity in the previous time interval.

If a reflex agent is controlled by a deterministic program, it represents a function of the set of all inputs to the set of all outputs. An agent with memory, on the other hand, is in general not a function. Why? (See Exercise 1.5 on page 21.) Reflex agents are sufficient in cases where the problem to be solved involves a Markov decision process. This is a process in which only the current state is needed to determine the optimal next action (see Chap. 10).

A mobile robot which should move from room 112 to room 179 in a building takes actions different from those of a robot that should move to room 105. In other words, the actions depend on the goal. Such agents are called *goal-based*.

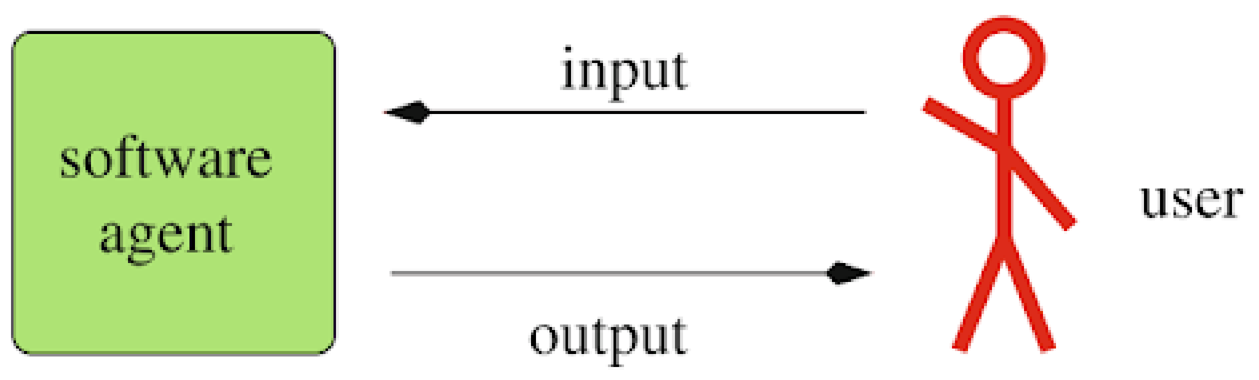

Fig. 1.5 A software agent with user interaction

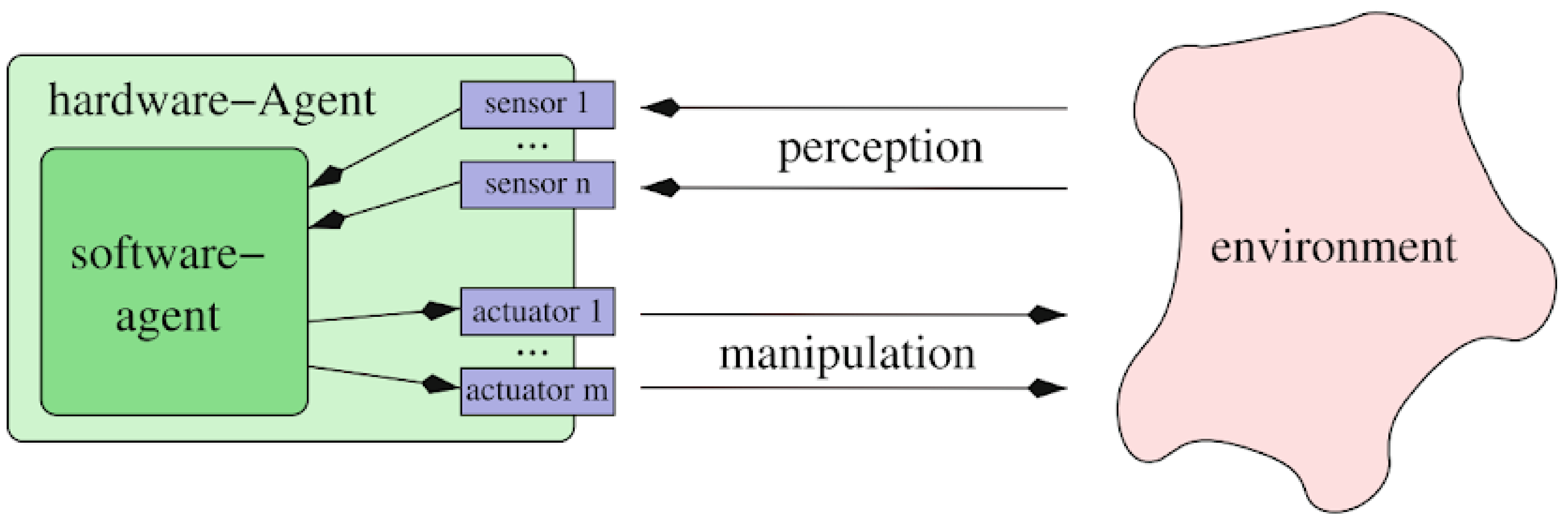

Fig. 1.6 A hardware agent

Example 1.1 A spam filter is an agent that puts incoming emails into wanted or unwanted (spam) categories, and deletes any unwanted emails. Its goal as a goal-based agent is to put all emails in the right category. In the course of this not-so-simple task, the agent can occasionally make mistakes. Because its goal is to classify all emails correctly, it will attempt to make as few errors as possible. However, that is not always what the user has in mind. Let us compare the following two agents. Out of 1,000 emails, Agent 1 makes only 12 errors. Agent 2 on the other hand makes 38 errors with the same 1,000 emails. Is it therefore worse than Agent 1? The errors of both agents are shown in more detail in the following table, the so-called "confusion matrix":

Agent 1:

		correct class	
		wanted	spam
spam filter decides	wanted	189	1
	spam	11	799

Agent 2:

		correct class	
		wanted	spam
spam filter decides	wanted	200	38
	spam	0	762

Agent 1 in fact makes fewer errors than Agent 2, but those few errors are severe because the user loses 11 potentially important emails. Because there are in this case two types of errors of differing severity, each error should be weighted with the appropriate cost factor (see Sect. 7.3.5 and Exercise 1.7 on page 21).

The sum of all weighted errors gives the total cost caused by erroneous decisions.The goal of a *cost-based agent* is to minimize the cost of erroneous decisions in the long term, that is, on average. In Sect. 7.3 we will become familiar with the medical diagnosis system LEXMED as an example of a cost-based agent.

Analogously, the goal of a utility-based agent is to maximize the utility derived from correct decisions in the long term, that is, on average. The sum of all decisions weighted by their respective utility factors gives the total utility.

Of particular interest in AI are *Learning agents*, which are capable of changing themselves given training examples or through positive or negative feedback, such that the average utility of their actions grows over time (see Chap. 8).

As mentioned in Sect. 1.2.5, *distributed agents* are increasingly coming into use, whose intelligence are not localized in one agent, but rather can only be seen through cooperation of many agents.

The design of an agent is oriented, along with its objective, strongly toward its *environment*, or alternately its picture of the environment, which strongly depends on it sensors. The environment is *observable* if the agent always knows the complete state of the world. Otherwise the environment is only *partially observable*. If an action always leads to the same result, then the environment is *deterministic*. Otherwise it is *nondeterministic*. In a *discrete environment* only finitely many states and actions occur, whereas a *continuous environment* boasts infinitely many states or actions.

1.5 Knowledge-Based Systems

An agent is a program that implements a mapping from perceptions to actions. For simple agents this way of looking at the problem is sufficient. For complex applications in which the agent must be able to rely on a large amount of information and is meant to do a difficult task, programming the agent can be very costly and unclear how to proceed. Here AI provides a clear path to follow that will greatly simplify the work.

First we separate *knowledge* from the system or program, which uses the knowledge to, for example, reach conclusions, answer queries, or come up with a plan. This system is called the *inference mechanism*. The knowledge is stored in a *knowledge base* (*KB*). Acquisition of knowledge in the knowledge base is denoted *Knowledge Engineering* and is based on various knowledge sources such as human experts, the knowledge engineer, and databases. Active learning systems can also acquire knowledge through active exploration of the world (see Chap. 10). In Fig. 1.7 the general architecture of knowledge-based systems is presented.

Moving toward a separation of knowledge and inference has several crucial advantages. The separation of knowledge and inference can allow inference systems to be implemented in a largely application-independent way. For example, application of a medical expert system to other diseases is much easier by replacing the knowledge base rather than by programming a whole new system.

Through the decoupling of the knowledge base from inference, knowledge can be stored declaratively. In the knowledge base there is only a description of the knowledge, which is independent from the inference system in use. Without this clear separation, knowledge and processing of inference steps would be interwoven, and any changes to the knowledge would be very costly.

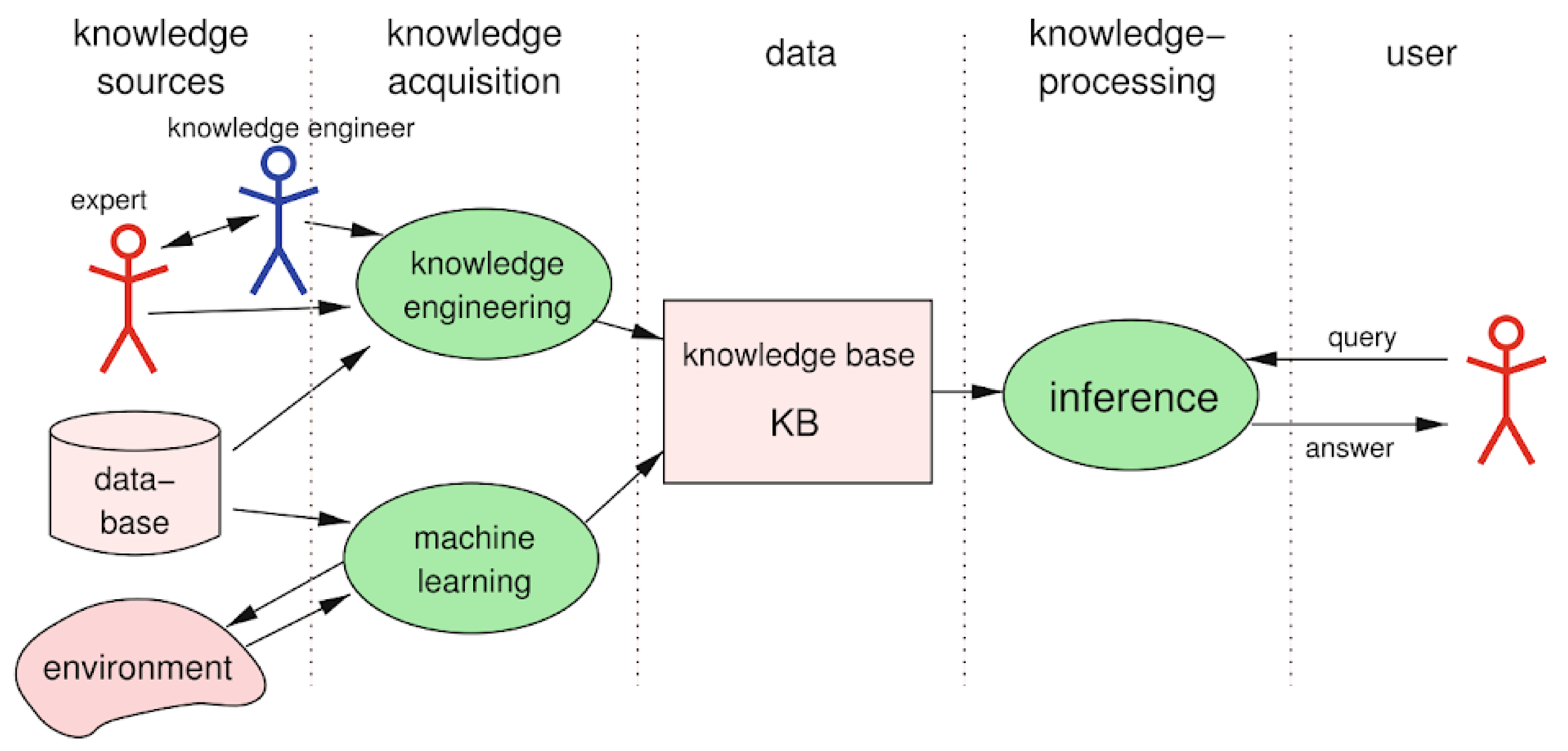

Fig. 1.7 Structure of a classic knowledge-processing system

Formal language as a convenient interface between man and machine lends itself to the representation of knowledge in the knowledge base. In the following chapters we will get to know a whole series of such languages. First, in Chaps. 2 and 3 there are *propositional calculus* and *first-order predicate logic* (PL1). But other formalisms such as probabilistic logic and decision trees are also presented. We start with propositional calculus and the related inference systems. Building on that, we will present predicate logic, a powerful language that is accessible by machines and very important in AI.

As an example for a large scale knowledge based system we want to refer to the software agent "Watson". Developed at IBM together with a number of universities, Watson is a question answering program, that can be fed with clues given in natural language. It works on a knowledge base comprising four terabytes of hard disk storage, including the full text of Wikipedia [FNA+09]. Watson was developed within IBM's DeepQA project which is characterized in [Dee11] as follows:

> The DeepQA project at IBM shapes a grand challenge in Computer Science that aims to illustrate how the wide and growing accessibility of natural language content and the integration and advancement of Natural Language Processing, Information Retrieval, Machine Learning, Knowledge Representation and Reasoning, and massively parallel computation can drive open-domain automatic Question Answering technology to a point where it clearly and consistently rivals the best human performance.

In the U.S. television quiz show "Jeopardy!", in February 2011, Watson defeated the two human champions Brad Rutter and Ken Jennings in a two-game, combined-point match and won the one million dollar price. One of Watson's particular strengths was its very fast reaction to the questions with the result that Watson often hit the buzzer (using a solenoid) faster than its human competitors and then was able to give the first answer to the question.

The high performance and short reaction times of Watson were due to an implementation on 90 IBM Power 750 servers, each of which contains 32 processors, resulting in 2880 parallel processors.

1.6 Exercises

Exercise 1.1 Test some of the chatterbots available on the internet. Start for example with www.hs-weingarten.de/~ertel/aibook in the collection of links under Turingtest/Chatterbots, or at www.simonlaven.com or www.alicebot.org. Write down a starting question and measure the time it takes, for each of the various programs, until you know for certain that it is not a human.

❄ ❄ **Exercise 1.2** At www.pandorabots.com you will find a server on which you can build a chatterbot with the markup language AIML quite easily. Depending on your interest level, develop a simple or complex chatterbot, or change an existing one.

Exercise 1.3 Give reasons for the unsuitability of the Turing test as a definition of "*artificial intelligence*" in practical AI.

➳ **Exercise 1.4** Many well-known inference processes, learning processes, etc. are NP-complete or even undecidable. What does this mean for AI?

Exercise 1.5

(a) Why is a deterministic agent with memory not a function from the set of all inputs to the set of all outputs, in the mathematical sense?

(b) How can one change the agent with memory, or model it, such that it becomes equivalent to a function but does not lose its memory?

Exercise 1.6 Let there be an agent with memory that can move within a plane. From its sensors, it receives at clock ticks of a regular interval Δt its exact position (x, y) in Cartesian coordinates.

(a) Give a formula with which the agent can calculate its velocity from the current time t and the previous measurement of $t - \Delta t$.

(b) How must the agent be changed so that it can also calculate its acceleration? Provide a formula here as well.

❄ **Exercise 1.7**

(a) Determine for both agents in Example 1.1 on page 18 the costs created by the errors and compare the results. Assume here that having to manually delete a spam email costs one cent and retrieving a deleted email, or the loss of an email, costs one dollar.

(b) Determine for both agents the profit created by correct classifications and compare the results. Assume that for every desired email recognized, a profit of one dollar accrues and for every correctly deleted spam email, a profit of one cent.

Propositional Logic 2

In propositional logic, as the name suggests, propositions are connected by logical operators. The statement "*the street is wet*" is a proposition, as is "*it is raining*". These two propositions can be connected to form the new proposition

if it is raining the street is wet.

Written more formally

it is raining $\Rightarrow$ *the street is wet.*

This notation has the advantage that the elemental propositions appear again in unaltered form. So that we can work with propositional logic precisely, we will begin with a definition of the set of all propositional logic formulas.

2.1 Syntax

Definition 2.1 Let $Op = \{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, (,)\}$ be the set of logical operators and Σ a set of symbols. The sets Op, Σ and $\{t, f\}$ are pairwise disjoint. Σ is called the *signature* and its elements are the *proposition variables*. The set of propositional logic formulas is now recursively defined:

- t and f are (atomic) formulas.
- All proposition variables, that is all elements from Σ, are (atomic) formulas.
- If A and B are formulas, then $\neg A$, (A), $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ are also formulas.

W. Ertel, *Introduction to Artificial Intelligence*, Undergraduate Topics in Computer Science, DOI 10.1007/978-3-319-58487-4_2

This elegant recursive definition of the set of all formulas allows us to generate infinitely many formulas. For example, given $\Sigma = \{A, B, C\}$,

$$A \wedge B, \quad A \wedge B \wedge C, \quad A \wedge A \wedge A, \quad C \wedge B \vee A, \quad (\neg A \wedge B) \Rightarrow (\neg C \vee A)$$

are formulas. $(((A)) \vee B)$ is also a syntactically correct formula.

Definition 2.2 We read the symbols and operators in the following way:

t:	"true"	
f:	"false"	
$\neg A$:	"not A"	(negation)
$A \wedge B$:	"A and B"	(conjunction)
$A \vee B$:	"A or B"	(disjunction)
$A \Rightarrow B$:	"if A then B"	(implication (also called *material implication*))
$A \Leftrightarrow B$:	"A if and only if B"	(equivalence)

The formulas defined in this way are so far purely syntactic constructions without meaning. We are still missing the semantics.

2.2 Semantics

In propositional logic there are two truth values: t for "true" and f for "false". We begin with an example and ask ourselves whether the formula $A \wedge B$ is true. The answer is: it depends on whether the variables A and B are true. For example, if A stands for "*It is raining today*" and B for "*It is cold today*" and these are both true, then $A \wedge B$ is true. If, however, B represents "*It is hot today*" (and this is false), then $A \wedge B$ is false.

We must obviously assign truth values that reflect the state of the world to proposition variables. Therefore we define

Definition 2.3 A mapping $I : \Sigma \rightarrow \{t, f\}$, which assigns a truth value to every proposition variable, is called an *interpretation*.

Because every proposition variable can take on two truth values, every propositional logic formula with n different variables has 2^n different interpretations. We define the truth values for the basic operations by showing all possible interpretations in a *truth table* (see Table 2.1 on page 25).

Table 2.1 Definition of the logical operators by truth table

A	B	(A)	$\neg A$	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
t	t	t	f	t	t	t	t
t	f	t	f	f	t	f	f
f	t	f	t	f	t	t	f
f	f	f	t	f	f	t	t

The empty formula is true for all interpretations. In order to determine the truth value for complex formulas, we must also define the order of operations for logical operators. If expressions are parenthesized, the term in the parentheses is evaluated first. For unparenthesized formulas, the priorities are ordered as follows, beginning with the strongest binding: $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$.

To clearly differentiate between the equivalence of formulas and syntactic equivalence, we define

Definition 2.4 Two formulas F and G are called semantically equivalent if they take on the same truth value for all interpretations. We write $F \equiv G$.

Semantic equivalence serves above all to be able to use the meta-language, that is, natural language, to talk about the object language, namely logic. The statement "$A \equiv B$" conveys that the two formulas A and B are semantically equivalent. The statement "$A \Leftrightarrow B$" on the other hand is a syntactic object of the formal language of propositional logic.

According to the number of interpretations in which a formula is true, we can divide formulas into the following classes:

Definition 2.5 A formula is called

- *Satisfiable* if it is true for at least one interpretation.
- *Logically valid* or simply *valid* if it is true for all interpretations. True formulas are also called *tautologies*.
- *Unsatisfiable* if it is not true for any interpretation.

Every interpretation that satisfies a formula is called a *model* of the formula.

Clearly the negation of every generally valid formula is unsatisfiable. The negation of a satisfiable, but not generally valid formula F is satisfiable.

We are now able to create truth tables for complex formulas to ascertain their truth values. We put this into action immediately using equivalences of formulas which are important in practice.

with n proposition variables, for all 2^n interpretations of the variables the formula $KB \Rightarrow Q$ must be evaluated. The computation time grows therefore exponentially with the number of variables. Therefore this process is unusable for large variable counts, at least in the worst case.

If a formula KB entails a formula Q, then by the deduction theorem $KB \Rightarrow Q$ is a tautology. Therefore the negation $\neg(KB \Rightarrow Q)$ is unsatisfiable. We have

$$\neg(KB \Rightarrow Q) \equiv \neg(\neg KB \vee Q) \equiv KB \wedge \neg Q.$$

Therefore, $KB \wedge \neg Q$ is also unsatisfiable. We formulate this simple, but important consequence of the deduction theorem as a theorem.

Theorem 2.3 (Proof by contradiction) *$KB \models Q$ if and only if $KB \wedge \neg Q$ is unsatisfiable.*

To show that the query Q follows from the knowledge base KB, we can also add the negated query $\neg Q$ to the knowledge base and derive a contradiction. Because of the equivalence $A \wedge \neg A \Leftrightarrow f$ from Theorem 2.1 on page 26 we know that a contradiction is unsatisfiable. Therefore, Q has been proved. This procedure, which is frequently used in mathematics, is also used in various automatic proof calculi such as the resolution calculus and in the processing of PROLOG programs.

One way of avoiding having to test all interpretations with the truth table method is the syntactic manipulation of the formulas KB and Q by application of inference rules with the goal of greatly simplifying them, such that in the end we can instantly see that $KB \models Q$. We call this syntactic process *derivation* and write $KB \vdash Q$. Such syntactic proof systems are called *calculi*. To ensure that a calculus does not generate errors, we define two fundamental properties of calculi.

Definition 2.7 A calculus is called *sound* if every derived proposition follows semantically. That is, if it holds for formulas KB and Q that

$$\text{if} \quad KB \vdash Q \quad \text{then} \quad KB \models Q.$$

A calculus is called *complete* if all semantic consequences can be derived. That is, for formulas KB and Q the following holds:

$$\text{if} \quad KB \models Q \quad \text{then} \quad KB \vdash Q.$$

The soundness of a calculus ensures that all derived formulas are in fact semantic consequences of the knowledge base. The calculus does not produce any “false consequences”. The completeness of a calculus, on the other hand, ensures that the calculus does not overlook anything. A complete calculus always finds a proof if

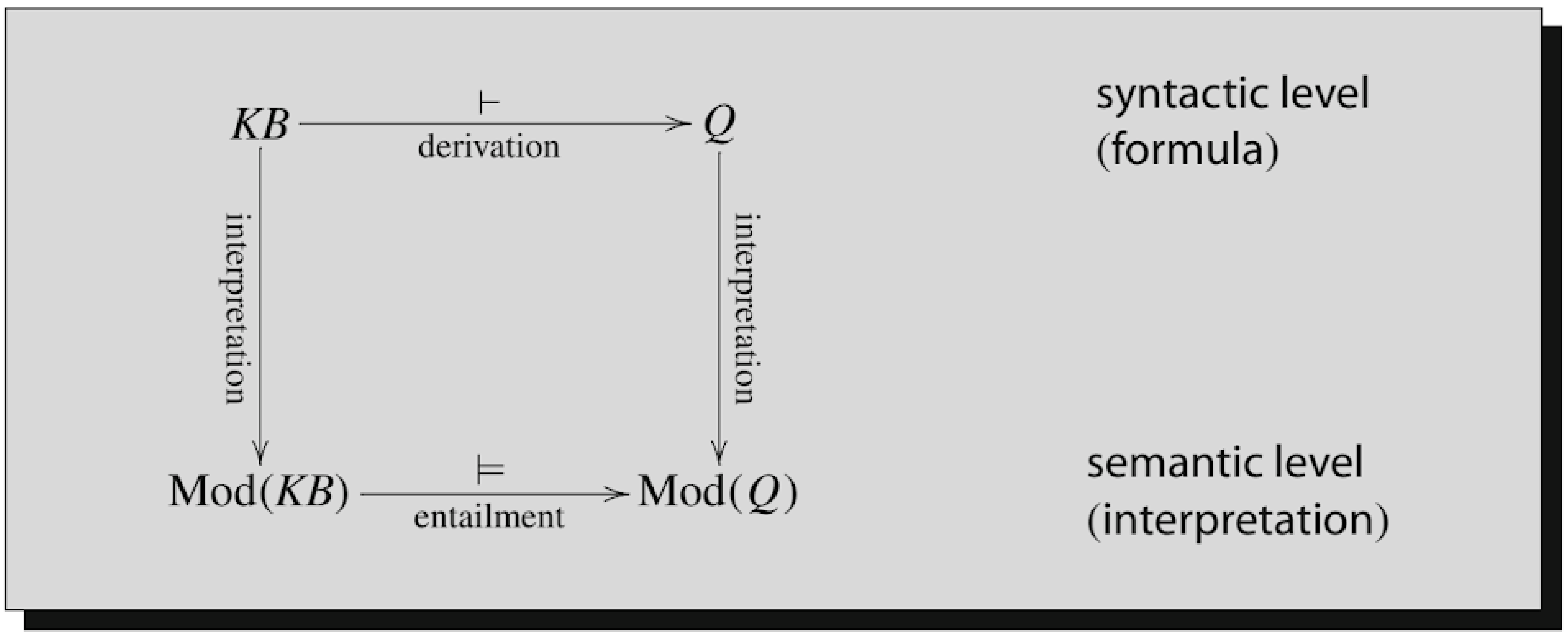
$$\begin{tikzcd}
KB \arrow[r, "\vdash", "\text{derivation}"'] \arrow[d, "\text{interpretation}"'] & Q \arrow[d, "\text{interpretation}"] \\
\mathrm{Mod}(KB) \arrow[r, "\models", "\text{entailment}"'] & \mathrm{Mod}(Q)
\end{tikzcd}$$

Fig. 2.1 Syntactic derivation and semantic entailment. Mod(X) represents the set of models of a formula X

the formula to be proved follows from the knowledge base. If a calculus is sound and complete, then syntactic derivation and semantic entailment are two equivalent relations (see Fig. 2.1).

To keep automatic proof systems as simple as possible, these are usually made to operate on formulas in conjunctive normal form.

Definition 2.8 A formula is in *conjunctive normal form* (*CNF*) if and only if it consists of a *conjunction*

$$K_1 \wedge K_2 \wedge \cdots \wedge K_m$$

of clauses. A clause K_i consists of a *disjunction*

$$(L_{i1} \vee L_{i2} \vee \cdots \vee L_{in_i})$$

of literals. Finally, a *literal* is a variable (positive literal) or a negated variable (negative literal).

The formula $(A \vee B \vee \neg C) \wedge (A \vee B) \wedge (\neg B \vee \neg C)$ is in conjunctive normal form. The conjunctive normal form does not place a restriction on the set of formulas because:

Theorem 2.4 *Every propositional logic formula can be transformed into an equivalent conjunctive normal form.*

Example 2.1 We put $A \vee B \Rightarrow C \wedge D$ into conjunctive normal form by using the equivalences from Theorem 2.1 on page 26:

$$\begin{aligned}
&A \vee B \Rightarrow C \wedge D \\
&\quad \equiv \neg(A \vee B) \vee (C \wedge D) && \text{(implication)} \\
&\quad \equiv (\neg A \wedge \neg B) \vee (C \wedge D) && \text{(de Morgan)} \\
&\quad \equiv (\neg A \vee (C \wedge D)) \wedge (\neg B \vee (C \wedge D)) && \text{(distributive law)} \\
&\quad \equiv ((\neg A \vee C) \wedge (\neg A \vee D)) \wedge ((\neg B \vee C) \wedge (\neg B \vee D)) && \text{(distributive law)} \\
&\quad \equiv (\neg A \vee C) \wedge (\neg A \vee D) \wedge (\neg B \vee C) \wedge (\neg B \vee D) && \text{(associative law)}
\end{aligned}$$

We are now only missing a calculus for syntactic proof of propositional logic formulas. We start with the *modus ponens*, a simple, intuitive rule of inference, which, from the validity of A and $A \Rightarrow B$, allows the derivation of B. We write this formally as

$$\frac{A, \quad A \Rightarrow B}{B}.$$

This notation means that we can derive the formula(s) below the line from the comma-separated formulas above the line. Modus ponens as a rule by itself, while sound, is not complete. If we add additional rules we can create a complete calculus, which, however, we do not wish to consider here. Instead we will investigate the *resolution rule*

$$\frac{A \vee B, \quad \neg B \vee C}{A \vee C} \tag{2.1}$$

as an alternative. The derived clause is called *resolvent*. Through a simple transformation we obtain the equivalent form

$$\frac{A \vee B, \quad B \Rightarrow C}{A \vee C}.$$

If we set A to f, we see that the resolution rule is a generalization of the modus ponens. The resolution rule is equally usable if C is missing or if A and C are missing. In the latter case the empty clause can be derived from the contradiction $B \wedge \neg B$ (Exercise 2.7 on page 38).

2.4 Resolution

We now generalize the resolution rule again by allowing clauses with an arbitrary number of literals. With the literals $A_1, \ldots, A_m, B, C_1, \ldots, C_n$ the *general resolution rule* reads

$$\frac{(A_1 \vee \cdots \vee A_m \vee B), \quad (\neg B \vee C_1 \vee \cdots \vee C_n)}{(A_1 \vee \cdots \vee A_m \vee C_1 \vee \cdots \vee C_n)}. \tag{2.2}$$

We call the literals B and $\neg B$ complementary. The resolution rule deletes a pair of complementary literals from the two clauses and combines the rest of the literals into a new clause.

To prove that from a knowledge base KB, a query Q follows, we carry out a proof by contradiction. Following Theorem 2.3 on page 28 we must show that a contradiction can be derived from $KB \wedge \neg Q$. In formulas in conjunctive normal form, a contradiction appears in the form of two clauses (A) and $(\neg A)$, which lead to the empty clause as their resolvent. The following theorem ensures us that this process really works as desired.

For the calculus to be complete, we need a small addition, as shown by the following example. Let the formula $(A \vee A)$ be given as our knowledge base. To show by the resolution rule that from there we can derive $(A \wedge A)$, we must show that the empty clause can be derived from $(A \vee A) \wedge (\neg A \vee \neg A)$. With the resolution rule alone, this is impossible. With *factorization*, which allows deletion of copies of literals from clauses, this problem is eliminated. In the example, a double application of factorization leads to $(A) \wedge (\neg A)$, and a resolution step to the empty clause.

Theorem 2.5 *The resolution calculus for the proof of unsatisfiability of formulas in conjunctive normal form is sound and complete.*

Because it is the job of the resolution calculus to derive a contradiction from $KB \wedge \neg Q$, it is very important that the knowledge base KB is *consistent*:

Definition 2.9 A formula KB is called *consistent* if it is impossible to derive from it a contradiction, that is, a formula of the form $\phi \wedge \neg\phi$.

Otherwise anything can be derived from KB (see Exercise 2.8 on page 38). This is true not only of resolution, but also for many other calculi.

Of the calculi for automated deduction, resolution plays an exceptional role. Thus we wish to work a bit more closely with it. In contrast to other calculi, resolution has only two inference rules, and it works with formulas in conjunctive normal form. This makes its implementation simpler. A further advantage compared to many calculi lies in its reduction in the number of possibilities for the application of inference rules in every step of the proof, whereby the search space is reduced and computation time decreased.

As an example, we start with a simple logic puzzle that allows the important steps of a resolution proof to be shown.

Example 2.2 Logic puzzle number 7, entitled *A charming English family*, from the German book [Ber89] reads (translated to English):

Despite studying English for seven long years with brilliant success, I must admit that when I hear English people speaking English I'm totally perplexed. Recently, moved by noble feelings, I picked up three hitchhikers, a father, mother, and daughter, who I quickly realized were English and only spoke English. At each of the sentences that follow I wavered between two possible interpretations. They told me the following (the second possible meaning is in parentheses): The father: "We are going to Spain (we are from Newcastle)." The mother: "We are not going to Spain and are from Newcastle (we stopped in Paris and are not going to Spain)." The daughter: "We are not from Newcastle (we stopped in Paris)." What about this charming English family?

To solve this kind of problem we proceed in three steps: formalization, transformation into normal form, and proof. In many cases formalization is by far the most difficult step because it is easy to make mistakes or forget small details. (Thus practical exercise is very important. See Exercises 2.9–2.11 on page 38.)

Here we use the variables S for "*We are going to Spain*", N for "*We are from Newcastle*", and P for "*We stopped in Paris*" and obtain as a formalization of the three propositions of father, mother, and daughter

$$(S \vee N) \wedge [(\neg S \wedge N) \vee (P \wedge \neg S)] \wedge (\neg N \vee P).$$

Factoring out $\neg S$ in the middle sub-formula brings the formula into CNF in one step. Numbering the clauses with subscripted indices yields

$$KB \equiv (S \vee N)_1 \wedge (\neg S)_2 \wedge (P \vee N)_3 \wedge (\neg N \vee P)_4.$$

Now we begin the resolution proof, at first still without a query Q. An expression of the form "Res(m, n): $\langle clause \rangle_k$" means that $\langle$clause$\rangle$ is obtained by resolution of clause m with clause n and is numbered k.

$$\begin{aligned} \text{Res}(1,2): &\quad (N)_5 \\ \text{Res}(3,4): &\quad (P)_6 \\ \text{Res}(1,4): &\quad (S \vee P)_7 \end{aligned}$$

We could have derived clause P also from Res(4, 5) or Res(2, 7). Every further resolution step would lead to the derivation of clauses that are already available. Because it does not allow the derivation of the empty clause, it has therefore been shown that the knowledge base is non-contradictory. So far we have derived N and P. To show that $\neg S$ holds, we add the clause $(S)_8$ to the set of clauses as a negated query. With the resolution step

$$\text{Res}(2,8): \quad ()_9$$

the proof is complete. Thus $\neg S \wedge N \wedge P$ holds. The "charming English family" evidently comes from Newcastle, stopped in Paris, but is not going to Spain.

If we now want to know whether *skiing* holds, this can easily be derived. A slightly generalized modus ponens suffices here as an inference rule:

$$\frac{A_1 \wedge \cdots \wedge A_m, \quad A_1 \wedge \cdots \wedge A_m \Rightarrow B}{B}.$$

The proof of "*skiing*" has the following form (MP(i_1, …, i_k) represents application of the modus ponens on clauses i_1 to i_k:

$$\begin{aligned} \mathrm{MP}(2,3): &\quad (snow)_5 \\ \mathrm{MP}(1,5,4): &\quad (skiing)_6. \end{aligned}$$

With modus ponens we obtain a complete calculus for formulas that consist of propositional logic Horn clauses. In the case of large knowledge bases, however, modus ponens can derive many unnecessary formulas if one begins with the wrong clauses. Therefore, in many cases it is better to use a calculus that starts with the query and works backward until the facts are reached. Such systems are designated *backward chaining*, in contrast to *forward chaining* systems, which start with facts and finally derive the query, as in the above example with the modus ponens.

For backward chaining of Horn clauses, *SLD resolution* is used. SLD stands for "*Selection rule driven linear resolution for definite clauses*". In the above example, augmented by the negated query (*skiing* ⇒ *f*)

$$\begin{aligned} &(nice\_weather)_1 \\ &(snowfall)_2 \\ &(snowfall \Rightarrow snow)_3 \\ &(nice\_weather \wedge snow \Rightarrow skiing)_4 \\ &(skiing \Rightarrow f)_5 \end{aligned}$$

we carry out SLD resolution beginning with the resolution steps that follow from this clause

$$\begin{aligned} \mathrm{Res}(5,4): &\quad (nice\_weather \wedge snow \Rightarrow f)_6 \\ \mathrm{Res}(6,1): &\quad (snow \Rightarrow f)_7 \\ \mathrm{Res}(7,3): &\quad (snowfall \Rightarrow f)_8 \\ \mathrm{Res}(8,2): &\quad () \end{aligned}$$

and derive a contradiction with the empty clause. Here we can easily see "*linear resolution*", which means that further processing is always done on the currently derived clause. This leads to a great reduction of the search space. Furthermore, the literals of the current clause are always processed in a fixed order (for example, from right to left) ("*Selection rule driven*"). The literals of the current clause are called *subgoal*. The literals of the negated query are the *goals*. The inference rule for one step reads

$$\frac{A_1 \wedge \cdots \wedge A_m \Rightarrow B_1, \quad B_1 \wedge B_2 \wedge \cdots \wedge B_n \Rightarrow f}{A_1 \wedge \cdots \wedge A_m \wedge B_2 \wedge \cdots \wedge B_n \Rightarrow f}.$$

Before application of the inference rule, $B_1, B_2, \ldots, B_n$—the current subgoals—must be proved. After the application, B_1 is replaced by the new subgoal $A_1 \wedge \cdots \wedge A_m$. To show that B_1 is true, we must now show that $A_1 \wedge \cdots \wedge A_m$ are true. This process continues until the list of subgoals of the current clauses (the so-called *goal stack*) is empty. With that, a contradiction has been found. If, for a subgoal $\neg B_i$, there is no clause with the complementary literal B_i as its clause head, the proof terminates and no contradiction can be found. The query is thus unprovable.

SLD resolution plays an important role in practice because programs in the logic programming language PROLOG consist of predicate logic Horn clauses, and their processing is achieved by means of SLD resolution (see Exercise 2.13 on page 38, or Chap. 5).

2.6 Computability and Complexity

The truth table method, as the simplest semantic proof system for propositional logic, represents an algorithm that can determine every model of any formula in finite time. Thus the sets of unsatisfiable, satisfiable, and valid formulas are decidable. The computation time of the truth table method for satisfiability grows in the worst case exponentially with the number n of variables because the truth table has 2^n rows. An optimization, the method of *semantic trees*, avoids looking at variables that do not occur in clauses, and thus saves computation time in many cases, but in the worst case it is likewise exponential.

In resolution, in the worst case the number of derived clauses grows exponentially with the number of initial clauses. To decide between the two processes, we can therefore use the rule of thumb that in the case of many clauses with few variables, the truth table method is preferable, and in the case of few clauses with many variables, resolution will probably finish faster.

The question remains: can proof in propositional logic go faster? Are there better algorithms? The answer: probably not. After all, S. Cook, the founder of complexity theory, has shown that the 3-SAT problem is NP-complete. 3-SAT is the set of all CNF formulas whose clauses have exactly three literals. Thus it is clear that there is probably (modulo the P/NP problem) no polynomial algorithm for 3-SAT, and thus probably not a general one either. For Horn clauses, however, there is an algorithm in which the computation time for testing satisfiability grows only linearly as the number of literals in the formula increases.

2.7 Applications and Limitations

Theorem provers for propositional logic are part of the developer's everyday toolset in digital technology. For example, the verification of digital circuits and the generation of test patterns for testing of microprocessors in fabrication are some of these tasks. Special proof systems that work with binary decision diagrams (BDD) are also employed as a data structure for processing propositional logic formulas.

In AI, propositional logic is employed in simple applications. For example, simple expert systems can certainly work with propositional logic. However, the variables must all be discrete, with only a few values, and there may not be any cross-relations between variables. Complex logical connections can be expressed much more elegantly using predicate logic.

Probabilistic logic is a very interesting and current combination of propositional logic and probabilistic computation that allows modeling of uncertain knowledge. It is handled thoroughly in Chap. 7.

2.8 Exercises

➡ **Exercise 2.1** Give a Backus–Naur form grammar for the syntax of propositional logic.

Exercise 2.2 Show that the following formulas are tautologies:

(a) $\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$
(b) $A \Rightarrow B \Leftrightarrow \neg B \Rightarrow \neg A$
(c) $((A \Rightarrow B) \wedge (B \Rightarrow A)) \Leftrightarrow (A \Leftrightarrow B)$
(d) $(A \vee B) \wedge (\neg B \vee C) \Rightarrow (A \vee C)$

Exercise 2.3 Transform the following formulas into conjunctive normal form:

(a) $A \Leftrightarrow B$
(b) $A \wedge B \Leftrightarrow A \vee B$
(c) $A \wedge (A \Rightarrow B) \Rightarrow B$

Exercise 2.4 Check the following statements for satisfiability or validity.

(a) (play_lottery ∧ six_right) ⇒ winner
(b) (play_lottery ∧ six_right ∧ (six_right ⇒ win)) ⇒ win
(c) ¬(¬gas_in_tank ∧ (gas_in_tank ∨ ¬car_starts) ⇒ ¬car_starts)

❄❄ **Exercise 2.5** Using the programming language of your choice, program a theorem prover for propositional logic using the truth table method for formulas in conjunctive normal form. To avoid a costly syntax check of the formulas, you may represent clauses as lists or sets of literals, and the formulas as lists or sets of clauses. The program should indicate whether the formula is unsatisfiable, satisfiable, or true, and output the number of different interpretations and models.

Exercise 2.6
(a) Show that modus ponens is a valid inference rule by showing that $A \wedge (A \Rightarrow B) \models B$.
(b) Show that the resolution rule (2.1) is a valid inference rule.

❄ **Exercise 2.7** Show by application of the resolution rule that, in conjunctive normal form, the empty clause is equivalent to the false statement.

❄ **Exercise 2.8** Show that, with resolution, one can "derive" any arbitrary clause from a knowledge base that contains a contradiction.

Exercise 2.9 Formalize the following logical functions with the logical operators and show that your formula is valid. Present the result in CNF.
(a) The XOR operation (exclusive or) between two variables.
(b) The statement *at least two of the three variables* A, B, C *are true*.

❄ **Exercise 2.10** Solve the following case with the help of a resolution proof: "If the criminal had an accomplice, then he came in a car. The criminal had no accomplice and did not have the key, or he had the key and an accomplice. The criminal had the key. Did the criminal come in a car or not?"

Exercise 2.11 Show by resolution that the formula from
(a) Exercise 2.2(d) is a tautology.
(b) Exercise 2.4(c) is unsatisfiable.

Exercise 2.12 Prove the following equivalences, which are important for working with Horn clauses:
(a) $(\neg A_1 \vee \cdots \vee \neg A_m \vee B) \equiv A_1 \wedge \cdots \wedge A_m \Rightarrow B$
(b) $(\neg A_1 \vee \cdots \vee \neg A_m) \equiv A_1 \wedge \cdots \wedge A_m \Rightarrow f$
(c) $A \equiv w \Rightarrow A$

Exercise 2.13 Show by SLD resolution that the following Horn clause set is unsatisfiable.

$$\begin{array}{lll}
(A)_1 & (D)_4 & (A \wedge D \Rightarrow G)_7 \\
(B)_2 & (E)_5 & (C \wedge F \wedge E \Rightarrow H)_8 \\
(C)_3 & (A \wedge B \wedge C \Rightarrow F)_6 & (H \Rightarrow f)_9
\end{array}$$

➠ **Exercise 2.14** In Sect. 2.6 it says: "Thus it is clear that there is probably (modulo the P/NP problem) no polynomial algorithm for 3-SAT, and thus probably not a general one either." Justify the "probably" in this sentence.

First-order Predicate Logic 3

Many practical, relevant problems cannot be or can only very inconveniently be formulated in the language of propositional logic, as we can easily recognize in the following example. The statement

"*Robot 7 is situated at the xy position (35, 79)*"

can in fact be directly used as the propositional logic variable

"*Robot_7_is_situated_at_xy_position_(35, 79)*"

for reasoning with propositional logic, but reasoning with this kind of proposition is very inconvenient. Assume 100 of these robots can stop anywhere on a grid of 100×100 points. To describe every position of every robot, we would need $100 \cdot 100 \cdot 100 = 1\,000\,000 = 10^6$ different variables. The definition of relationships between objects (here robots) becomes truly difficult. The relation

"*Robot A is to the right of robot B.*"

is semantically nothing more than a set of pairs. Of the 10 000 possible pairs of x-coordinates there are $(99 \cdot 98)/2 = 4851$ ordered pairs. Together with all 10 000 combinations of possible y-values for both robots, there are $(100 \cdot 99) = 9900$ formulas of the type

$$\begin{aligned}&\mathit{Robot\_7\_is\_to\_the\_right\_of\_robot\_12} \Leftrightarrow\\ &\quad \mathit{Robot\_7\_is\_situated\_at\_xy\_position\_}(35, 79)\\ &\quad \wedge \mathit{Robot\_12\_is\_situated\_at\_xy\_position\_}(10, 93) \vee \ldots\end{aligned}$$

defining these relations, each of them with $(10^4)^2 \cdot 0.485 = 0.485 \cdot 10^8$ alternatives on the right side. In first-order predicate logic, we can define a predicate

W. Ertel, *Introduction to Artificial Intelligence*, Undergraduate Topics in Computer Science, DOI 10.1007/978-3-319-58487-4_3

Definition 3.3 An *interpretation* $\mathbb{I}$ is defined as

- A mapping from the set of constants and variables $K \cup V$ to a set W of names of objects in the world.
- A mapping from the set of function symbols to the set of functions in the world. Every n-place function symbol is assigned an n-place function.
- A mapping from the set of predicate symbols to the set of relations in the world. Every n-place predicate symbol is assigned an n-place relation.

Example 3.1 Let c_1, c_2, c_3 be constants, "*plus*" a two-place function symbol, and "*gr*" a two-place predicate symbol. The truth of the formula

$$F \equiv gr(plus(c_1, c_3), c_2)$$

depends on the interpretation $\mathbb{I}$. We first choose the following obvious interpretation of constants, the function, and of the predicates in the natural numbers:

$$\mathbb{I}_1\colon\ c_1 \mapsto 1,\ c_2 \mapsto 2,\ c_3 \mapsto 3, \quad plus \mapsto +, \quad gr \mapsto > .$$

Thus the formula is mapped to

$$1 + 3 > 2, \quad \text{or after evaluation} \quad 4 > 2.$$

The greater-than relation on the set $\{1, 2, 3, 4\}$ is the set of all pairs (x, y) of numbers with $x > y$, meaning the set $G = \{(4, 3), (4, 2), (4, 1), (3, 2), (3, 1), (2, 1)\}$. Because $(4, 2) \in G$, the formula F is true under the interpretation $\mathbb{I}_1$. However, if we choose the interpretation

$$\mathbb{I}_2\colon\ c_1 \mapsto 2,\ c_2 \mapsto 3,\ c_3 \mapsto 1, \quad plus \mapsto -, \quad gr \mapsto > ,$$

we obtain

$$2 - 1 > 3, \quad \text{or} \quad 1 > 3.$$

The pair $(1, 3)$ is not a member of G. The formula F is false under the interpretation $\mathbb{I}_2$. Obviously, the truth of a formula in PL1 depends on the interpretation. Now, after this preview, we define truth.

Definition 3.4

- An atomic formula $p(t_1, \dots, t_n)$ is *true* (or valid) under the interpretation $\mathbb{I}$ if, after interpretation and evaluation of all terms $t_1, \dots, t_n$ and interpretation of the predicate p through the n-place relation r, it holds that

$$(\mathbb{I}(t_1), \dots, \mathbb{I}(t_n)) \in r.$$

- The truth of quantifierless formulas follows from the truth of atomic formulas—as in propositional calculus—through the semantics of the logical operators defined in Table 2.1 on page 25.
- A formula $\forall x\ F$ is true under the interpretation $\mathbb{I}$ exactly when it is true given an arbitrary change of the interpretation for the variable x (and only for x)
- A formula $\exists x\ F$ is true under the interpretation $\mathbb{I}$ exactly when there is an interpretation for x which makes the formula true.

The definitions of semantic equivalence of formulas, for the concepts satisfiable, true, unsatisfiable, and model, along with semantic entailment (Definitions 2.4, 2.5, 2.6) carry over unchanged from propositional calculus to predicate logic.

Theorem 3.1 *Theorems 2.2 (deduction theorem) and 2.3 (proof by contradiction) hold analogously for PL1.*

Example 3.2 The family tree given in Fig. 3.1 graphically represents (in the semantic level) the relation

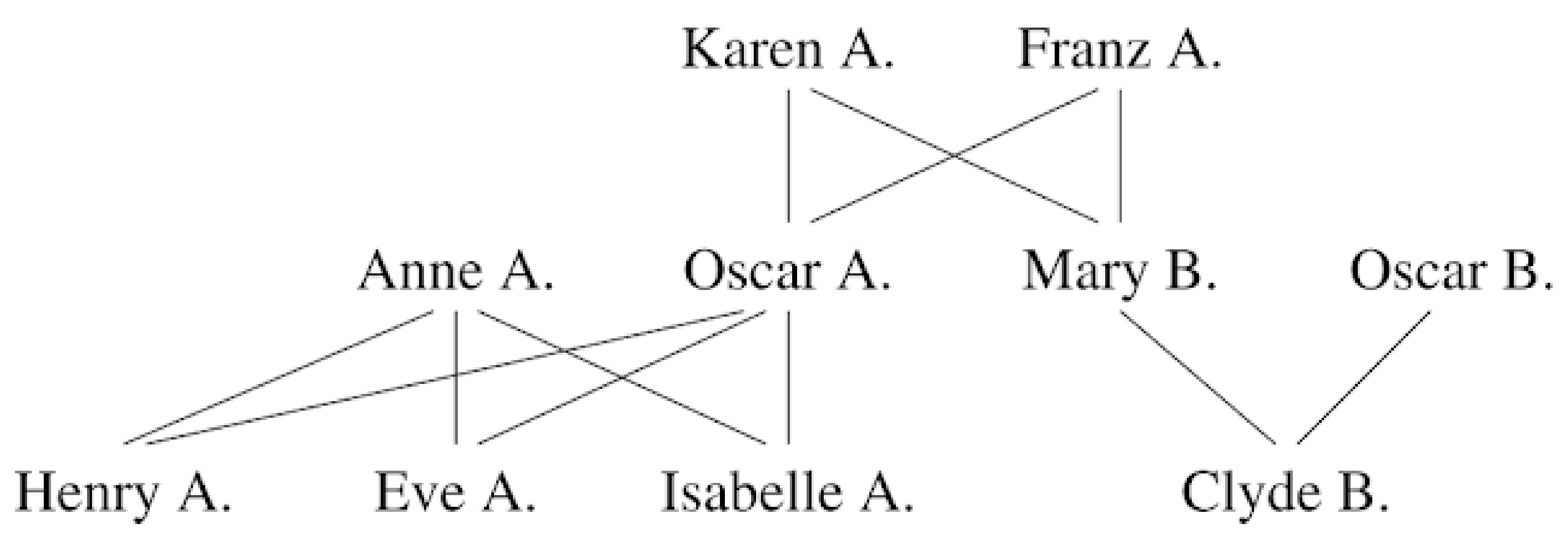

Fig. 3.1 A family tree. The edges going from Clyde B. upward to Mary B. and Oscar B. represent the element (Clyde B., Mary B., Oscar B.) as a child relationship

$$\begin{aligned}\text{Child} = \{&(\text{Oscar A., Karen A., Frank A.}), && (\text{Mary B., Karen A., Frank A.}),\\ &(\text{Henry A., Anne A., Oscar A.}), && (\text{Eve A., Anne A., Oscar A.}),\\ &(\text{Isabelle A., Anne A., Oscar A.}), && (\text{Clyde B., Mary B., Oscar B.})\}\end{aligned}$$

For example, the triple (Oscar A., Karen A., Frank A.) stands for the proposition "*Oscar A. is a child of Karen A. and Frank A.*". From the names we read off the one-place relation

$$\text{Female} = \{\text{Karen A., Anne A., Mary B., Eve A., Isabelle A.}\}$$

of the women. We now want to establish formulas for family relationships. First we define a three-place predicate *child*(x, y, z) with the semantic

$$\mathbb{I}(child(x,y,z)) = w \equiv (\mathbb{I}(x), \mathbb{I}(y), \mathbb{I}(z)) \in \text{Kind}.$$

Under the interpretation $\mathbb{I}(oscar) = \text{Oscar A.}$, $\mathbb{I}(eve) = \text{Eve A.}$, $\mathbb{I}(anne) = \text{Anne A.}$, it is also true that *child*(*eve*, *anne*, *oscar*). For *child*(*eve*, *oscar*, *anne*) to be true, we require, with

$$\forall x\ \forall y\ \forall z\ child(x,y,z)\ \Leftrightarrow\ child(x,z,y),$$

symmetry of the predicate *child* in the last two arguments. For further definitions we refer to Exercise 3.1 on page 63 and define the predicate *descendant* recursively as

$$\begin{aligned}\forall x\ \forall y\ descendant(x,\ y) \Leftrightarrow \exists z\ child(x,\ y,\ z) \vee\\ (\exists u\ \exists v\ child(x,\ u,\ v)\ \wedge\ descendant(u,\ y)).\end{aligned}$$

Now we build a small knowledge base with rules and facts. Let

$$\begin{aligned}KB \equiv\ & female(karen)\ \wedge\ female(anne)\ \wedge\ female(mary)\\ &\wedge\ female(eve)\ \wedge\ female(isabelle)\\ &\wedge\ child(oscar,\ karen,\ franz)\ \wedge\ child(mary,\ karen,\ franz)\\ &\wedge\ child(eve,\ anne,\ oscar)\ \wedge\ child(henry,\ anne,\ oscar)\\ &\wedge\ child(isabelle,\ anne,\ oscar)\ \wedge\ child(clyde,\ mary,\ oscarb)\\ &\wedge\ (\forall x\ \forall y\ \forall z\ child(x,\ y,\ z)\ \Rightarrow\ child(x,\ z,\ y))\\ &\wedge\ (\forall x\ \forall y\ descendant(x,\ y)\ \Leftrightarrow\ \exists z\ child(x,\ y,\ z)\\ &\vee\ (\exists u\ \exists v\ child(x,\ u,\ v)\ \wedge\ descendant(u,\ y))).\end{aligned}$$

We can now ask, for example, whether the propositions *child*(*eve*, *oscar*, *anne*) or *descendant*(*eve*, *franz*) are derivable. To that end we require a calculus.