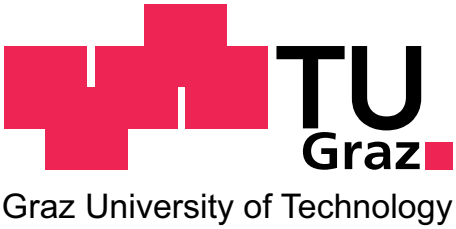


Graz University of Technology

DESIGN AND EVALUATION OF AN INTRODUCTORY ARTIFICIAL INTELLIGENCE CLASS IN HIGHSCHOOLS

A diploma thesis by

HARALD BURGSTEINER

supervised by

Ass.Prof. Dipl.-Ing. Dr.techn. Gerald Steinbauer

and submitted to the

Graz University of Technology

to achieve the university degree of

Magister rerum naturalium (Mag.rer.net. / MSc)

June 2016

To you, Anna and Noah, in the hope that you will one day learn about cool things like these in school.

Acknowledgements

First I'd like to thank the people at the RoboLab of the Institute for Software Technology for the possibility to work on an interesting topic and the offer to use their equipment during the making of this course, especially Gerald Steinbauer and Martin Kandlhofer, who gave me many valuable suggestions and support.

Furthermore I'd like to thank the colleagues from the BRG Kepler for their support during the course. And dear students, I also would like thank you for your participation, your collaboration and your feedback. We had some very valuable discussions for me during the course and I rarely see students with that much motivation and skill in schools!

Thank you too Gabi, for cheering me up (sometimes!) when I became nervous before exams and also of course for proof reading this thesis and discussing it with me. Very big hugs go to Noah and Anna who had to sacrifice some time with their dad because of his studies. I promise, I'll make it up to you again!

Abstract

Profound knowledge about at least the fundamentals of Artificial Intelligence (AI) will become increasingly important for careers in science and engineering. Therefore we present an innovative educational project teaching fundamental concepts of artificial intelligence at a high school level. Currently, computer science education in school does not include teaching these fundamental AI topics in an adequate manner. In order to overcome this shortcoming we developed a high school AI-course (called '*iRobot*') dealing with major topics of AI and computer science (automatons, agent systems, data structures, search algorithms, graphs, problem solving by searching, planning, machine learning) according to suggestions in the current literature. The main difficulties were selecting suitable topics, abstracting them for highschool students and offering social forms to explore the contents accordingly.

The course was divided into seven weekly teaching units of two hours each, comprising both theoretical and hands-on components. We conducted and empirically evaluated a pilot project in a representative Austrian high school which integrates robotics in its regular curriculum. The results of the evaluation show that the participating students have become familiar with the concepts included in the course and the various topics addressed. Results and lessons learned from this project form the basis for further projects in different schools which intend to integrate artificial intelligence in future secondary science education.

Zusammenfassung

Gute Kenntnisse von zumindest den Grundlagen der Künstlichen Intelligenz (KI) werden zunehmend wichtiger für Karrieren in der Wissenschaft und der Technik. Darum präsentieren wir ein innovatives Lernprojekt, das die fundamentalen Konzepte der künstlichen Intelligenz auf Gymnasialniveau lehrt. Momentan beinhaltet die Informatikausbildung in Schulen diese grundlegenden KI-Themen in einem nicht ausreichenden Maß. Um diesem Mangel entgegenzuwirken, entwickelten wir einen Kurs namens *iRobot* aus dem Bereich der KI, der wichtige Themen der KI und der Informatik, wie Automaten, Agenten, Datenstrukturen, Suchalgorithmen, Graphen, Problemlösen durch Suchen, Planen und maschinelles Lernen beinhaltet. Die Themen wurden anhand von Vorschlägen in der aktuellen Literatur ausgewählt. Die größten Herausforderungen waren, sinnvolle Themen auszuwählen, diese für die Schüler_innen zu abstrahieren und passende Sozialformen für das Erforschen der Inhalte anzubieten.

Der Kurs wurde aufgeteilt in sieben wöchentliche Unterrichtseinheiten zu je zwei Stunden. Die Einheiten enthielten sowohl theoretische als auch praktische Komponenten. Wir hielten den Kurs in einem repräsentativen, österreichischen Gymnasium ab, das Robotikunterricht im regulären Stundenplan beinhaltet. Der Kurs wurde danach empirisch evaluiert und ausgewertet. Die Ergebnisse der Evaluierung zeigen, dass die teilnehmenden Schülerinnen und Schüler mit den verschiedenen inkludierten Konzepten und behandelten Themen im Kurs vertraut wurden. Die Ergebnisse und Erkenntnisse aus diesem Projekt formen die Basis für weitere Projekte an anderen Gymnasien, die künstliche Intelligenz in ihr zukünftiges Curriculum im Bereich der Naturwissenschaften aufnehmen wollen.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Nowadays nearly every day an article is published in a newspaper in which a new gadget or a machine is being described. Many of those innovations are labelled with tags like "adaptive", "learning", "smart" or "intelligent". To some of us, stories like these are exciting and fascinating. Others are frightened of machines that might be able to learn, outsmart mankind one day and terminate biological life on earth (like in many science fiction movies).

Therefore, it will be important for our civilisation to be able to differentiate between just smart gadgets that contain e.g. an adaptable algorithm to better sort their shopping list and real artificial intelligence (AI) that indeed might eventually become smarter than humans like many renowned scientists like Steven Hawkins and others argue [Future of Life Institute, 2015].

Smart tools and machines are currently flooding the market. Examples start e.g. with a simple smart watch like in [Deutsch and Burgsteiner, 2016] that is able to adapt itself to the everyday life of an elderly person and alert relatives when unusual inactivity or a fall was detected. Other examples include intelligent household appliances, smart-phones, Apple's Siri, AI in computer games and self driving cars like the Google car [Google Inc., 2016] etc.

Many of us know about the existence of services and devices based on AI, but hardly anybody knows about the technology behind them. An important competence will be to use such smart tools in every day life. When programming a video recorder was a competence with which one could become famous in the whole neighbourhood in the 1980s, a skilled handling or even the "teaching" of intelligent tools might be its equivalent in the 2020s.

1.1 Motivation

Therefore it is of great importance to already familiarize young people in school with the technical background and the underlying concepts including algorithms, data structures and programming/coding. Like classic literacy which includes writing, reading and mathematics, literacy in AI/computer science will become a major issue in future. Furthermore, with AI literacy pupils also receive a solid preparation for subsequent studies at university level and their future career.

The remaining question is: where can one learn about basics or foundations of artificial intelligence – at least to a level of everyday use of intelligent or smart tools? E.g. in Austria currently even basic education about AI is mainly restricted to university programs. When looking at lower level education like high schools, one does not even find a single side remark about artificial intelligence in the official curricula of computer science classes. See [Bundesministerium für Bildung und Frauen, 2016] for details. The computer science education itself is officially compulsory only during the ninth grade for two hours every week. What several schools offer are optional classes that students can choose outside the official curricula or as a part of a module in an area of specialisation. Currently, only such optional classes would fit perfectly to host several lessons about artificial intelligence without having to adapt the official curricula. Hence, one future goal has to be to include AI topics in the general curriculum of computer science education.

1.2 Goals of this diploma thesis

We asked ourselves, if schools become more interested in education about artificial intelligence during the next years ...

- ... what contents are important enough to be fit into e.g. a class of one hour each week?
- ... how could such classes be structured?
- ... which suggestions already exist in the literature?
- ... what prior knowledge would be required to understand some foundations of artificial intelligence?
- ... what methods or tools could be used during such a class?

The goal of this diploma thesis is to work out answers to the questions above. But more than that, a prototype of such a class should be taught in one of our partner schools. The lessons will be evaluated to acquire valuable feedback of the first students that participate in this class. Building on this feedback a possible structure and the contents of such a class will be suggested and discussed.

1.3 Methods used

The work in this thesis comprises several different methods. First two literature researches will be done. On the one hand we have to find out about the current situation of computer science education in Austrian high schools. Therefore the legal framework on which we can set up a course upon will be evaluated. Then we want to find out which courses about artificial intelligence already exist. We have to distinguish between courses on high school level and e.g. university courses.

To be able to choose among the various topics an introductory course about AI could offer we also searched for literature. [Wollowski et al., 2016] did a survey in which they asked educators what topics of AI they are currently teaching, what practitioners are demanding and what topics both groups would suggest to teach. This forms the basis for the selection of our contents.

Based on the findings during the literature research we can start to plan a prototypic course about AI for an Austrian high school. We would like to focus the course on AI. Hence, we will restrict ourselves to high schools where at least some extended computer science education like a participation in the RoboCup [The RoboCup Federation, 2016] is offered and we can assume a certain level of programming skills. This prototypic course is conducted in one of our partner schools.

After the course, the lessons will be evaluated by the participating students. This feedback will then incorporated in the planning of the next version of this AI course. This concludes the first loop of a standard quality management cycle like the famous Deming cycle [Deming, 1986] of Plan – Do – Check/Study – Act.

Hence, this thesis makes several contributions to the area of AI education. We design a concept for an AI course in highschools, do a proof of concept implementation of it and also evaluate it afterwards. Some of the results of this project have already been published in the conference paper [Burgsteiner et al., 2016].

1.4 Structure of the remaining chapters

In the next chapter we will present the results of the literature researches. Based on this we can outline a possible extent of an AI class in high schools and its determining factors. We will discuss various didactic methods and how they can fit into our course.

Chapter 3 represents the main part of this thesis. It contains the results of the first planning sessions and the detailed outline of the complete course. In that chapter the golden thread that is woven throughout the course will be explained. Each class will be described in detail with a brief discussion of each topic and why it was chosen. For each class we will also describe a possible homework that can be given.

In the fourth chapter our experiences and lessons learnt during and after the the conduction of the classes will be described. That chapter also contains the elaboration of the questions for the evaluation of the classes and summarizes the feedback of the students.

The last chapter discusses the whole work and makes implications and outlines for possible future classes about artificial intelligence in high schools.

Chapter 2

Background and related work

Looking at the contents of curricula of typical Austrian high schools in [Bundesministerium für Bildung und Frauen, 2016] that are not specialising on a specific technical education, one does not even find a single side remark about artificial intelligence in the official curricula of computer science classes. Even worse, the computer science education itself is officially compulsory only during the ninth grade for two hours a week. The curriculum states that the essential purpose of the computer science education is "to teach fundamental knowledge of computer science and information technology to enable students to use this knowledge with confidence [...] to solve given problems."[1]

Additionally, the contentwise requirements in this curriculum demands students to be able to (shortened and translated from [Bundesministerium für Bildung und Frauen, 2016]:

- manage information and organise the individual learning and self-improvement with qualified software in practical life, thereby exploit existing information sources and different information visualisations based on the previous knowledge
- systematise and structure contents, summarize results and present them with multimedia technology
- establish and use a linked up information systems for the individual work

[1] Translated and shortened from the German original: "Es ist eine wesentliche Aufgabe des Informatikunterrichts, Schülerinnen und Schülern informatische und informationstechnische Grundkenntnisse zu vermitteln, um sie zu befähigen, diese zur Lösung einer Problemstellung sicher und kritisch einzusetzen."

- reach a confident handling with standard software for written correspondence, for documentation, for the publication of written documents, for multimedia presentations and for communication
- construct calculation models and asses and interpret the results; use a simple database
- gain insights in essential vocabulary and methods of computer science, its typical thinking and methods of operation, its historical development and get to know basic principles of automatons, algorithms und programs
- get to know essential methods and legal foundations of data protection, data security and copyright law and comprehend the impact of the use of technology on individuals and the society
- get to know possible applications of computer science in different occupational areas and thereby support an orientation towards possible later professions

One question that results from these demands immediately is: how can such a high goal be achieved with only two hours a week in only one year during the education of a student? What several schools offer therefore are optional classes that students can choose to visit outside the official curricula. Some schools also offer modules in several areas of specialisation that the students can select (so called "elective compulsory subjects"). Depending on the general orientation of the school these modules or special courses also sometimes include topics like general computer science, programming, and robotics.

Also when looking at international literature, teaching the fundamental concepts and techniques of AI independent of any platform or programming language at school level is quite rare. Many approaches focus on undergraduate/graduate students at university level. [Wollowski, 2014] describes a research oriented course for undergraduate students in which the IBM Watson was used to teach AI fundamentals. [Torrey, 2012] suggests ways to structure lessons and assignments for undergraduates to teach them basic problem solving strategies and skill with an additional focus on pedagogical methods. In [McGovern et al., 2011] a course that has been running for six years is presented, where students have to use Java to program suitable agents for certain games. This course is again for undergraduate and graduate students. Combinations of programming AI and using LEGO robots can be found in [Selkowitz and Burhans, 2014] where undergraduate students are encouraged to build an autonomous chess agent with a robot that moves real chess figures. [Kumar and

Meeden, 1998] report about the design of a robot lab to be used in introductory AI classes. They argue that students can learn many central issues of computer science including the interaction between hardware and software better with the use robots. Some universities also offer training courses for future teachers about how AI can be taught in school, like in [Dilger, 2005], who also discusses topics like problem solving, planning, robotics and learning system.

Common approaches at school level typically deal with more general areas of AI like its history, the Turing test [Turing, 1948] or chat bots. [Heinze et al., 2010] report about a course in Australia that introduces pupils to AI by doing experiments with real robots or discussing philosophical question concerning AI on several levels for kids of different grades. A highly sophisticated single event project at high school level is reported by [Fok and Ong, 1996] where two students use an industrial robot arm to train a feed-forward neural network to play Tic-Tac-Toe. This course did focus on the promotion of young talents with a small selection of technology and did not have the aim to teach AI in general. [Featherston et al., 2014] use a developed special educational platform called Dorothy to teach robotics and core computing skills to students without prior programming experience. The tool offers a drag and drop interface to create graphical routines for robots in virtual worlds and also a possibility to generate code for real-world robots. It is not outlined how this platform can or should be used for AI classes.

2.1 Competencies

A modern approach to education is the use of so called "competencies" the students should acquire during their times of study instead of traditional facts that they should "know" and be able to reproduce during a test. A competence is defined as the ability of an individual to do a job properly. This is in fact more important than just e.g. to be able enumerate several facts that have no meaning to an individual. Typically, competencies can be described with sentences beginning with "I can ...", like "I can evaluate the efficiency of algorithms."

Current educational reforms in many countries try to develop curricula and exam regulations that try to emphasize the necessary competencies that the students must acquire. This is being done for most fields of studies (sometimes even with inter-disciplinary competencies). There also exists a competencies based draft of the contents of computer science education in Austria [Bundesministerium für Bildung und Frauen, 2013]. This

draft defines competencies that have to be achieved when a students selects computer science as an elective compulsory subject with an extent of at least additional 6 hours during the last 3 years in high school.

The law concerning the regulations of oral exams during the final high school exams defines in [RPVO, 2012, §29] that every competencies based task must contain the following aspects:

- aspects of reproduction
- aspects of knowledge transfer
- aspects of reflection and problem solving

The contents of the subject are defined via the curriculum. These contain the areas of "Computer Science, Human and Society", "Information Systems", "Applied Computer Science" and "Practical Computer Science". The latter area contains also a sub-area called "Intelligent Systems". To be able to define concrete competencies a working group defined relevant dimensions of actions. The dimensions of actions can be divided in the areas "knowing and understanding" (aspect of reproduction), "use and design/create" (aspect of knowledge transfer and problem solving) and "reflect and evaluate" (aspect of reflection).

In the intersection of these dimensions of action and the contents of the area "Intelligent Systems" one can find the competencies that the working group defined:

- Knowing and Understanding:
  - I can describe areas of application in which computer systems behave intelligent.
  - I can explain the difference between human and artificial intelligence.
- Use and Design/Create:
  - I can use intelligent information technology.
- Reflect and Evaluate:
  - I can compare and assess aspects of human and artificial intelligence.

To us, these are quite fuzzy descriptions and definitions that do not yield precise contents of possible courses. To design lessons in which some foundations of artificial intelligence are taught one has to select concrete subjects. Especially the number of competencies has to be expanded to allow for more detailed definitions that correspond e.g. with more concrete goals of the course.

It doesn't get any better in terms of clarity when one takes a look at descriptions for the competencies in the area of algorithms, data structures (sub-areas that play an important role in AI) or programming. The competencies defined are merely (again from [Bundesministerium für Bildung und Frauen, 2013]):

- I can explain the term "algorithm".
- I can describe tasks and problems algorithmically and formally using suitable data structures.
- I can explain essential aspects of the procedural, functional and object-oriented programming with the help of examples thereof.
- I can model tasks through the use of computer science.
- I can develop algorithms and visualise, implement and test them.
- I can assess the efficiency of algorithms.
- I can efficiently search for and correct errors in computer programs.

We will try to define more concrete competencies for the areas of artificial intelligence and data structures that suite our planned course and reflect the previous knowledge of the students in the next section.

2.2 Formulating goals and competencies for our course

Based on the possible contents found in [Wollowski et al., 2016] and our intentions of teaching foundation of artificial intelligence we can begin to formulate goals and competencies that should be reached during our course. Due to the fact that robotics can play a important role in motivating students to engage in computer science [Altin and Pedaste, 2013], it is obvious to work mainly with methods and examples originating from

this area of application. Important for us are not only competencies in artificial intelligence, but also to show the connection to important topics of classic computer science like data structures, to be able to explain and possibly program algorithms (e.g. searching).

Resulting from these considerations are the following goals and competencies for our course from the areas of "artificial intelligence in robotics" and the important foundations in "data structures":

- I can explain reactive agents and its difference to intelligent systems.
- I can design simple reactive and model-based robot controllers.
- I can apply fundamental data structures like stack, queue, graph and tree and explain some typical areas of applications.
- I can apply different search algorithms in trees and graphs (e.g. breadth first and depth first, greedy, $A^\star$) and explain their advantages and disadvantages.
- I can explain classic planning mechanisms of AI agents like forward and backward search and apply them with paper and pencil.

2.3 Participating school and its students

We found a school that is very interested in offering some of its students a class in the area of artificial intelligence. The [BRG Kepler, 2015] is an Austrian high school located in the city of Graz that has a focus in natural and computer sciences. Together we offered a totally voluntary, additional class on Friday afternoons. The group of nine students that subscribed to this special course comes from different age groups from the 9^{th}, 10^{th} and 11^{th} year of school attendance (i.e. approximately between 15 and 18 years old, average age 16.5 years). Hence, they also had a very varying previous knowledge in the area of computer science (none of them in AI). But additionally all of them are participating for several years in robotics electives, especially in various leagues in the [The RoboCup Federation, 2016]. Amongst them are also winners of the Austrian RoboCup championships and participants of the world championships. According to this we are assuming a high motivation and knowledge beyond the traditional contents of school education in computer science. There were 1 female and 8 male students attending.

Preliminary talks with some students and teachers showed, that the programming skills can't be described as very proficient or structured. They describe their code themselves as "spaghetti code". What's missing is e.g. an abstract description of the controllers of their robots and accurate strategies for the control.

This first analysis gives us several inputs how our course could be conducted especially in this participating school. For a stronger motivation the own robot that has been used during the robotcis contests will be used. Considering the previous knowledge of the students and the very limited time the following topics might be suitable:

1. Clarify how the self-built existing robot is working at the moment. Which mechanisms and sensors are used? How are the sensor values evaluated. How are the decision processes designed based on which data?

2. What is a state in the sense of an automaton? What are state transitions? What is a finite state machine and how can it be modelled graphically? How can a robot's decision be modelled with such states? How do state transitions relate to behaviour?

3. What is intelligence and on which properties of a living being can it be observed? Are the own robots intelligent according to theses measures? What are they possibly missing to be called intelligent?

4. Possible intelligent agents: reactive controllers (e.g. [Braitenberg, 1984]), reflex agents, model-based reflex agents, goal-based agents, teleo-reactive systems [Nilsson, 1994], subsumption architecture [Brooks, 1986]. Each including examples of usage, differences and limits of the specific approach.

5. The data structure of a graph including directed and undirected graphs. Example how to represent a labyrinth as a graph.

6. Modelling the search of a route in a labyrinth. Introduce trees as a data structure.

7. Problem solving through searching in graphs or trees, introducing queues and stacks. Breadth first search and depth first search, greedy search, $A^\star$-search. Comparison of the search algorithms and discussing optimality.

8. Planning of actions, classic planning.

2.4 Learning theory and legal requirements

Computer science is basically a formal science. During its education it is therefore important to organize the contents in a way that it has a direct connection to the prior knowledge and experiences of the students. Ideally, the topics and areas of application meet the interests and hobbies of the students.

[Bundesministerium für Bildung und Frauen, 2016] also gives many suggestion how to organize a computer science course. It recommends varying types of work like solitary work, group work or team work to enable students to research new contents and apply them in different communicative settings. The importance of joint problem solving has to be considered especially in computer science education. The students should also get the opportunity to ensure their learning progress through connections and knowledge transfer to areas of life. The authors also suggest to use several methods for a higher motivation of the students and a better benefit of the classes. This includes possibilities to present their knowledge or work and thereby to confront themselves with critique and learn to argue their work. The design of a pleasant and successful climate for learning is based on trust, the promotion of individual strengths and the creative potential. Additionally [Bundesministerium für Bildung und Frauen, 2016] demand to react to possibly different requirements of girls and boys.

2.4.1 Self-directed learning and Constructionism

Seymour Papert plays a crucial role in both AI and computer science education. In his book [Papert, 1993] he describes the idea behind the earlier programming language called LOGO and how young students can use it to explore fundamental ideas in computer science [Schwill, 1997], like recursion or the principle of divide-and-conquer. The children start with very simple but yet exciting problem like drawing triangles or squares with a virtual turtle. During the construction of more complicated geometrical drawings they learn e.g. how to reuse and combine simpler methods to achieve their goal.

Papert and others are also advocates of the learning method they called “constructionism”, in which they oppose the classic frontal, teacher based learning concepts, where teachers give their knowledge, their wisdom to the children. Moreover should the learners be enabled to explore the knowledge themselves and thereby construct new knowledge

built upon their past experience. This concept has also been studies in [Alimisis and Kynigos, 2009] where the concept of constructionism was combined with robotics.

According to [Spohrer, 2009] learning is more sustainable when the students get eager or curious to know something. Therefore authentic problems must be presented which lead to gaps in the current knowledge. These gaps then require the construction of new knowledge. For example students involved in RoboCup can see that it is difficult to make a plan to let their robot escape from an arbitrary labyrinth. Showing them connections between graphs, searching and labyrinths, the motivated student will probably start to research how such an algorithm works and how it can be implemented.

In such active learning scenarios, the learners should be able to recognise and maybe also to reflect their own thinking and learning processes. The goal of this process and goal based approach is to promote self directed learning and hence not only to learn certain facts but to directly experience possible life-long learning strategies.

Following these concepts one premise of our course is to actively involve the students in the learning process. Activities can include e.g. paper-and-pencil or programming exercises, robot construction, discussions, group works etc. depending on given tasks or problems. Therefore the contents have to be adapted and set in context with the pupils' prior knowledge in robotics and programming. A course in a school that actively participates e.g. in junior robotics competitions can include assignments on the basis of the existing robots and code and let the students see possible shortcomings or possibilities for enhancements of their current approach. This way we try to point to ways how AI could eventually increase the performance of their robots in the contests.

2.5 Evaluation of learning processes

Current methods to control learning processes and to evaluate what students should have learnt include so called learning objective controls. These are being used in several schools in Austria and are sometimes also used instead of classic school certificates based on grades. In doing so goals and sub-goals that can be achieved are verbalised in advance. The degree of achievement can also be nuanced. This yields e.g. a three valued grade system for each goal or sub-goal like "achieved", "not achieved" or "partly achieved".

The first step – after defining the content of the lessons — is a precise definition of the learning goals and/or sub-goals. According to [Stangl, 2016] there do not exist suitable proceedings to measure the success of actual learning – when success is being defined as reaching a certain learning goal. Therefore one has to construct suitable examples or problems that can be solved with the knowledge gained from the lessons for measurement purposes. The success or failure in solving such an example must not simply be assigned with points or transformed to a grade. Instead the concrete actions of a student should be observed to be able to assess which contents or competencies were not, partly or fully achieved.

The schedule of a learning goal assessment can consist of the following steps:

1. Define the learning goals (including the sub-goals and possibly the steps leading to them).
2. Formulate the expected competencies that have to be achieved when reaching the learning goals.
3. Create examples or problems for assessment of the individual learning steps.
4. Interpret the results (by asking and/or observing).
5. Adapt or redesign the teaching concept if necessary.

As also [Stangl, 2016] writes, the actual learning processes remain hidden and the learning results of individual students are different. But because the current learning results influence the future starting point of new learning scenarios, it is important for the teachers to know something about these previous learning results. The problem is that these are not directly accessible. We can only get hints by observing the ability of students to solve some real and new examples. It is nevertheless difficult to get a feedback for the design of the lessons from this. Even if a student is able to solve the given problem, we do not know when, how or where he or she acquired the competency. But also if one is unable to solve a problem does this not immediately imply a lack of learning.

Concluding one can say that it is possible to test the ability to solve a problem, but an assessment of a learning does not reliable prove a learning success. In most cases we only assess an ability and interpret it in terms of competency and learning. In our case and in our design of the course we will restrict the assessment to a self-evaluation of the students and an accompanying observation of the teachers to get feedback about

the learning processes. Additionally the course itself will be evaluated by the students. Details about these evaluations can be found in Chapter 4.

Chapter 3

Design of the classes

This chapter will describe the preconditions we had in planning the course about artificial intelligence and the considerations that led to the concrete structure and contents. After that, the individual classes will be described in detail, including the detailed planning of the classes as well as the accompanying experiments and assignments.

3.1 General Outline

[Wollowski et al., 2016] did a survey of current practice and teaching of AI. This survey asked educators what they are currently teaching in various AI courses and practitioners what techniques they use in practice. According to the educators the primary goal of current AI courses is to teach the basics or main ideas of AI. The views of the practitioners do not differ here. The recommended topics of the educators list "Search" and "Knowledge Representation and Reasoning" (KR&R) in the first places, followed by "Machine Learning". Also the practitioners do say that KR&R is the most wanted topics in AI, followed by "Machine Learning". These topics will form the basic outline of our course, although machine learning will certainly only be coverable as a sort of overview or outlook, because the variety of different approaches and the necessary mathematical skills demand much more time and background knowledge.

Due to these expected limitations in the students prior knowledge for this course, we can only cover introductory topics of artificial intelligence. We have to include some necessary general topics like data structures (e.g. trees) or fundamental algorithms (like searching) as well. We limited the general foundations to those that are really necessary to understand basic topics of classic artificial intelligence and some applications thereof

in the area of robotics and e.g. especially in RoboCup, which is a very popular topic in Austrian highschools. We recommend that schools applying this course outline in their curriculum should already have some experiences in RoboCup since to students this is a very interesting, motivating and fun platform to work with.

The extent of topics is further limited due to the time restrictions and the planned methods of teaching. We plan not to use a classic ex-cathedra teaching but an approach consisting of a lot of self-conducted experiments, exercises and research by the students for at least some of the topics. A constructionistic approach like outlined in Section 2.4.1 demands a lot of time. Even more time would be needed to implement a self-directed learning scheme. This could only be suggested to the students in form of experiments and assignments where they can use the opportunity to deepen their knowledge within one or more of the topics considered during the lessons.

Finally, we also tried to find a single book that possibly comprises all of the topics we came up with. This book should also be comprehensive enough to serve the interested reader as a source for continued reading and deeper understanding of the topics. Additionally it should be usable as a study book, including e.g. exercises and scientific references. We found that the classic text book from [Russel and Norvig, 2012] fulfils all these requirements (it is also available in a translated german version which helps german speaking students to understand its contents). Hence, we used this book as the fundamental text book accompanying our course.

A typical voluntary class consists of one hour of teaching each week during a semester. Due to public holidays this typically results in about 14 lessons per semester. Since 50 minutes of teaching are very short, we grouped these 14 lessons in double units, resulting in a seven weeks course. The planned final structure and the detailed contents of the seven classes can be found in the next seven sections. Additionally, the golden thread and the concrete contents of our course are illustrated in Figure 3.1.

Each subsection first contains a general description and definition. The interested reader can also find more theoretical backgrounds on each of the topics, some examples and applications in [Russel and Norvig, 2012]. We then structure the content for each class and describe methods and examples how the topics can be presented to and motivated for the students. We also suggest social forms how we think the topics can be explored by the students and possible practical assignments for further studying and self-directed learning of the students. Many more social forms that can be helpful for teaching can be found in [Meyer, 1987a] and [Meyer, 1987b].

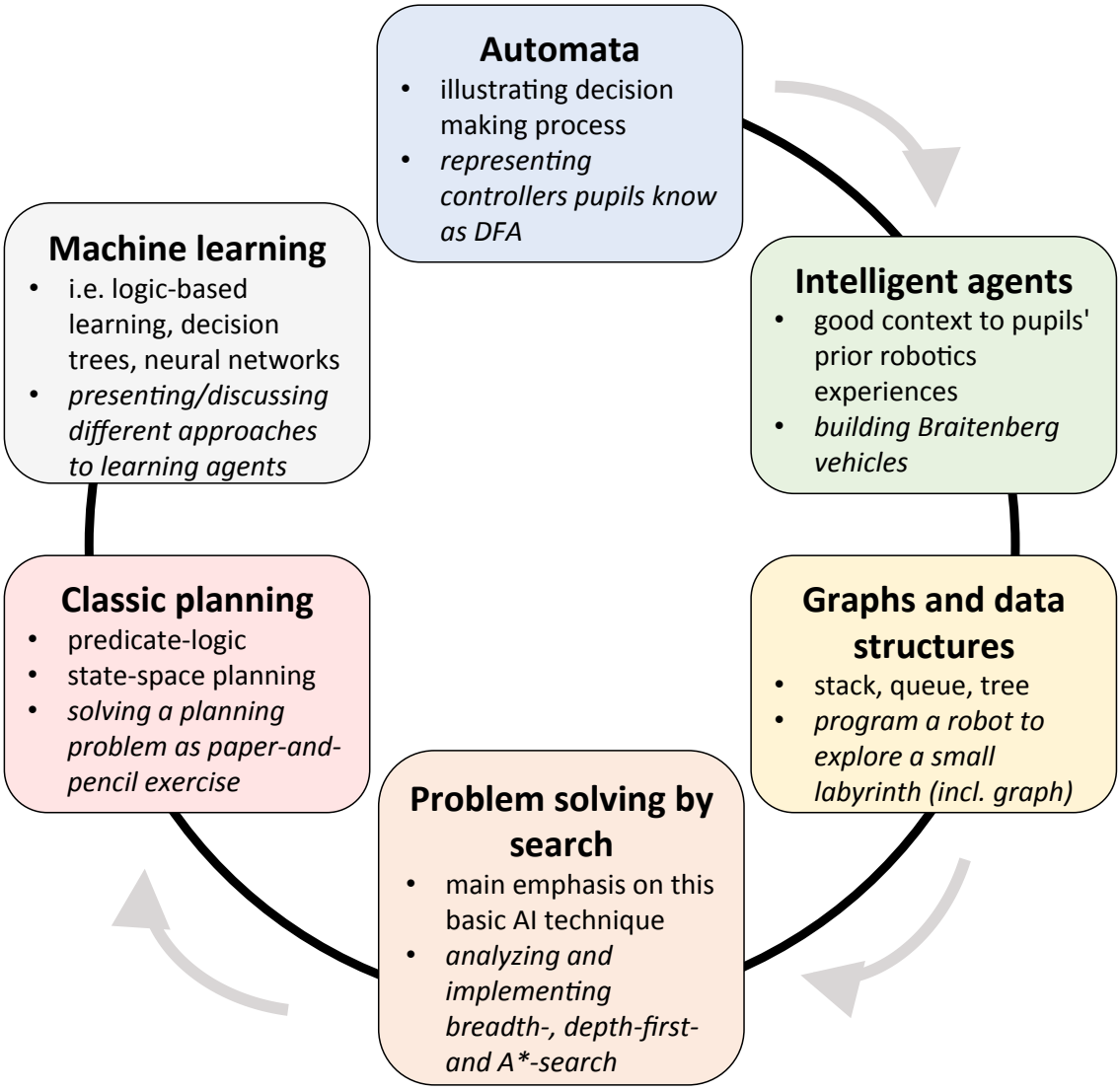


Figure 3.1: An illustration from [Burgsteiner et al., 2016] of the various AI topics covered in our introductory course. The topics were spread among the seven classes we held. The assignments are written in italics.

3.2 First class: Introduction and Automata

The idea for the first class is to get to know each other, to outline the course, the reasons of being here and the way we will work together for the next weeks. After this, the first basic terms of "intelligence" and foundations thereof will be discussed.

This is followed by a more philosophical discussion about the term "intelligence" incited by the question "What distinguishes a fly from a toaster?". To which properties of an

object or creature can intelligence be pinned to? Aimed to construct an individual answer what intelligence is to the students. This should help the students to answer the question whether their own robots can already be regarded as intelligent. What should they maybe additionally be able to do to be intelligent? What are they still missing? Also a different approach to intelligence in form of the test proposed by Alan Turing [Turing, 1948] will be discussed.

Answers to the questions above might include the identification of a sort of memory that is necessary to produce a perceivable intelligent behaviour. This leads to different forms of memory. The simplest form of memory that is available in computer science are the different states of finite state machines. We will first explore the theoretical components of automata like: what is a state in the sense of automata theory? How does an automata transfer from one state to another? What is a deterministic finite automata (DFA, sometimes also called deterministic finite state machine)? How can one graphically model a DFA? The DFA will not be introduced in the meaning of an automata that accepts or refuses words of a certain language as in theoretical computer science. It will be presented in a (simplified) way in the sense of constructivism, so the students might understand it better: depending on their concrete past experiences e.g. as a controller that is able to operate a machine or a robot, like they know from the RoboCup.

Automata like deterministic and non-deterministic finite automata, timed automata and input/output automata play a crucial role in current research too. They additionally form the basis for various applications in automated testing, verification and diagnosis of models. See e.g. [Baier and Katoen, 2008], [Utting and Legeard, 2006], [Sampath et al., 1995] or [Cassandras and , 2008] for more details on the importance of automata in various applications.

As a first example we construct a simple DFA step by step together. As an example we look at a possible control of a vending machine like in Figure 3.2. This example can be modified to e.g. accept a wider range of coinage or different sorts of beverages. After this joint construction and discussion, the students have to think about a similar simple control for themselves. In a pairwise work or in a group of three they should try to design such a simple control and thereby answer the following questions:

- What different states are necessary to implement a working control?
- In which state will the DFA start when it is switched on?

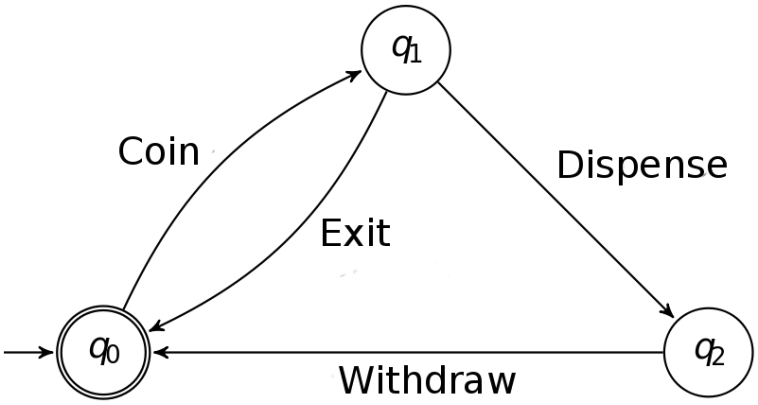


Figure 3.2: A simple finite state machine that shows a possible control of a vending machine as the starting example of automata theory. After insertion of a coin (state q_1) it is possible to still exit the buying process or continue to dispense e.g. a bottle. While the collection tray is still blocked (state q_2) no further bottle can be bought. After the withdrawal of the bottle the vending machine is ready for another purchase (state q_0).

- What sensors or buttons (inputs) does it have? Which actuators (outputs) are available?
- Which state transitions are possible and allowed? In each state, what effects (state transitions and/or outputs) will all possible sensor readings have?
- After which sequence(s) will the controller be in its starting state again?

The students should have enough time to construct such a DFA. Possible questions or occurring problems from the students are discussed and answered. Afterwards, each student presents their work on the whiteboard and explains their thoughts during construction to the others. We discuss the DFA together and talk about possible extensions or improvements.

At the end of the first class an assignment is presented. The task is similar to the example we do during the class but refers to the robot the students should have been working with e.g. during the RoboCup. Here comes the constructionistic approach into play: the students should create something real on the basis of what the have heard so far and compare it to their knowledge and experience hitherto. Again the student shall work together in groups, preferable the same groups they formed during the RoboCup. The questions to solve are:

- What decisions did you make during the programming of your robot?

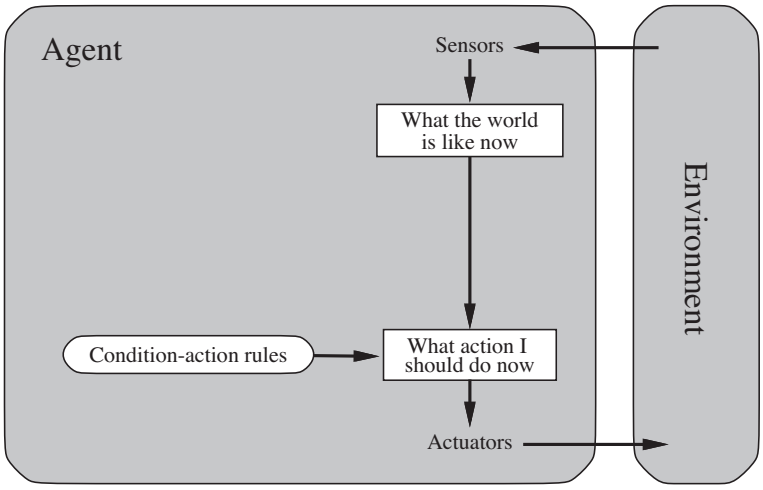


Figure 3.3: A simple reflex agent that has no internal memory of recent events. Reacts directly to external stimuli (from [Russel and Norvig, 2012]).

- How could possible states of your robot be characterised? Can appropriate names be found for each state?
- What sensors are being used? What do they actually sense?
- Which events, sensor readings etc. can occur?
- Due to what circumstances does the state of your robot change?
- Which state lead to which other states?
- Draw a possibly complete state diagram of the controller of your robot!

The students are asked to design and create a poster until next class. We will review and discuss their results at the begin of the next class. According to our schedule they have two weeks time.

3.3 Second class: Intelligent Agents

The second class starts with the presentation of the previous assignment. Each group presents their robot controller one by one. The presented solutions and possible difficulties are discussed. This is also a good possibility to review the lessons learned from the previous class that we need for this one.

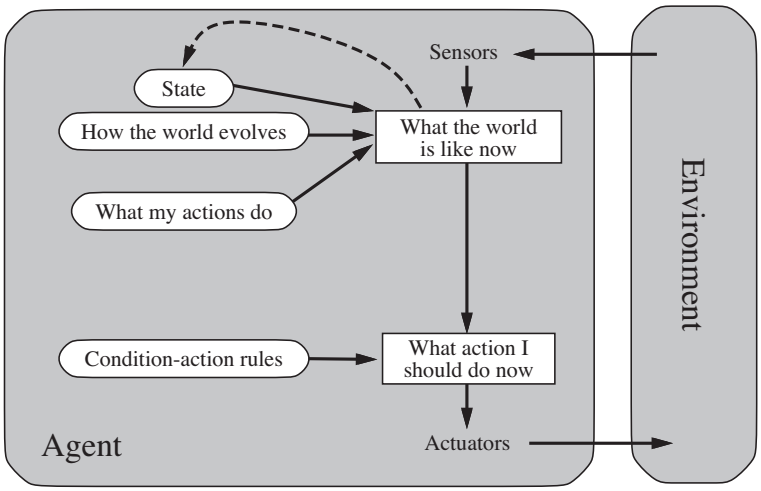


Figure 3.4: The model based reflex agent include an internal memory. It knows about the outside world and can react to stimuli based on their knowledge of the world (from [Russel and Norvig, 2012]).

To be not too theoretic we also take a look at some practical considerations. First, we discuss how such DFAs could actually be programmed in C++ or Java to find out what ideas the students already have and – since we do not know the students that good – what they have learned in other computer science lessons so far. Questions include e.g.: How can a (complete) DFA be programmed easily? What data structures do we need for that? What basic code structures would be recommendable?

Simple methods like two dimensional arrays with the size of the number of different states times the number of different sensor events would be possible solutions. With this method one can simply use such an array as a look-up-table to decide from which current state the DFA changes with which sensor events.

This also leads to the discussion about possible drawbacks and limitations of this approach. First, one has to model every theoretically state change not only those that could occur in practice. And second, this two dimensional arrays can get huge easily, when different sensor events can also occur concurrently (e.g. a distance and a light sensor of a mobile robot could detect objects simultaneously). When these combinations also play a crucial role during the decision to which state to change, the number of sensor event combinations and hence the size of the look-up-table grows exponentially. Other approaches to programming simple DFAs like the usage of enumerated sets of states in combination with switch-statements should be discussed and compared as well.

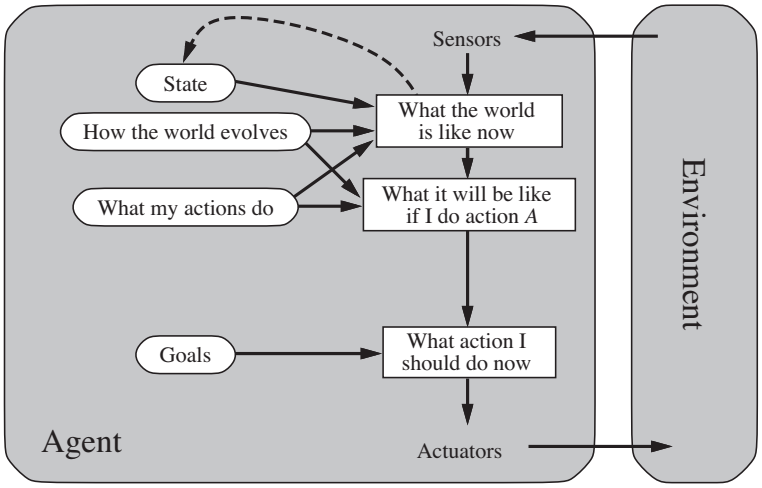


Figure 3.5: An illustration of a goal based agent. These agents can decide what action to take based on simple reasoning about the outside world. They typically select actions that bring them nearer to a defined goal (from [Russel and Norvig, 2012]).

With this background knowledge about states and state transitions we can advance to some basic structures of agents that are also described in [Russel and Norvig, 2012]. These basic agents display methods of internal organisations of intelligent agents that are still subject to research in AI. There still doesn't exist any kind of "golden model" that researchers agree on being superior over all others. Structures give possible answers to the important question how intelligent behaviour could be organised or internally represented.

These basic agent structures include with increasing complexity:

1. Simple reflex agents: conditions resulting from sensor readings trigger actions directly without any internal memory of past actions or events. See Figure 3.3

2. Model based reflex agents: these agents include a simple model of the outside world and "know" what their actions trigger. See Figure 3.4. The DFAs we heard of are examples of such agents.

3. Goal based agent: Agents of this type not only have a memory of recent events and action but also can reason about the outside world. They can infer what happens when they select a specific action. This way they can decide whether an action brings them nearer to a defined goal or not. See Figure 3.5. Details of these agents will be discussed in the next lessons when we talk about searching and planning.

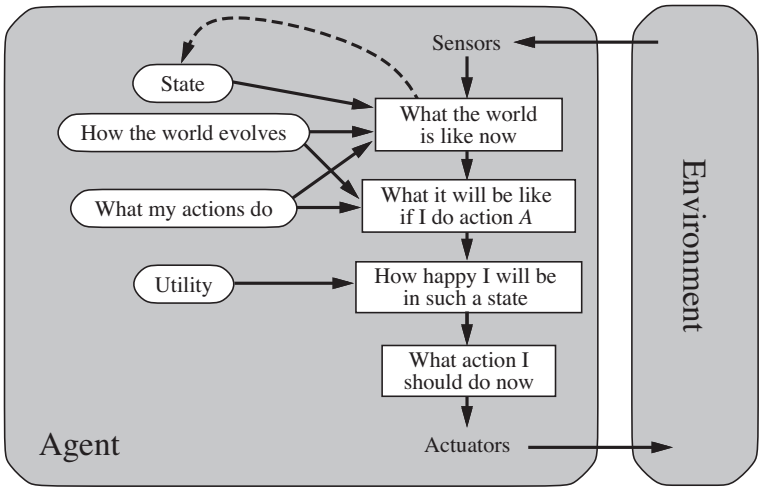


Figure 3.6: A utility based agent is an advanced version of a goal based agent that also incorporates a utility function (from [Russel and Norvig, 2012]).

4. Utility-based agent: A utility based agent is an advanced version of a goal based agent that also incorporates a utility function. Typically, a goal can be reached in several ways. The utility function additionally gives a feedback about how "happy" an agent is when it takes a certain action. This way the agent can maximize its happiness on its way to reach a goal. See Figure 3.6.

5. Learning agent: The agent in Figure 3.7 shows a common structure of a learning agent including the typical blocks "critic", "learning element" and "problem generator" that are necessary regardless of the actual machine learning algorithms used in the background.

Other approaches of the organisation of intelligent behaviour outside of these classic agent models are e.g. Brooks subsumption architecture [Brooks, 1986] or cognitive architectures like in [Laird et al., 1987]. A special case of simple reflex agents is well suited for a practical experiment during the lesson: Braitenberg vehicles [Braitenberg, 1984]. These vehicles have direct connections from sensors to actuators like in Figure 3.8. Typically light or distance sensors are used in experiments. The actual wiring decides about the behaviour of the robot. Braitenberg vehicles can show a broad range of quite complex behaviours despite their simple architecture when one varies the wiring, the connections strengths, combines multiple sensors and/or incorporates non-linear coupling between sensors and actuators. After a short introduction the students are motivated to take one of the LEGO Mindstorms robot at hand in the computer lab to quickly program a Braitenberg vehicle

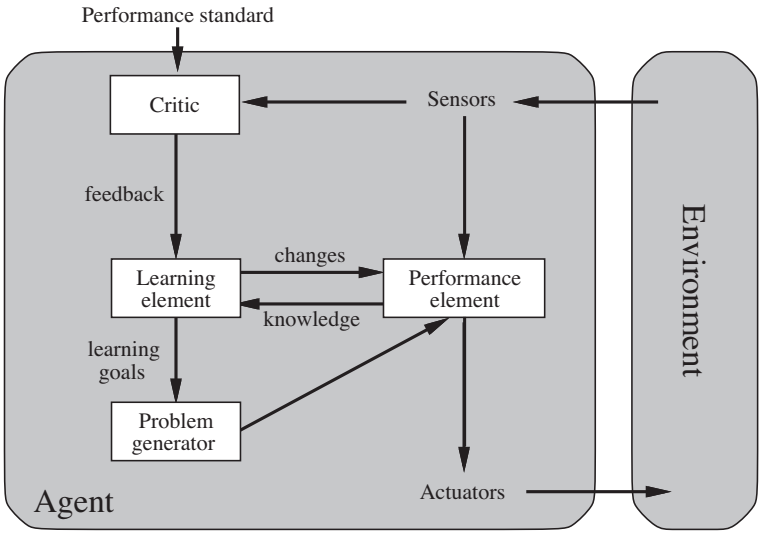


Figure 3.7: A learning agent is able to optimize its behaviour itself. It utilizes a critic that gives feedback about the current behaviour to adapt the actions taken in various situations (from [Russel and Norvig, 2012]).

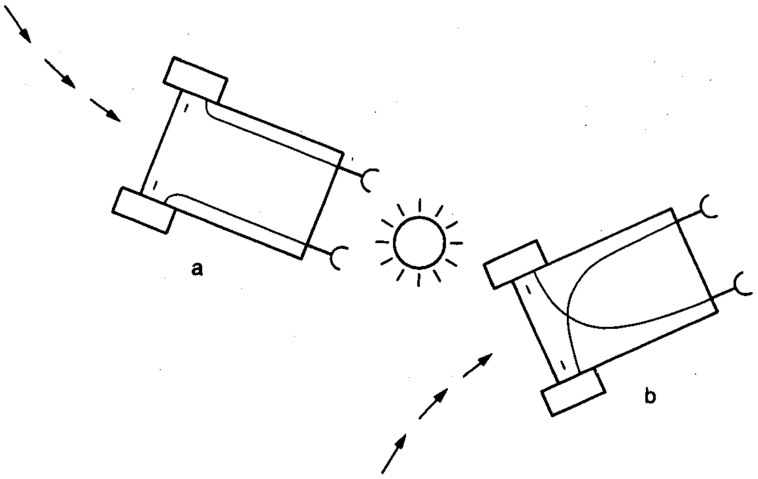


Figure 3.8: Braitenberg vehicles are examples of simple reflex agent that have no memory of recent events. They typically react directly to external stimuli like light or sound. The way in which the sensors are connected to the actuators decides about the behaviour. It can be seen that a simple crossing-out of the cables turns a light follower into a light avoider (from [Braitenberg, 1984]).

of their choice and to experiment with different settings. The different behaviours are then briefly presented by the students, compared and discussed.

The class concludes with the assignment for the next or next but one lesson. This time the students are asked to try to program a model based agent of their like with the LEGO NXT or EV3 robots available in school. They are free to choose any behaviour or task they feel comfortable with or that they are interested in, like e.g. from the junior robot leagues.

3.4 Third class: Problems and Graphs

At the beginning of the third class the students have the possibility to present their model based agent of the previous assignment that they might have programmed.

In this class we have to deal with some theoretic foundations that we need for the rest of the class. These include the definition of a "problem" in artificial intelligence and some more advanced data structures that the students might not have learned of in school until now, like graphs and trees.

In theoretical computer science many real world problems can be reduced to graphs. These include famous problem like the travelling salesman problem or Euler's bridge problem from Königsberg or the Hamiltonian path problem. Solutions to these problems can often be found be searching within those graphs starting from a certain vertex. In general, solutions to such problems can be very hard to find and are sometimes even NP-complete problems. This means that solutions can only be calculated with non-polynomial time consumption although a given solution can be tested in polynomial time.

Finding solutions for problems defined as graphs have also relations to general problems whose solutions have to satisfy a number of constraints or limitations. These are known as CSP ("constraint satisfaction problems") or SAT ("boolean satisfiability problem") that are being used today again for diagnosis or verification of models or for planning purposes. See e.g. [Biere et al., 2009] or [Dechter, 2003] for more details.

We start with the definition of a problem like in [Russel and Norvig, 2012, 101] consisting of

- a set of states: what different states can occur?
- a starting state: in which state does the robot start with?

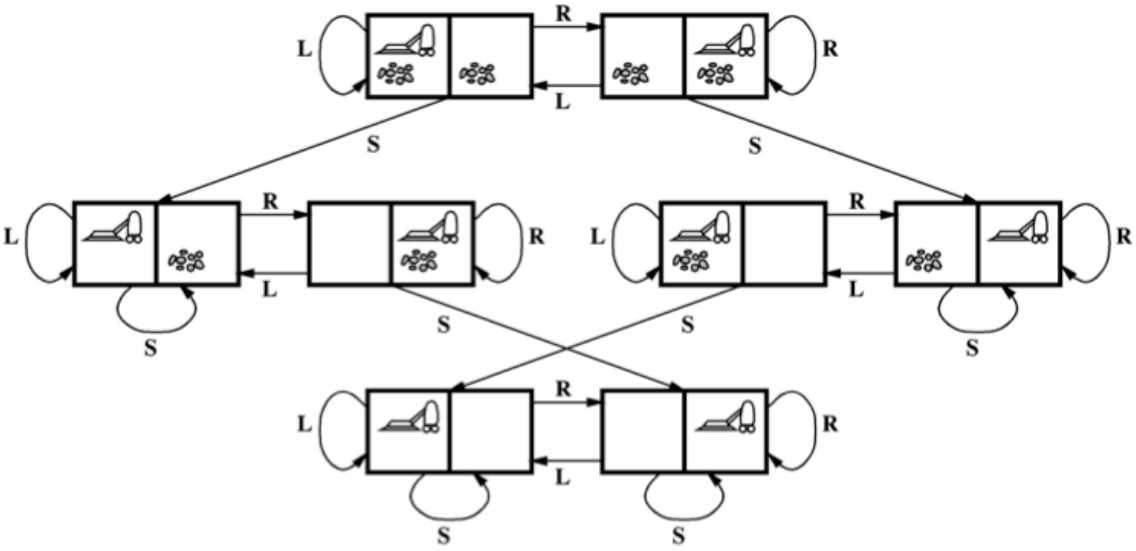


Figure 3.9: Illustration of a very simple example of a problem from [Russel and Norvig, 2012]. The world consists of only two rooms that can either be dirty or tidy. The robot can only move between those two rooms and suck the dirt. The starting state could be any of the possible states. The goal state is a totally clean world.

- a set of actions: what actions are basically available to the robot? The subset of possible actions then depends on the current state.
- a transition model: descriptions what each action in a certain state causes.
- a goal test: gives feedback about whether the agent has reached a certain goal yet.
- cost function: a way to define the effort of a certain action. With such a function an agent could optimize its plans how to reach a certain goal.

A simple example that can be used to introduce these ideas are e.g. the vacuum world from [Russel and Norvig, 2012] as seen in Figure 3.9. Here, the world of the vacuum robot consists of only two rooms that can either be sensed as tidy or dirty and the robot has only three actions (left, right, clean). The states are thus defined as all possible combinations of dirty/tidy rooms and the position of the robot. It is good to illustrate the steps necessary to solve a given problem. We can start in any of the given states. The goal is to clean every room. The actions the robot has to do to reach its goal is simply to apply appropriate actions to get from the starting state to the goal state. Hence, the solution to a problem can be seen as a walk in this state diagram.

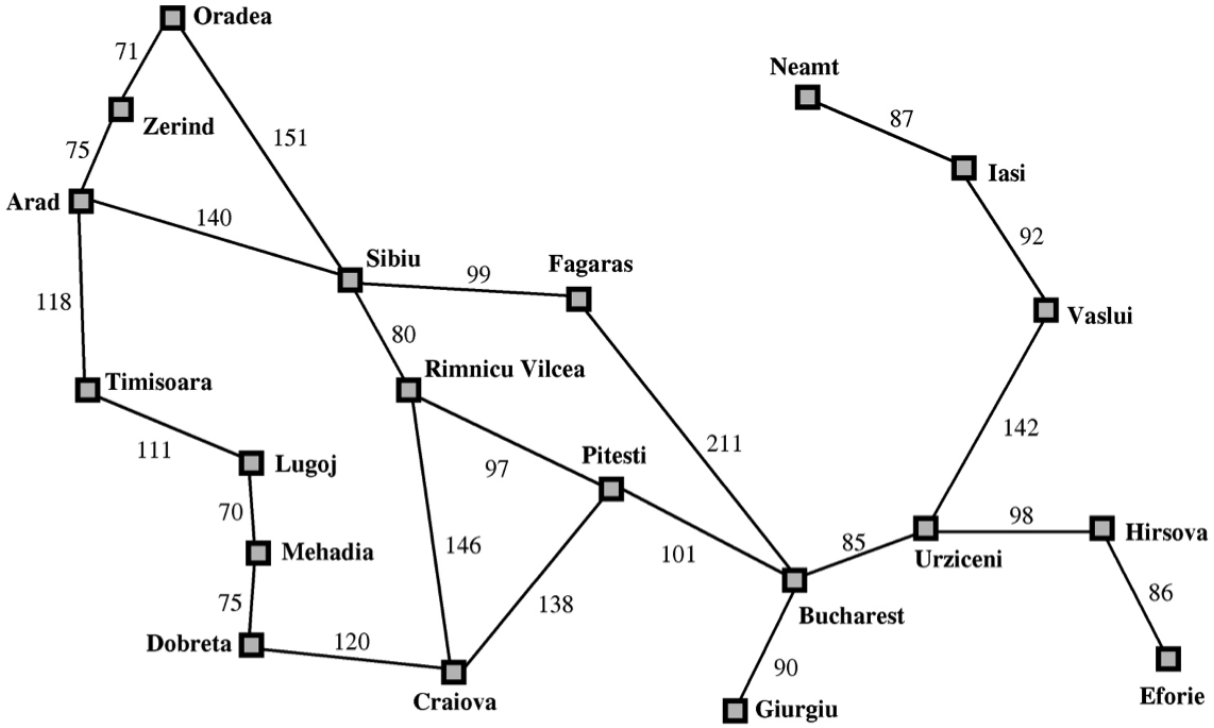


Figure 3.10: A simplified version of the street map of Romania is the starting point for several problem definitions. Problems are typically assignments to get from a specific town to another one. Costs of actions are introduced as the travel distance between connected towns. Hence, some of the solutions, i.e. different paths taken, can be better than others in term of total travel distance (illustration from [Russel and Norvig, 2012]).

Another classic problem that will be introduced here because it is used in some of the following classes as well is the problem of travelling in Romania that can be seen in Figure 3.10. Here, a problem might be defined as getting from one of the towns to another. Additionally, we introduce the term of the "cost of an action" that can be figuratively seen as the travel distance between two connected towns. So different solution to the same problem, like different ways taken, can have a different total cost. Hence, some solutions are better than others in terms of the total travel distance.

Again this real world problem definition can well be transferred to the computer science language (to the terms of a DFA and/or the data structure called graph that will be introduced shortly):

- Each town corresponds to a state or a vertex.
- Each connection between two towns is like a state transition or an edge.

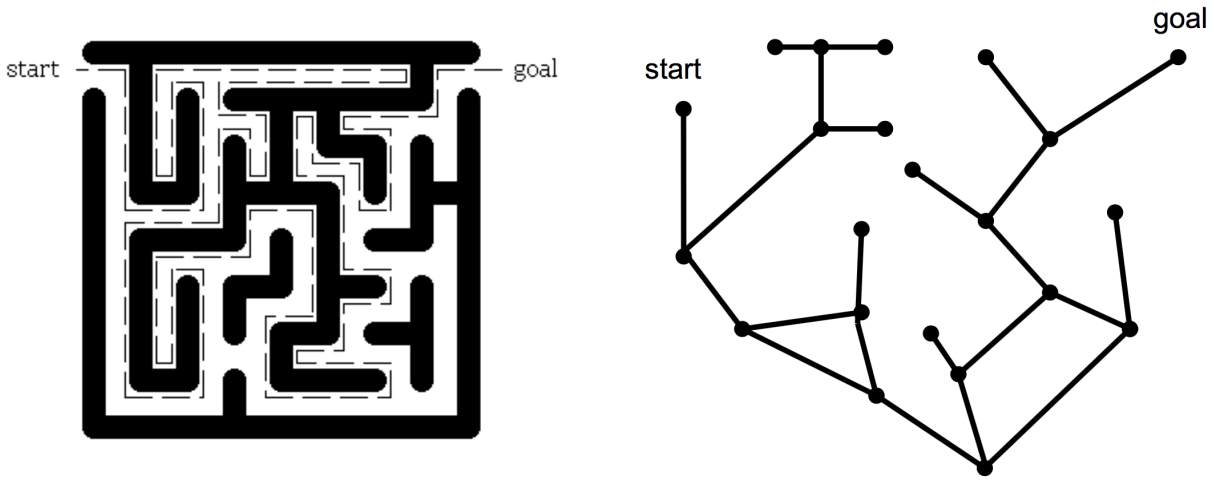


Figure 3.11: A labyrinth can also be seen as a graph. A labyrinth on the left side and its abstract representation on the right (illustration from [Futschek, 2012]).

- Each town could be used as a starting point for the trip like a starting state in a DFA.
- The end point of our trip corresponds to the goal state the DFA tries to reach.
- Connections (edges) can be directed ("one ways") or undirected.
- Connections can have different lengths (in kilometers) or weights.

With the help of this example also the data structure "graph" is introduced. During this class it is not necessary to get into more details about the data structure itself. Many implementations in the students preferred computer language can be found in the internet. Concrete implementations are depending on the programming language (i.e. classes and/or pointers) and are thus discussed in the next class. But it is good to describe how such a data structure can be used to describe real world problem in an abstract way.

A different use of a graph can be explored in a context the students already know very well: a labyrinth in the RoboCup. It can easily be seen by the students that an arbitrary labyrinth can be mapped to a graph: each crossing section corresponds to a vertex, each way from one crossing to another to an edge. See Figure 3.11 for an example. A huge difference between a problem defined in the map of Romania in comparison to a labyrinth seen by a robot during the RoboCup is also obvious: the map of Romania is given while

the robot when he starts his tasks of e.g. finding a certain spot in the labyrinth has no map available.

With this background knowledge we discuss how a robot can build up a map with the help of a graph by adding vertices and edges as it travels through a labyrinth like in Figure 3.11. Also difficulties that occur for a real robot are mentioned. Is has to deal e.g. with the orientation of the robot and how to detect whether a place has already been visited.

As usual the class closes with a voluntary assignment. This time the students are asked to think about how an algorithm looks like that builds up a graph from a real labyrinth. The students are also motivated to either take their own RoboCup robot or one of the LEGO robots available in the computer lab and program it to solve the given task. At the end the algorithm should print out the graph in a suitable way. This assignment is definitely more complex than the others but the next class will take place in two weeks.

3.5 Fourth and fifth class: Problem Solving by Searching in Trees

The next important topic is too extensive to fit into a single class. So this is spread among the next two classes.

During the previous lessons we discussed how real world problems could be fit into a formal description for the computer science. These descriptions can be stored or illustrated as graphs. We also showed that solving a problem then is equivalent to searching for a way in this graph from an arbitrary starting point to a designated target or goal state. A way in this context is simply an instruction from which vertex to go to which one next. How to find a way or the "best" way in a graph is discussed in this class.

Searching for solutions in graphs is a general approach that works for many given real world problems, also e.g. for CSP and SAT that have already been described in the previous section. Many of these problems are NP-complete or NP-hard. Thus current research includes different heuristic approaches to find possible solutions. A good overview can be found in [Edelkamp and Schroedl, 2011].

Without considering any constraints, basically, we start by taking our starting vertex and expand it by adding all edges that lead from it to any other vertex and the connected vertices themselves, too. Then, for each added vertex again we expand it by adding all edges and their vertices. This process continues until we reach the goal. The problem that occurs is, that the developing structure contains many irrelevant vertices as well. Also many vertices will be added twice or many times more. In the real world this would mean, that e.g. the robot reaches a state in which it has already been before. This is a queue of redundant or circular actions and has to be avoided. So the algorithm has also keep track of whether a certain vertex has already been visited or not and only add new vertices and the edges leading to it.

The resulting structure then is a subset of the original graph called a tree. At this point it is important to pin down some important differences between graphs and trees and hence introduce the data structure tree briefly, i.e. a graph without cycles.

After that we can start to develop searching as a mean to solve problems. Additionally we have to point out that classic searching can come in two flavours. The so called uninformed and the informed search. In the subtask of uninformed search the searcher has no outside or top view of the problem. No additional information like costs or distances are given. Just the vertices and edges are known. Informed search adds the possibility to get additional data about the edges of the graph and hence enables algorithms to optimize the results.

At this point to be able to talk about search algorithms we have to describe the data structures stack and queue briefly. It will be sufficient to – at least visually – describe the methods stacks ("push" and "pop") and queues ("add" and "remove") offer and discuss possible other applications than searching.

With the help of this knowledge we can now introduce some of the classic search algorithms: breadth first search (BFS) and depth first search (DFS) for algorithms of the uninformed search tasks and as a representative of informed search the $A^\star$ search algorithm.

Instead of explaining these classic search algorithms we develop a classroom learning station scenario. The students will split up in three groups. Each group works together at each of the three stations. At each station the groups find a description of an unknown algorithm plus an illustration of a graph. A working sheet is available for each individual student. The graph is the same for all of the three algorithms and can be seen in Figure

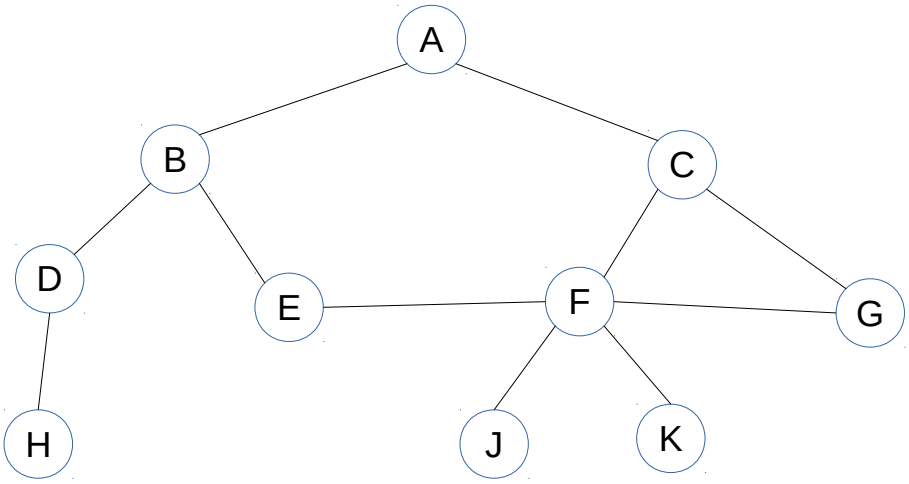


Figure 3.12: The graph that is used as the basis for all three search algorithms. Breadth-First-Search (BFS) and Depth-First-Search (DFS) can work directly with this graph. The weights of the edges that are needed to calculate the optimal solution with the $A^{\star}$ algorithm can e.g. simply be derived by measuring the distances between the vertices.

3.12. The task is to mimic a computer and exactly follow the instructions of the algorithm. The result and output of each algorithm is a specific tree. The three complete work sheets of the learning stations for the algorithms can found in Appendix A.1 starting on page 74.

The algorithms themselves are not written in any particular programming language. We use so called pseudo-code for the description of the algorithms. The example of the breadth first search can be seen in Listing 3.1.

```
1  Breadth-First-Search (G, s)
2  // G ... graph, s ... starting vertex (here: vertex A)
3
4  Create new, empty Queue Q
5  Mark s as visited
6  Q.add(s)
7  While Q is not empty {
8      k = Q.remove()
9      Print k
10     For each neighbours j of k in G do {
```

```
11          Mark j as visited
12          Q.add(j)
13      }
14  }
```

Listing 3.1: Example of the breadth first search algorithm written in pseudo code as it was used in one of the learning stations.

After enough time to work at each of the learning stations, we briefly discuss the results and possible problems together. Especially the $A^\star$ algorithm might pose some difficulties to the students. Each of the algorithms will be discussed with another example of a graph or a tree. It is important also to highlight the advantages and weaknesses of each of the algorithms: Under which conditions can a specific algorithm be used? When to switch to a different one? Are there differences in memory used? What about computing time? Which of the solutions are optimal results? Which algorithm might be used in the RoboCup to find a way through the labyrinth once we would have a map of it? It is important to note that searching in graphs can give good results but one does not have a guarantee to find optimal solutions or even any solution at all.

We plan to end the fourth class after a short discussion about the results of the learning stations. The summary and comparison of the three algorithms is a good starting point for the fifth class. Thereby we also have a short summary of the last lesson.

Since searching for an (optimal) solution is a central element in AI we elaborate a bit more on this topic. We plan to have some time available at the beginning of the fifth class for a more extensive example using the $A^\star$ algorithm for optimal route planning. The rest of the class is used for a practical exercise. The students have already been introduced to object oriented programming in the language C♯. For this reason we use a free IDE called "♯develop" ("SharpDevelop")[2] and available implementations of the graph, stack and queue data structures.

To make connections with the surroundings and current interests of the students we choose a task that is of importance to them. The assignment for the rest of the fifth lessons that can also be completed until the next time is:

- Visualize your training labyrinth from the RoboCup rescue league:

[2]The homepage of this IDE and possibilities to download can be found at `http://www.icsharpcode.net/opensource/sd/`

  - Draw the labyrinth as graph on paper
  - Use one vertex per square area
  - Draw an edge when a way between two adjacent areas exists
- Implement a search algorithm in ♯Develop:
  - Form teams of 2 or 3 students each
  - Create an empty instance of a graph
  - Statically add vertices and edges of your graph
  - Implement one search algorithm of your choice
  - Printout and illustrate the resulting tree

3.6 Sixth class: Classic Planning

The main topic of the second to last class is "classic planning". A good starting point is to recall what we have learnt so far to solve problems in computer science: we can formulate a problem with states and transitions and visualize and store it as a graph. Then we define a starting state and one or more goal states. Finally we can start to search – with a reasonable search strategy – for a way from the starting state to one of the goals. The result is a tree that shows a plan how to solve the given problem.

Searching to solve a problem is a reasonable strategy when one already is in possession of a graph of the given problem. In many real world situations only coarse descriptions of states and their properties are available. Additionally one knows possible effects of actions that can also be different in various situations. The question dominating this class will be: how to build a problem solver under these circumstances?

Planning is a very vast topic that usually consumes a whole lecture of its own. It forms a basis that can be used in various applications like robotics, agents, testing etc. A good introduction to planning is [Nau and Ghallab, 2004]. A classic planning algorithm is STRIPS (Stanford Research Institute Problem Solver) by Richard Fikes and Nils Nilsson [Fikes and Nilsson, 1971].

General, a problem is formulated different than before:

- Where (in which state) do we start?
- What can we do, i.e. what actions are available in each state?
- What happens when we apply an action? What is the result of each action?
- Are we done yet? Do we already have reached the goal?

In these scenarios states are defined or described with logical statements. This can either be propositional logic in the simplest case or e.g. first-order logic for more advanced scenarios. A simple example for a state description is

$$TrafficLightRed \wedge CarStopped \wedge PedestrianCrossing$$

where each of the propositions can either be true or false. And hence a state can be defined as a concatenation of different propositions with some of the classic logical connectives.

At this point it will be important to recapitulate logic briefly. Since the students have some experiences in programming they will already know a little bit about logic from premises in if-statements. But we have to be sure that at least the standard logic operators (NOT, AND, OR) are known and e.g. truth tables can be fill out correctly.

First-order logic (FOL) will definitely be a new topic for the students. We do not have to introduce it extensively but at least to a degree where it will be understandable how first-order logic can be used to describe the world more accurate and simpler than with propositional logic. In this case it suffices to explain how objects themselves (constants), functions, properties of and relations between real world objects can be defined with FOL. FOL is very important to AI because it forms the basis of most approaches to knowledge representation and reasoning.

A simple but good example can be seen in Figure 3.13. After defining objects like “John”, “Richard” and “Kingscrown”, some properties of the world can be modelled with them, like:

$$Crown(Kingscrown), IsHappy(Richard), HasMoney(John),$$
$$OnTheHead(John, Kingscrown), Brother(Richard, John), \ldots$$

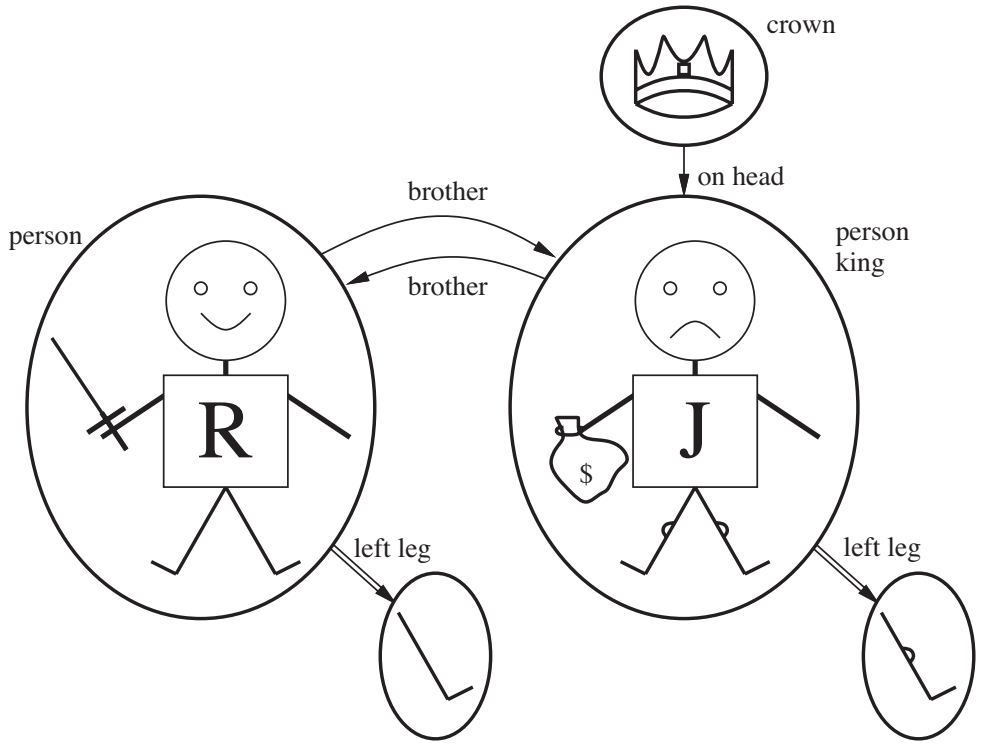


Figure 3.13: A nice example from [Russel and Norvig, 2012] to describe how properties and relations between real world objects – here Richard, John and a crown – can be defined in first-order logic.

We can then form more complex statements about the world to describe possible states, like e.g.

$$King(John) \wedge \neg King(Richard) \wedge HasMoney(John) \wedge HasSword(Richard)$$

Additionally, quantifiers could be at least mentioned briefly and how they enable us to make statements for all of the objects that fulfil certain criteria with the universal quantifier (e.g. all things that bark: $\forall x Bark(x)$) or for at least one object with the existential quantifier (e.g. there exists at least one thing that bites: $\exists x Bite(x)$).

Each of these properties and statements can again either be true or false. Now, states including starting and goal states can be defined like in the following example from [Russel

and Norvig, 2012]. The students should now be able to infer what is being described with the next two statements:

$$Init(Cargo(C_1) \land Cargo(C_2) \land Plane(P_1) \land Plane(P_2) \land \\ Airport(JFK) \land Airport(SFO) \land \\ At(C_1, SFO) \land At(C_2, JFK) \land At(P_1, SFO) \land At(P_2, JFK))$$

$$Goal(At(C_1, JFK) \land At(C_2, SFO))$$

We can now introduce actions based on so-called pre- and post-conditions. Each action has a unique name and a description ("pre-condition") when it can be taken. Also a possible result or effect ("post-condition") has to be given for each action, like:

$$Action(EatCake, \\ Precond : CakeExists \\ Effect : \neg CakeExists$$

$$Action(Fly(p, from, to), \\ Precond : Plane(p) \land Airport(from) \land Airport(to) \land At(p, from) \\ Effect : \neg At(p, from) \land At(p, to)$$

Finally, we can now describe the classic planning approach in computer science. We restrict ourselves to the so-called state-space planning, where

- we search among all possible state combinations
- states are connected via actions
- we have to find a state in which all goal conditions are fulfilled
- the solution is the chain of action from starting to goal state

This can best be described with an example like the famous "block world" from the Sussman anomaly as seen in Figure 3.14. This can be done just as a visual model of a problem definition and a basis for a discussion or, if time allows it this can also be used to describe the Sussman anomaly in more detail, i.e. why planning is non-trivial and how

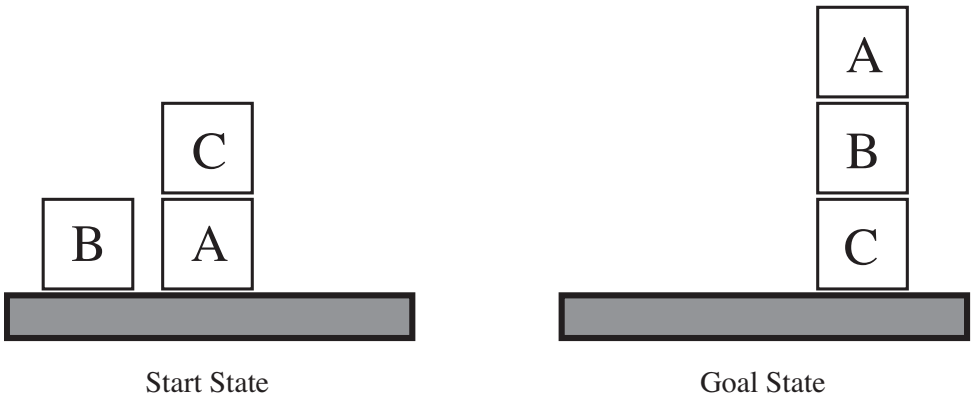


Figure 3.14: A simple problem that can be used to think about strategies how to get from the starting state to a goal state.

naive algorithms can fail here. The students should now try to find out how planning can be done under these circumstances. This will best be done in a group work in which the students have to solve the problem puzzle themselves first, then reflect their doings and answer the following questions afterwards:

- Which strategy did you choose?
- How did you choose the sequence of actions?
- What has to be remembered in each step?
- Can you pack your strategy of searching for a sequence of actions in an algorithm?

One of the examples to be used in this group work is the classic planning problem of the monkey that desperately wants to have some bananas that are suspending from the ceiling of a room and that he just can't reach. It can be described like in Listing 3.2. Here the monkey has to set up a simple plan, like move the box beneath the suspending bananas, climb the box and finally grab the bananas. This and the second simple classic planning problem to be used can be found in Appendix A.2.

```
1 Initial state: At(A) ∧ Level(low) ∧ BoxAt(C) ∧ BananasAt(B)
2 Goal state:    Have(Bananas)
3
4 _Move(X, Y)_
5 Precond: At(X) ∧ Level(low)
6 Effect:  ¬At(X) ∧ At(Y)
7
```

```
_ClimbUp(Location)_
Precond: At(Location) ∧ BoxAt(Location) ∧ Level(low)
Effect:  Level(high) ∧ ¬Level(low)

_ClimbDown(Location)_
Precond: At(Location) ∧ BoxAt(Location) ∧ Level(high)
Effect:  Level(low) ∧ ¬Level(high)

_MoveBox(X, Y)_
Precond: At(X) ∧ BoxAt(X) ∧ Level(low)
Effect:  BoxAt(Y) ∧ ¬BoxAt(X) ∧ At(Y) ∧ ¬At(X)

_TakeBananas(Location)_
Precond: At(Location) ∧ BananasAt(Location) ∧ Level(high)
Effect:  Have(bananas)
```

Listing 3.2: The formal description of the classic bananas problem where a monkey has to come up with a simple plan to grab some bananas suspending from the ceiling.

In this form the problem will easily be solved by any student just by reading the names of the possible actions “Move”, “MoveBox”, “ClimbUp”, “ClimbDown”, “TakeBananas”. So a possible strategy will probably not be constructed only by using the actions but by using the context of the actions names and hence, using the experience of the students themselves. To help the students to think “algorithmically” and not to use their own human planning strategies we harden the planning problem by stripping it from its real-life context and thereby removing the semantic connection. This can achieved be e.g. just renaming all actions and objects. Compare the context-free problem in Listing 3.3 with the original in Listing 3.2.

```
Initial state: Ta(A) ∧ Veell(Oool) ∧ Xotab(C) ∧ Naassabta(B)
Goal state:    Eahv(Nnaaabs)

Action ( Brrrk(X, Y)
Precond: Ta(X) ∧ Veell(Oool)
Effect:  ¬Ta(X) ∧ Ta(Y) )

Action ( Vrrzch(Bzzzktkt)
```

```
Precond: Ta(Bzzzktkt) ∧ Xotab(Bzzzktkt) ∧ Veell(Oool)
Effect:  Veell(HHgi) ∧ ¬Veell(Oool) )

Action ( Knoimff(Bzzzktkt)
Precond: Ta(Bzzzktkt) ∧ Xotab(Bzzzktkt) ∧ Veell(HHgi)
Effect:  Veell(Oool) ∧ ¬Veell(HHgi) )

Action ( Arrrgh(X, Y)
Precond: Ta(X) ∧ Xotab(X) ∧ Veell(Oool)
Effect:  Xotab(Y) ∧ ¬Xotab(X) ∧ Ta(Y) ∧ ¬Ta(X) )

Action ( Ckckch(Bzzzktkt)
Precond: Ta(Bzzzktkt) ∧ Naassabta(Bzzzktkt) ∧ Veell(HHgi)
Effect:  Eahv(Nnaaabs) )
```

Listing 3.3: The formal description of the classic bananas problem where a monkey has to come up with a simple plan to grab some bananas suspending from the ceiling.

The idea is to force students to think algorithmically. How do I get from the start to the goal? Which action should I choose first? Next? Where do I start? Start from the beginning and try some actions to see where I can get? Or do I look a the goal and try to see how I got there? We tell the students to explicitly reflect about their strategies they might evolve and write them down.

If the students come up with an algorithm or at least are able to describe their strategy, we can analyse and summarize it. Probably it will be similar to one of the two classic problem solving approaches: forward and backward search. Anyway, we will describe the two algorithms briefly in pseudo code like in Listings 3.4 and 3.5.

```
ForwardSearch ( D, I, G )
  s = I          // we start in the initial state
  A = empty      // and haven't planned anything yet
  loop
    if ( s fulfils G ) return A    // goals already reached
    usable = List (a in D where precond(a) in s is fulfilled)
    if ( usable == empty ) return ERROR   // no action found!
    randomly choose an action a from usable
```

```
9     s = execute (s, a)  // execute action in current state
10    A = A + a           // concatenate action to our plan
```

Listing 3.4: The forward search algorithm for problem solving in pseudo code where D is the set of all actions, I is the initial state and G is the goal state.

It is important to discuss the differences between the two approaches. While forward search is the simplest and most intuitive strategy, one of its difficulties is the random choice of an action. This can e.g. lead to infinite loops of actions. One will find a solution randomly indeed. Since one has to consider all suitable actions in each state this can lead to a huge graph in the end and hence, consume much memory. The optimal solution could be found by doing a breadth first search.

```
1 BackwardSearch ( D, I, G )
2   A = empty       // nothing planned yet
3   Loop
4     if ( s fulfils G ) return A
5     relevant = List (a in D that are relevant for G)
6     if ( relevant == empty ) return ERROR
7     randomly choose an action a from relevant
8     A = A + a     // concatenate action to our plan
9     G = execute_backwards (G, a)
```

Listing 3.5: The forward search algorithm for problem solving in pseudo code where D is the set of all actions, I is the initial state and G is the goal state.

Backward search is more goal oriented. It starts at the goal state and tries to find out which actions could have led to the goal. Only these relevant actions are considered in each recursion. Hence, less memory is needed in this algorithm compared with the forward search. The crucial step here is find out the relevant actions that led to the goal and to execute an action backwards. This information will not be available for all problems.

An assignment at the end of the class could come again from the RoboCup league. The students can take a league of their choice and try to formulate the underlying problem in propositional or first-order logic. Is especially one league or which leagues are better suited for this approach? Write down difficulties that occur during formulation! What exactly does it make difficult? The answers will be discussed at the beginning of the next class.

3.7 Seventh class: Machine Learning and Outlook

This last class is used for several topics. First, together we briefly repeat what we have learnt so far. The topics included

- Deterministic finite automata (finite state machines)
- Reactive controllers (reflex controllers, Braitenberg vehicles ...)
- Model-based reflex agents
- Goal oriented agents
- Utility based agents
- Basic data structures like graph, tree, stack and queue
- Problem solving with searching
- Planning of a sequence of actions

Looking back, we compare some of these approaches. Which ones are easy, which ones are hard to understand. Which parts of these ideas can be directly used by the students in their daily life? Coming back to the constructionistic approach, which can be incorporated in their struggles in the RoboCup leagues? What algorithms can be used? Are all leagues suited to incorporate AI algorithms? Why or why not?

Concluding we can say, that we have mostly talked about readily programmed robots so far. These don't generate new knowledge in terms of behaviour. They use absolutely no knowledge that the programmer hasn't thought about himself or herself. All algorithms and strategies are programmed into the robots memory. In that sense, also e.g. searching for solutions in generated trees only adapts a pre-programmed behaviour to a possibly new surrounding.

Referring back to Figure 3.7 on page 25 we explain the differences to general learning agents. Based on these differences we formulate some of the principles of machine learning like e.g.

- there is no static knowledge base, only some previous knowledge might be available
- there have to be mechanisms to expand the knowledge base, called learning strategies

- there has to be some sort of critic and feedback coming from within or from outside the agent – supervised versus unsupervised learning

The idea for the rest of the class is to show and briefly explain some of the vast examples and different approaches that can be found in the literature. We select suitable machine learning algorithms based on the following two questions. Which can be most useful in robotics or especially in the RoboCup leagues? And which represent very unique approaches in their basic idea about how learning can be done. Hence, we focus in this class on the presentation of "Reinforcement Learning" (RL), "Logic based inference" (classic AI), "Decision trees" and the connectionistic approach of "Neural Networks".

Reinforcement Learning According to [Sutton and Barto, 1998] reinforcement learning uses the idea of rewards that an agent receives at the end of a possibly long sequence of actions. In this case the agent does not get any information about which of its individual actions was good or bad. It merely receives information about success or failure of the total actions sequence.

This can be illustrated e.g. with chess playing where at the end of the game it is a defeat or a victory (or a remis). But the result itself can in most cases not be pinned down to a single action. A similar example is that of a mouse that has to find a piece a cheese in a labyrinth. The mouse receives a positive feedback by finding the cheese. When it does so it will probably not be aware of which of its actions was the best one nor does it receive a tiny bit of cheese at each corner. But it will have to remember the sequence of actions somehow. Reinforcement learning deals with the problem of distributing a positive (or negative) reward amongst a sequence of actions according to various algorithms. We also briefly mention the problem of exploration of new knowledge versus exploitation of existing knowledge in this context.

In robotics or especially in RoboCup this could be used e.g. with a sort of module based reinforcement learning like in [Kalmar et al., 1998] where sequences of readily programmed basic actions are being learnt. Several short scientific videos can help to visualize the learning process of robots. Suitable videos come from the domain of RoboCup and show how a robot learns e.g. to follow a line[3], a forward movement[4] or to shoot a penalty [Hester et al., 2010][5].

[3] `https://www.youtube.com/watch?v=jaTOSLd56hI`
[4] `https://www.youtube.com/watch?v=2iNrJx6IDEo`
[5] `https://www.youtube.com/watch?v=mRpX9DFCdwI`

Logic based inference The idea of logic based inference will have to be kept briefly since the students lack a deeper understanding of logic at this grade in school. We have introduced propositional logic and first-order logic during the sixth class on June 26^{th}. So we can at least show with simple example how basic mechanisms like deduction or inference work with logic. Starting with an example like

$$\forall x Dog(x) \wedge Bark(x) \Rightarrow \neg Bite(x)$$

we can briefly show operations like the "Modus Ponens", "Modus Tollens", the $\wedge$-introduction or the $\wedge$-elimination when certain facts are given. These simple inference rules give examples of inference and reasoning in general and hence demonstrate, how new facts can be inferred in such knowledge databases. A more detailed description of inference and reasoning can be found in [Brachman and Levesque, 2004].

Decision Trees Decision trees are a first good example of how we can implement learning based on a set of examples. Because the trees resulting from the learning procedure can also be viewed as if-then-rules that the students already know. See Figure 3.15 for an example of a decision tree. It would be too detailed to discuss topics of learning theory like overfitting, training and test sets, cross validation etc. A good software tool for experimenting with decision trees and other machine learning algorithms is WEKA [Hall et al., 2009].

We can let the students experience the difficulty of generating rules from data by giving them a short table of data and ask them to create correct and possibly short rules that reflect the data correctly by hand. It is then easy to imagine for the students that creating rules becomes even more difficult when the number of attributes and/or the number of examples increases.

It suffices in this class to show an example of some complex data and a resulting decision tree afterwards to give a picture of what decision tree algorithms are capable of.

Neural Networks As a representative for connectionistic learning methods neural networks conclude this presentation of learning agents. Some facts about the human brain like 100 billion (10^{11}) neurons, the average of 10000 connections between each of these neurons, the resulting 10^{15} synapses and its about 4 km of neural "cabling" per cubic

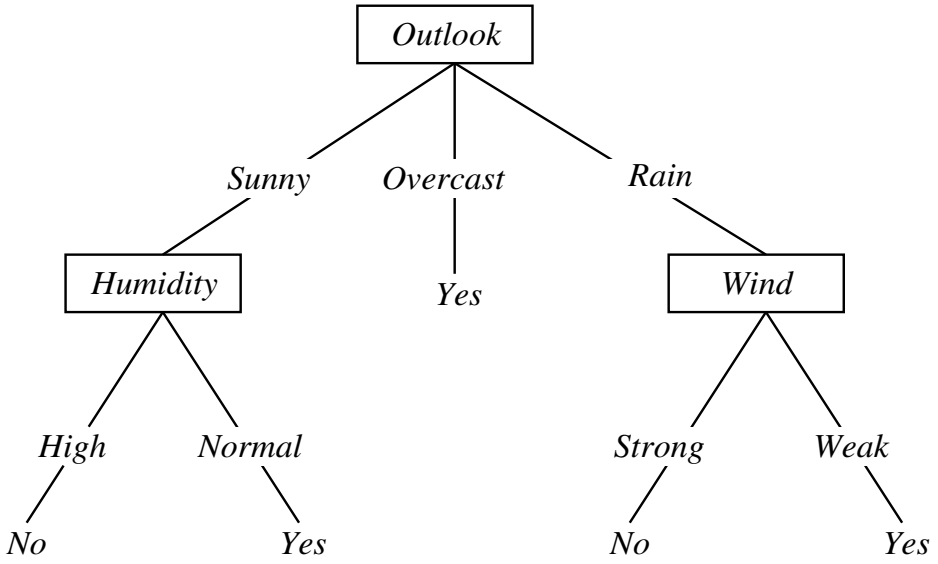


Figure 3.15: A example of a simple decision tree from [Mitchell, 1997] to decide whether one should play tennis ("Yes") or not ("No"). Each leaf represents a decision. Decisions are depending on certain attributes like "Outlook", "Humidity" and "Wind". A complete rule can be read starting at the root of the tree down to a specific leaf.

millimeter of brain can motivate the question about how does this thing work [Herculano-Houzel, 2009].

At least the basic idea of how the mechanisms of a biological neuron are mimicked by a mathematical function can be shown. Referring to the perceptron, the simplest neural model, the strength of a synapse receiving input from the i^{th} neuron is represented by a more or less static weight factor w_i, the amount of input to a neuron corresponds to a variable x_i, the integration of post synaptic potentials in the soma can be modelled by the summation of the weighted inputs and the role of the axon hilloc to generate the output o_j of the j^{th} neuron is reduced to a non-linear limiting function φ. In the end each biological neuron is modelled by an equation like

$$o_j = \varphi\left(\sum_{i=1}^{n} w_i \cdot x_i + w_0\right)$$

where w_0 represents a resting potential or bias. There are several limiting functions in use like the Heaviside function (or step function) or the differentiable sigmoid function $sig(t) := \frac{1}{1+e^{-t}}$.

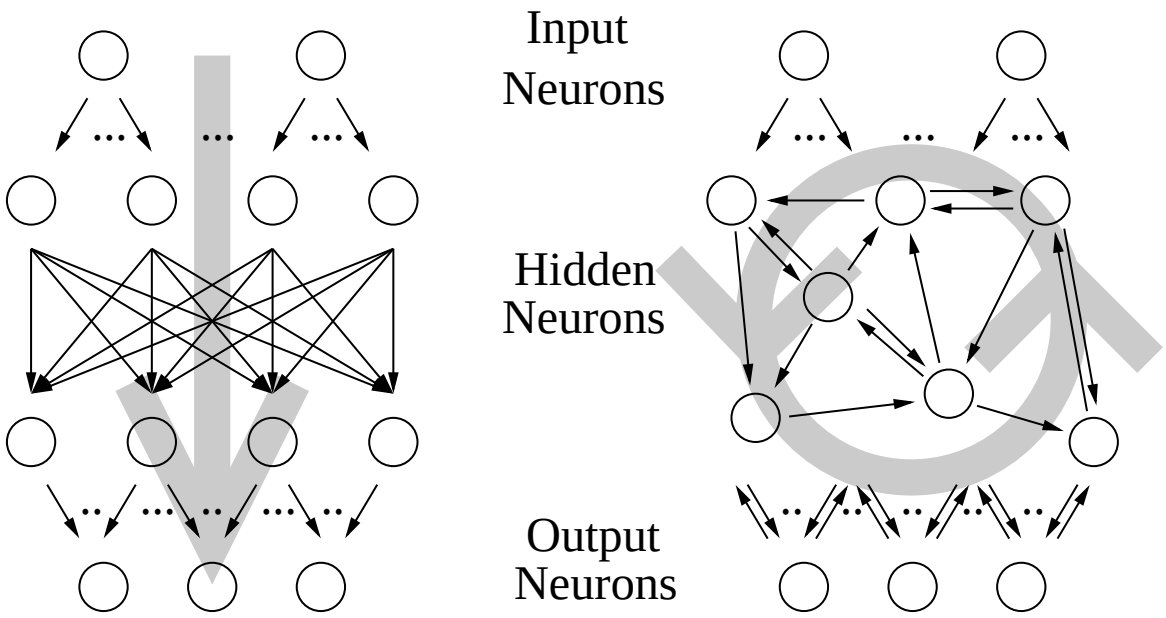


Figure 3.16: Two simple drawings from [Burgsteiner, 2006] of neural network to illustrate the structure of artificial neural networks. A pure feed-forward network is shown on the left hand side where connections strictly emerge from the input in the direction towards the output layer. The recurrent network on the right hand side allows arbitrary connections within the network.

Based on Figure 3.16 the problem of learning with neural networks can be shown. Here, learning is the adaptation of all the weights within the neural network until the output of the network exhibits the same behaviour for each input as e.g. a given table of examples demands. Again, concrete learning algorithms like back propagation would be too detailed for this class. The basic ideas of such algorithms like minimizing an error function suffices in our opinion.

The rest of the class must be used for the evaluation of the complete course. We distribute questionnaires and ask the students to fill them out honestly. This part of the evaluation is completely anonymous. It is important to stress out that not the students are being evaluated but the course itself and the teachers. Furthermore we need the feedback to develop the course for possible future implementations. After collecting the filled out questionnaires we can thank the students for their participation and collaboration during the whole course and wish them pleasant holidays with the final remark that they can contact us any time if they have further questions.

Chapter 4

Realization and Evaluation

4.1 General restrictions and participating school

The general subject area of the planned course as it was agreed with the participating school BRG Kepler [BRG Kepler, 2015] was "Artificial Intelligence". The teachers and students also wanted to have a strong connection with a voluntary class "RoboCup"[The RoboCup Federation, 2016] they had during the last years. All of the students were participants in one or more RoboCup world cup championships. So they have experience in programming a robot for one of the RoboCup leagues. Furthermore some students also wanted some connections with the individual topics chosen for their final year thesis. Especially the last wish is hard to fulfil for all of the students, because the topics had a very broad range, e.g. "Development of software to solve the game Tic-Tac-Toe with a humanoid robot", "Analysis of the possibilities to construct and read QR-codes" or "Localisation of a noise source with the robot Nao"[Aldebaran Robotics, 2016].

The computer science lessons in Austrian general high-schools don't include topics like artificial intelligence, nor are software engineering or even algorithms contained [Bundesministerium für Bildung und Frauen, 2016]. Hence, our course could currently only take place as a voluntary class. Hopefully, this will change in the future. But this also provided the opportunity (and challenge) that students of different age and knowledge could attend our course. We had students from the 5^{th} to the 7^{th} grade (i.e. 9^{th} to 11^{th} year of school attendance) subscribed. But these students do already have a bit of knowledge of object oriented programming in the language C♯.

One drawback is that the assignments during the classes can not be made mandatory. This class is voluntary and no grades are given at the end. So on the one hand, there is no need to collect gradable work done by students. On the other hand, in our approach of teaching we don't like to force the students to do something. We want to motivate the students to do these assignments voluntarily. It has to be considered that at this time of the school year the students have also a lot of studying to do for all the other school subjects. Hence, additionally we should limit the amount of work to do until next classes and don't expect that all of them do the assignment. It should be considered that these appointments can also be just pointers to interesting topics and further readings – instructions for possible later projects on their own.

With our school partner BRG Kepler [BRG Kepler, 2015] we agreed on timeslots where the students (coming from different classes) would all have time. To be able to teach our seven planned units, we used time slots on friday afternoons from 13:20 to 15:20 from May to July 2015. This was a very limiting factor since we now only had $7 \cdot 2 = 14$ hours for an introduction to artificial intelligence. The public holidays created pauses of 2 weeks between some of the lectures. These could be used for more extensive experiments, research assignments or software development. The next step was to create a golden thread through all of the topics outlined above and to portion them accordingly. Furthermore it was necessary to create appealing and interesting exercises and project work to fill the time between the lessons.

4.2 Qualitative monitoring during the classes

This chapter briefly reviews the designs of our seven units in comparison with the actual realisations. This is the next step in our course development cycle of planning – doing – studying – acting. We took some pictures and field notes during or directly after the individual units. These serve as the foundations for the next paragraphs.

Unit 1: Introduction and automata The first class was scheduled for May 8th 2015. We started with a short address of welcome and introduced ourselves, since we were new to the students. They never had classes with one of us before. The recent victories of some of the students during the Austrian Open of the RoboCup 2015 (where they qualified for

the world cup finals) gave an opportunity to introduce themselves to us and also let them present the robot they constructed for the RoboCup.

The unit worked out liked it had been planned before. The students started to join in from the very beginning. We would describe it as a relaxed atmosphere. Although some students seemed a bit more participating than some others all of them joined the discussions and attended the class very concentrated.

We expected that a formal representation of a FSM would be difficult for the students to understand. To our surprise, the concept of finite state machines was obviously not hard to understand at all. Explaining the foundations with some examples and showing the principles on the basis of a robot controller helped to comprehend the concept. The students were able to construct their own FSMs during this first unit.

With the presentations of the results on the whiteboard we gave the students the opportunity to discuss their ideas with the plenum. See Figure 4.1 for an example of one such presentation. Partially they did quote humoristic performances which reflected the relaxed atmosphere. Thereby we received immediate feedback about the first unit too. By watching the presentations and seeing how the students explained their solutions we concluded that the students achieved the competency of constructing a simple FSM. The first unit ended with the first homework assignment:

- Is it possible to illustrate your own robot from the RoboCup as a FSM?
- Reflect in your existing RoboCup teams what you were thinking during the programming of your robot!
  - Can you characterise the states? Can you find names for them?
  - Which events are possible?
  - Due to which events do the states change?
  - Which state transitions are possible?
- Design a poster or a PDF with your FSM until the next class!

Figure 4.1: Two students present the result of their group work during the first class. They constructed a controller for a toaster that is able to toast the bread in two different crisp settings including a panic button that leads to an explosion.

Unit 2: Intelligent agents The presentation of homework at the beginning of this second unit on May 22nd 2015 showed some problems. Of all students only one had answered all questions and designed a complete FSM. The others had some fragments of their controllers. What we could see indicated that the previous lesson was successful. Some said it was impossible to determine states out of their unstructured code. Some simply had tests in school and needed time to learn.

During the unit we wanted to get the students to program with a real robot. But the planned brief overview about different basic agent models (i.e. reflex agent, model-based reflex agent, goal oriented agent, utility based agent and learning agent) turned into a lengthy discussion. We wanted to just give an overview which type of agent has which advantages and disadvantages and use it as a possibility to point to coming units in which these models would be discussed in more details. But the interest of the students turned out to be higher than expected. We didn't want to let some questions be

unanswered and so we decided to leave out some topics for the next unit (model based, utility based and learning agent) to have some more time for practical work during the unit.

The second half of the unit was used to implement some different Braitenberg controllers on Lego Mindstorms NXT robots that were available in the schools robot lab. Working groups of 2 students were formed to implement a Braitenberg controller of their choice. Some of the teams were not able to finish their software till the end of the class. So we left out the intended assignment of developing a model-based agent and defined the completion of the Braitenberg vehicle as the assignment for this unit.

Unit 3: Problems and graphs The third unit on May 29th 2015 started with the presentation of the self-programmed Braitenberg vehicles. According to the students the task wasn't hard but only some of them did find enough time to complete this assignment. The agents showed the expected behaviour. Watching robots turning away from sound or seeking a light turned out to introduce some fun in the classroom. It is definitely worth letting students implement the simple but powerful Braitenberg vehicles to show how tiny modifications dramatically can change the behaviour of a robot. But from this time on we decided to change the assignments to just "pointers to additional voluntary work that can be done sometime in the future."

Although the rest of the unit was planned to be quite theoretic, the students showed high interest in it and participated very actively during the unit. This might be due to the possible relation of graphs and the RoboCup junior. Here for the first time the students saw practical alternative solutions to the labyrinth task from the RoboCup (the rescue league) and how a certain data structure and some algorithms might help them achieving better results in the future. Especially the possibility to build a map of a labyrinth in form of a graph was anticipated very well.

The optional assignment in this unit was thus:

- Form teams
- Use the RoboCup labyrinths available in the robotics lab
- Each square represents a vertex
- The sensors give hints which directions are available
- This gives possible paths and edges

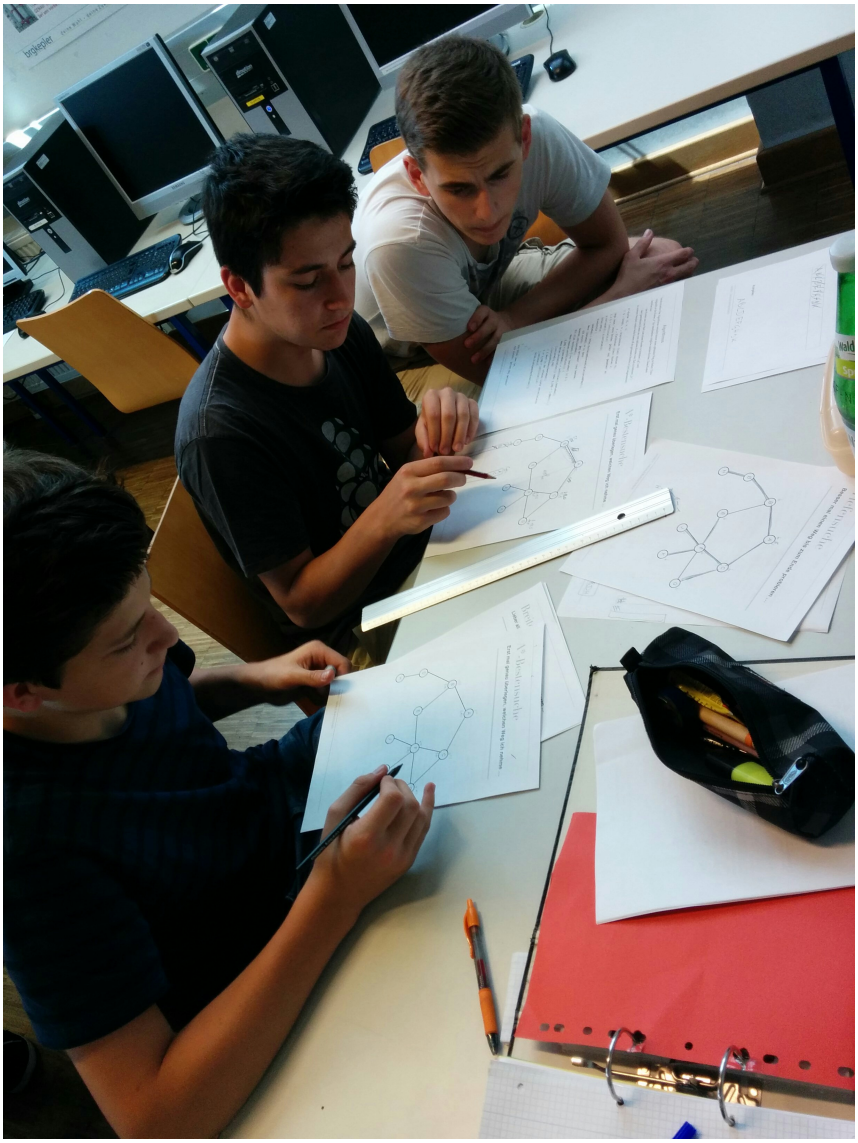


Figure 4.2: Students work together in a group of three to research the $A^\star$ search algorithm. They should understand the algorithm by analysing the source code on the basis of a given graph.

- Store these information and optionally visualise it
- Bonus question: how can one be sure to have explored all possible paths?

Unit 4: Problem solving by searching in trees This unit took place on June 12$^{\text{th}}$ 2015. We didn't expect that a student had really solved the optional assignment from the last unit. Nevertheless did one student indeed program her robot to build a map

from the RoboCup labyrinth and store the information in a self designed array construct. At the end she printed out an overview of connections within the labyrinth in a matrix.

We spent much time in recapitulating and discussing how a problem can be mapped to a graph. But this essential foundation made it a lot easier then to build up an understanding how searching in a tree can lead to a solution. And how and why the solution is a data structure called Tree.

The short input about what a Stack and a Queue is sufficed to initiate successful classroom stations. At each of the three stations one of the three groups had to find out how a certain algorithm works by themselves. At each station they had about 15 minutes time. See Figure 4.2 for an impression. Depth first search and breadth first search showed to be no problem. Only the $A^\star$ algorithm was a bit harder to analyse and to understand. Students and teacher discussed all of the algorithms together afterwards. We did not offer an explicit assignment at the end of the unit, instead we suggested to work an any unsolved assignments of interest from the previous units. This was also because we planned to do a more practical session in the next unit.

Unit 5: Problem solving by searching in trees – continued The unit on June 19$^{\text{th}}$ 2015 started with a repetition of the $A^\star$ algorithm because this one turned out to be really hard to understand. We discussed and calculated the example with the Romanian street map from [Russel and Norvig, 2012].

The rest of the unit consisted of an in class programming assignment which could also be completed until the next class if the time during the unit was not sufficient:

- Draw (just on a piece of paper!) the RoboCup labyrinth as a graph
  - One vertex per squared field
  - Add an edge when there is path to an adjacent square
- With your SharpDevelop IDE (in teams of two or three students each):
  - Generate a Graph object
  - Just statically add vertices and edges
  - Program one of the search introduced search algorithms

– Output the resulting tree on the screen (in a suitable way)

This task turned out to be no problem for the students. The students helped each other when questions about the labyrinth representation arose. The teacher had to help sometimes with syntax problems in the programming language. But in the end each group was able print out (textual) information about the labyrinth on the screen. None of the groups chose the $A^\star$ algorithm, only breadth first and depth first algorithms were implemented.

Unit 6: Classic planning The second to last class took place on June 26$^{\text{th}}$ 2015. Planning this sixth unit was difficult. The students should learn about very new and very formal content – propositional and first order logic. Additionally they should immediately be able to apply their knowledge to given problems. We restricted ourselves to a basic introduction to logic, just to be able to discuss planning algorithms in general and let the students get a feeling about the difficulties and strategies.

The exercise turned out to be quite exciting to the students. We wanted to completely blank out any knowledge of the context and thereby focus the attention on the algorithmic problem solving strategies of the students (see Appendix A.2 on page 81 for the original scrambled assignments). It was felt by them more like a puzzle or a game than an exercise. See Section 3.6 for a more detailed description of the exercises. The exercise during the unit was:

- Think about how to solve the given problem (in groups)!
- Reflect on what strategy you follow!
- According to what criteria do you choose which of the given action?
- What has to be remembered or noted from action to action?
- Try to formulate your strategy in an algorithm!

The algorithms that the students developed for themselves formed the basis for the comparison of the classic forward and backward search algorithms. The students formulated algorithms that had a similar strategy as the forward algorithm. With that kind of exercise the formal content was anticipated quite well.

Unit 7: Machine learning and outlook The beginning of the last unit on July 3rd 2015 served as a possibility to reflect a bit about what we had discussed during the course. Some open questions could be answered.

The second part was dedicated to a brief overview of some machine learning algorithms and give some ideas what AI is capable of today. This was mainly done by just giving the basic ideas of it and discuss differences to other approaches. Additionally examples with YouTube in which mainly robots tried to learn something were shown. Some of the videos had a direct connections with the RoboCup. These examples gained much interest from the students and immediately discussions started which approach could be incorporated in the next years robots. So the students definitely understood the basic idea of learning in robotics.

We then explained the reasons and the intentions of the upcoming evaluation and emphasized the anonymity. The last 15 minutes the students had time to fill out the questionnaire and then we finished the course by thanking them for the cooperation and wishing them pleasant holidays.

4.2.1 Homework assignments

The assignments that were meant to offer a platform for self-directed learning showed some problems that had not been considered by us so far. The end of the semester is coming nearer and the students have lots of tests to learn for. So a time consuming assignment in a voluntary class does not have the highest priority when it comes to time dispersion.

The number of students who regularly did the assignments became less from unit to unit. Only one attending student did nearly all of the assignments including the very lengthy one from unit three. This might be due to the fact that the assignment suited her final paper topic very well and also the RoboCup rescue league was her favourite league. She programmed an impressive robot controller that really steered a robot to build a map from an unknown labyrinth. All nodes and vertices were stored in a self designed data format.

Additionally, knowing a group of students for some time help a lot when planning tasks for a unit or for an assignment. The regular teachers tried to inform us about the strengths and weaknesses of the students. But too many details about the previous knowledge

came to light only during the units. So some tasks that were described by the teachers as being very easy for the students turned out to be very time consuming. For example programming a Mindstorms NXT robot was something that the students did the last time years ago. It took some time to get into it again. Previous knowledge in mathematics or programming turned out to be immediately available. These are competencies the students still need in school nearly every week.

4.3 Results of the empirical evaluations

The empirical project evaluation was done using reliable qualitative and quantitative empirical research methods [Diekmann, 2007]. In terms of quantitative evaluation we applied a paper-and-pencil post-questionnaire (Likert-scale, open-ended questions) comprising a self-assessment of acquired skills as well as feedback on the structure and teaching method of the weekly teaching units. Further qualitative data was collected by using techniques of participant observation (field notes, discussions, taking pictures) during the weekly teaching units [Jorgensen, 1989]. All collected data were treated confidentially and anonymously.

4.3.1 Self and foreign evaluation

Since this course is not a part of the regular or compulsory curriculum for computer science education in high schools in Austria, it does not make sense to stick with the current widely used notion of grades[6]. Due to the voluntary participation of the students in this course we are assuming a hight self-motivation anyway. A possible pressure coming from grading is rather counter productive. To be able to somehow evaluate a learning success, we apply current methods of learning goal assessments, especially self evaluation and foreign evaluation.

According to [Mabe and West, 1982] a self evaluation can be accurate and consistent with classic grading mechanisms when several conditions are met. These conditions include e.g. whether one assesses an ability or a past or future performance, the confidentiality and the expectations of the self-evaluations. The self evaluation can be done with a matrix where one can e.g. vertically list all competencies of the course and horizontally a scale

[6] In Austria common grades range from "Very Good", "Good", "Satisfying", "Sufficient" to "Not Sufficient" or have the numerical values from 1 to 5 respectively

with “I feel confident when doing this”, “I feel a bit unconfident” to “I definitely need help!”. Within this matrix every student has to rate himself or herself how good one thinks he or she has succeeded in achieving every competency. Such competence matrices can be found in numerous fields of studies. An example of such a matrix from the subject of mathematics can e.g. be found in [Prediger, 2008].

For our course we finally defined 17 core learning goals or competencies (see Table 4.1). The contents of the course were defined to enable the students to achieve each and every of those competencies. After the course the students were asked to fill out the self-evaluation anonymously. For an additional learning effect the self-evaluation could also be filled out with the names of students to be able to discuss the differences between self and foreign perception of the students. But this was not intended in our course. Table 4.1 shows the matrix that the students used to evaluate themselves. Note that the competencies matrix was of course written in German for the use during the course. The original evaluation consisting of the self evaluation and the course evaluation can be found in Appendix B.

Note that normally such self-evaluations can or should be used during a course and not retrospectively. This can give valuable feedback to the teacher and enables him or her to e.g. add additional examples or material to the course especially for these areas where students say they still do not feel confident. In our case this was not possible due to the rigid time constraints. So this self-evaluation can give us only clues what to change or add in next courses.

Table 4.2 shows the accumulated answers of the students about their self-evaluation. From the nine participating students seven filled out and returned the questionnaire. The results show that the students have a good feeling about their own competencies. Nearly all ticks in the column “Help me!” resulted from only one student. We assume that this student might have been the one who missed three out of the seven classes. This seems obvious because those competencies are all coming from those classes that were missed. Since there was no time for repetitions it is clear that he or she would need some help when confronted with problems coming from those areas. The numbers also gives us feedback about which topics were perceived harder than others by the students. These are especially the parts containing propositional and first order logic and the one about the different architectures of agents. One can also argue that the latter one is not very essential for the practical use of AI. Overall we are satisfied with the results. The feedback tells us that it is possible to teach foundations of artificial intelligence understandably –

Competency: I can ...	Confident	Unsure	Help me!
... explain and apply basic data structures like stack and queue on the basis of examples	◯	◯	◯
... explain and apply basic data structures like graph and tree on the basis of examples	◯	◯	◯
... construct and explain finite automatons to be used as simple controllers	◯	◯	◯
... explain basic properties of artificial intelligence	◯	◯	◯
... distinguish between some basic agent structures on which the most intelligent systems are built upon	◯	◯	◯
... explain the differences between and the advantages/disadvantages of reflex based vs. model based agents	◯	◯	◯
... explain the differences between static and learning agents	◯	◯	◯
... explain how to visualise a labyrinth in form of a graph and how to build it	◯	◯	◯
... formulate a simple problem as a graph	◯	◯	◯
... explain the differences between graphs and trees	◯	◯	◯
... how to find a solution to a problem which is given as a graph	◯	◯	◯
... explain the difference between the so called informed and uninformed search strategies	◯	◯	◯
... explain the differences between the breadth first and the depth first search and know about their dis-/advantages	◯	◯	◯
... explain the principles behind the $A^\star$ search algorithm	◯	◯	◯
... explain the ideas of classic logic based planning on the basis of simple examples	◯	◯	◯
... list some differences between propositional logic and first order logic	◯	◯	◯
... recite some strategies of machine learning algorithms	◯	◯	◯

Table 4.1: The matrix used to assess the students by themselves. For each competency (row) the students can tick one of the possible self-evaluations (columns) ranging from "I feel confident in doing this", "I can do it but i feel a bit insecure" to "I definitely need help!". This questionnaire can be filled out anonymously.

even in high school settings without the proper backgrounds in mathematics and computer science at an university level.

The foreign evaluation done by the teacher is basically the same but instead of the axis on which students rate themselves, the teacher has a list with the name of the students. In each corresponding field of the matrix the teacher now can takes notes which student has already achieved which competency and to which degree (e.g. "Not achieved" – "Achieved" – "Over-achieved"). The foreign evaluation by the teacher can be done during course. Thereby the teacher always keeps an overview about the progress of all the

Competency: I can ...	Confident	Unsure	Help me!
... explain and apply basic data structures like stack and queue on the basis of examples	7		
... explain and apply basic data structures like graph and tree on the basis of examples	7		
... construct and explain finite automatons to be used as simple controllers	7		
... explain basic properties of artificial intelligence	5	2	
... distinguish between some basic agent structures on which the most intelligent systems are built upon	4	2	1
... explain the differences between and the advantages/disadvantages of reflex based vs. model based agents	3	3	1
... explain the differences between static and learning agents	4	2	1
... explain how to visualise a labyrinth in form of a graph and how to build it	6	1	
... formulate a simple problem as a graph	7		
... explain the differences between graphs and trees	6	1	
... how to find a solution to a problem which is given as a graph	6	1	
... explain the difference between the so called informed and uninformed search strategies	4	2	1
... explain the differences between the breadth first and the depth first search and know about their dis-/advantages	7		
... explain the principles behind the $A^\star$ search algorithm	7		
... explain the ideas of classic logic based planning on the basis of simple examples	3	3	1
... list some differences between propositional logic and first order logic	1	5	1
... recite some strategies of machine learning algorithms	2	4	1

Table 4.2: The analysis of the self-evaluations filled out by the students. Of the nine participating students seven returned the questionnaire. The arithmetic mean values and corresponding standard deviations of the columns are 4.55/1.64, 2.1/1.1, 1/0 and 1/0 respectively.

students and is able to steer the individual learning processes of the students. In case that the course has to be graded e.g. due to some legal requirements this could still be done afterwards. This can be done by a simple mapping scheme that has to be adapted from course to course. Like e.g.

- When a student achieves all competencies the grade is positive.
- Define a threshold of a number of over-achieved competencies above which a student gets the best possible grade, e.g. 50 per cent.

- Introduce a linear function that maps any number of over-achieved competencies between 0 and the threshold defined above to an adequate grade in between.

Such foreign evaluations can be used as a feedback tool too. In this case as a feedback from teachers to their students. When used during a course where students have opportunities so show their competencies it gives the students an overview what topics they should still work on. A student has successfully finished a course, when he or she has mastered all defined competencies. In our case we did not do such a structured foreign evaluation because we had no necessity to grade the students afterwards.

4.4 Evaluation of the course

The course itself and the teacher were evaluated too. We selected questions that should give us feedback on several measures that can be grouped as (i) the contents, (ii) the presentation and (iii) the dealing with the students. The translated questions and the scale can be seen in Table 4.3. The original questionnaire can again be found in Appendix B on page 85.

That results of the evaluation of the course are given in Table 4.4. One can see that most of the students fully agree or nearly fully agree with most of the statements. Very positive aspects regarding the design of the course are, that the students agree to a high degree to the statements that they have learnt something new, that they could actively participate and that they are motivated to deepen their knowledge in the area of AI.

On the downside is, that one student felt that his or her previous knowledge has not been taken into account very well. This was probably the case because the students came from three different age groups (and grades). In future courses we could avoid this either by only admitting students coming from the same grade or by offering supportive material for those who are lacking certain foundations. One student said that he or she would rather not agree to have learnt things of practical use. This is definitive a shortcoming in this course. More time could have been given especially to the applied exercises. On the other hand the overview about the field of AI would have been much narrower. After this course it seems clear that such an extensive introduction including more practical exercises and application can only be done with much more time slots.

Statement	1	2	3	4	5	6
I have learnt something new during the iRobot course	◯	◯	◯	◯	◯	◯
I have the feeling that the learnt is also of practical use to me	◯	◯	◯	◯	◯	◯
The teacher has considered my previous knowledge and designed the course based on that	◯	◯	◯	◯	◯	◯
The contents were presented comprehensible	◯	◯	◯	◯	◯	◯
I think the teacher has explained the topics comprehensible and replicable	◯	◯	◯	◯	◯	◯
I could actively participate in the lessons	◯	◯	◯	◯	◯	◯
The teacher responded to my questions and answered them well	◯	◯	◯	◯	◯	◯
I feel motivated to further dive into the topics	◯	◯	◯	◯	◯	◯
The teacher treated the students correct and respectfully	◯	◯	◯	◯	◯	◯
The teaching material supported the comprehension of the contents	◯	◯	◯	◯	◯	◯
I would like to participate in a sequel of this course	◯	◯	◯	◯	◯	◯
What should be retained in future courses?						
What should be changed or enhanced in future courses?						

Table 4.3: The matrix used to evaluate the course itself by the students. For each question (row) the students could tick one of the check boxes ranging from "I fully agree" to "I totally disagree" corresponding to the numerical numbers from 1 to 6 respectively. Additionally a remark could be given for each statement on each row. The last two question left room for open answers. This questionnaire could be filled out anonymously.

The only student who stated he or she would rather not participate in a sequel of this course gave a very positive feedback to the rest of the items. Since we did not have the possibility to ask why he or she gave this answer, it could also just be a personal reason. Generally we have the feeling that some of the students had different expectations of this course (immediate and simple solution to robotics problem in the RoboCup and more direct input for their final papers).

The realisation has been perceived quite well. The presentation of the contents, the material for the course (including the exercises) and especially the explanations have reached very high approval rates. Finally, a very nice personal feedback for a teacher

Statement	1	2	3	4	5	6
I have learnt something new during the iRobot course	6	1				
I have the feeling that the learnt is also of practical use to me	3	2	1	1		
The teacher has considered my previous knowledge and designed the course based on that	2	4		1		
The contents were presented comprehensible	5	2				
I think the teacher has explained the topics comprehensible and replicable	6	1				
I could actively participate in the lessons	6	1				
The teacher responded to my questions and answered them well	5	2				
I feel motivated to further dive into the topics	4	2	1			
The teacher treated the students correct and respectfully	7					
The teaching material supported the comprehension of the contents	3	4				
I would like to participate in a sequel of this course	3	2	1	1		

Table 4.4: The results of the evaluation of the course. Remarks given by the students and the answers to the open questions can be found in the text.

is the one from question nine, where all students fully agreed with the statement "The teacher treated the students correct and respectfully".

The answers to the open questions augmented the impressions we already suspected. When asked what should be changed or enhanced in further such courses the students replied:

- More units!
- Better enthuse the weaker students.
- More applied exercises would be reasonable for these topics.
- More content per class should be done!
- Homework.
- The assignment took too much time.

The latter two items probably mean the same, i.e. that the assignments we gave to them were quite challenging and also would have been very time consuming partially. Knowing this in advance we did not make the assignments compulsory but rather presented them as suggestions how one could apply the theory with robots. The answers also show the different previous knowledge of the students. While some say the course and the assignment were too challenging, other would want even more content in shorter time.

This dilemma could be tackled with in future courses e.g. by offering optional additional material or assignments for the higher motivated students.

The second open question about what should be retained in similar future courses provided us with the following answers:

- Friendliness, responsiveness
- Openness, answering questions
- Responsive to questions
- Open for talks, insightful
- Slides very good, structured and aggregated
- Method how knowledge is presented (e.g. figures ...)

As one can see on the original evaluation sheets that can be found in Appendix Appendix B, the students also had to possibility to give additional textual feedback to each question of the course evaluation. These open answers emphasise the scaled answers above. Two positive answers regard the presentation of some contents and the material. Interestingly, most of these answers are directly related to the teacher as a person and do not relate to the contents of the course. This is a rare and valuable feedback because evaluations of teachers are typically not done systematically at Austrian high schools. An incidentally but very pleasant result.

Finally, the evaluation also asked to give some general feedback to the course itself. Here, three comments came from the students (without any special order):

- Offer more such courses but better during the autumn semester because there is more time.
- Our (i.e. other) teachers promised more direct connections to the topics of our final papers.
- The classes were very exciting and educational, sometimes a bit unchallenging.

The first two items regard the organisation of the course itself. We had no possibility to influence these points during the design phase of our course, nor were they considered for this prototypic course. For future courses it will definitely be important to think about the organisational requirements as well. Especially when we want to enable self-directed

learning. This learning model requires some amount of time that has to be given to the attending students. The last item reflects the high level of knowledge and motivation of some of the students in this class. While the theory in this course was admittedly taught on a very high level we never had the feeling that the students could not follow it. More than that at least one student had the feeling to be unchallenged. [Meyer, 1987b] also discusses methods to better deal with such highly motivated students. This would probably not be the case in other classes or some other schools. Looking back, we had a quite unique class attending this course.

Chapter 5

Discussion and Outlook

Like classic literacy which includes writing, reading and mathematics, literacy in AI/-computer science will become a major issue in future. Furthermore, with AI literacy pupils also receive a solid preparation for subsequent studies at university level and their future career. Therefore, we presented an educational project teaching fundamental concepts of artificial intelligence and computer science at high school level (grades 9-11). It was explained in detail what the general parameters were and argued a design for an AI course with 7 units of two hours each. We conducted and empirically evaluated a pilot project in a representative Austrian high school with nine voluntarily participating students in spring 2015. Weekly courses held by university researchers covered major AI topics. These included a basic discussion about artificial intelligence, the introduction of some fundamental data structures like stack, queue, graph and tree, some methods that are commonly applied by intelligent agents like searching or planning and an outlook to real self-learning systems that could be dealt with in future courses.

The contents were chosen according to current AI education literature and adapted and structured with respect to the students' prior knowledge and educational background. The units focused on the students' experiences and built the new knowledge on the basis of these. Together with a mixture of theory, various work settings (i.e. solitary work, group work, open discussions, learning stations etc.) and hands-on projects we employed a constructionistic learning model as good as it was possible.

The course was finally evaluated in two ways. First, the students had to self-evaluate their acquired competencies and second the course itself was evaluated in terms of content, presentation, methods and dealing with the students.

5.1 Conclusions

The results indicate that the participating students are confident about the various topics addressed during the teaching units and that they acquired the necessary competencies. The first pilot project was successful, nevertheless there were some drawbacks and shortcomings to be dealt with in future realisations of such a course.

First, for a well founded scientific evaluation of the course the sample of the participating students was quite small with only nine attending. Additionally, they had a quite different previous knowledge because they were coming from three different grades. So they formed a very inhomogenous group in terms of prior knowledge. What connected them was their interest in the RoboCup leagues. This built a common basis to work with.

Second, they had different expectations that were raised prior to the course by other teachers. Some expected a focused RoboCup tutorial with an artificial intelligence input, some thought the classes would be the background information they need for the final papers. Clearly, we could not meet all these expectations in this short course. Sadly, this led to some disappointments that had little to do with the course itself.

Summarizing the results of the evaluations, the project succeeded in teaching high school pupils the foundations of artificial intelligence. The participating students got a well founded understanding of at least the concepts we presented and discussed and the growing importance of artificial intelligence in every day life. They are now familiar with various topics addressed during the weekly teaching units and will benefit from the acquired content in their future, e.g. when participating in robotics competitions or starting engineering or science studies at universities.

5.2 Future Work

The results of the evaluations and the lessons learned from the pilot implementation of the project form the basis to adapt and improve the present AI course. We are planning to conduct the project in other high schools in the next few years, pursuing our long-term goal of integrating artificial intelligence topics into the regular science education in high schools and to foster 'AI literacy'.

What we clearly underestimated during this course was the amount of work they had to do for other courses of their normal curriculum. Since the end of the semester came nearer a lot of tests and homework had to be done for the compulsory courses. This left not much time to do the ("more interesting" like some said) assignments from this course. Thus we left out the home work and converted them to optional assignments and pointers to practical solutions of the theoretical inputs.

What also should be changed in future courses is the time structure of the course which was not very beneficial. We had too few classes to talk about and discuss all the different foundations of AI that we intended to include. Additionally, the course was quite compact placed at the end of the semester. It would have been better to at least have the seven units spread over the whole semester. This way the students would have had more time between the courses for their studies and some practical implementations of the algorithms and agents. This way to the students it felt like too much theory in comparison to practical work. This compact course setting also left too little time for our intended self directed learning scenario.

Overall we are satisfied with the results. The feedback tells us that it is possible to teach foundations of artificial intelligence understandably even in high school settings.

Bibliography

Alimisis, D. and Kynigos, C. (2009). Constructionism and robotics in education. *Teacher Education on Robotic-Enhanced Constructivist Pedagogical Methods*, pages 11–26.

Altin, H. and Pedaste, M. (2013). Learning approaches to applying robotics in science education. *Journal of Baltic Science Education*, 12(3):365–377.

Baier, C. and Katoen, J.-P. (2008). *Principles of Model Checking*. MIT Press.

Biere, A., Heule, M., Maaren, H. V., and Walsh, T., editors (2009). *Handbook of Satisfiability*. Frontiers in Artificial Intelligence and Applications. IOS Press.

Brachman, R. J. and Levesque, H. J. (2004). *Knowledge Representation and Reasoning*. Artificial Intelligence. Morgan Kaufmann.

Braitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology*. MIT Press, Cambridge.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23.

Bundesministerium für Bildung und Frauen (2013). *Die kompetenzorientierte Reifeprüfung Informatik – Richtlinien und Beispiele für Themenpool und Prüfungsaufgaben*. Bundesministerium für Bildung und Frauen.

Burgsteiner, H. (2006). Imitation learning for real-time robot control. *International Journal of Engineering Applications of Artificial Intelligence*, 19:741–752.

Burgsteiner, H., Kandlhofer, M., and Steinbauer, G. (2016). iRobot: Teaching the basics of artificial intelligence in high schools. In *Proceedings of the 6th AAAI Symposium on Educational Advances in Artificial Intelligence*, pages 4126–4127, Phoenix, USA. Association for the Advancement of Artificial Intelligence, AAAI Press.

Cassandras, C. G. and , S. L. (2008). *Introduction to Discrete Event Systems*. Springer.

Dechter, R. (2003). *Constraint Processing*. Artificial Intelligence. Morgan Kaufmann.

Deming, W. E. (1986). *Out of the Crisis*. Massachusetts Institute of Technology.

Deutsch, M. and Burgsteiner, H. (2016). Imitation learning for real-time robot control. *Studies in Health Technologies and Informatics*, page to be published.

Diekmann, A. (2007). *Empirische Sozialforschung*. Rowohlt.

Dilger, W. (2005). *Künstliche Intelligenz in der Schule*. Technische Universität Chemnitz.

Edelkamp, S. and Schroedl, S. (2011). *Heuristic Search: Theory and Applications*. Morgan Kaufmann.

Featherston, E., Sridharan, M., Urban, S., and Urban, J. (2014). Dorothy: Enhancing bidirectional communication between a 3d programming interface and mobile robots. In *5th Symposium on Educational Advances in Artificial Intelligence*. Association for the Advancement of Artificial Intelligence, AAAI Press.

Fikes, R. and Nilsson, N. (1971). Strips: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

Fok, S. and Ong, E. (1996). A high school project on artificial intelligence in robotics. *Artificial Intelligence in Engineering*, 10(1):61–70.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explorations*, 11(1).

Heinze, C., Haase, J., and Higgins, H. (2010). An action research report from a multi–year approach to teaching artificial intelligence at the k–6 level. In *1st Symposium on Educational Advances in Artificial Intelligence*.

Herculano-Houzel, S. (2009). The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in Human Neuroscience*.

Hester, T., Quinlan, M., and Stone, P. (2010). Generalized model learning for reinforcement learning on a humanoid robot. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Jorgensen, D. L. (1989). *Participant Observation: A Methodology for Human Studies*. Sage.

Kalmar, Z., Szepesvari, C., and Lorincz, A. (1998). Module-based reinforcement learning: Experiments with a real robot. *Machine Learning*, 31:55—-85.

Kumar, D. and Meeden, L. (1998). A robot laboratory for teaching artificial intelligence. In *Proceedings of the Twenty-ninth SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '98, pages 341–344, New York, NY, USA. ACM.

Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). Soar: an architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64.

Mabe, P. A. and West, S. G. (1982). Validity of self-evaluation of ability: A review and meta-analysis. *Journal of Applied Psychology*, 67(3):280–296.

McGovern, A., Tidwell, Z., and Rushing, D. (2011). Teaching introductory artificial intelligence through java-based games. In *2nd Symposium on Educational Advances in Artificial Intelligence*. Association for the Advancement of Artificial Intelligence, AAAI Press.

Meyer, H. (1987a). *Praxisbuch Meyer: Unterrichtsmethoden I, Theorieband*. Cornelsen.

Meyer, H. (1987b). *Praxisbuch Meyer: Unterrichtsmethoden II, Praxisband*. Cornelsen.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.

Nau, D. and Ghallab, M. (2004). *Automated Planning: Theory and Practice*. Artificial Intelligence. Morgan Kaufmann.

Nilsson, N. J. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1(1):139–158.

Papert, S. (1993). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, 2nd edition.

Prediger, S. (2008). Mit der Vielfalt rechnen – Aufgaben, Methoden und Strukturen für den Umgang mit Heterogenität im Mathematikunterricht. In Hußmann, Liegmann, Nyssen, Racherbäumer, and Walzebug, editors, *Indive – Individualisieren, Differenzieren, Vernetzen*. Franzbecker, Hildesheim.

RPVO (2012). Verordnung der Bundesministerin für Unterricht, Kunst und Kultur über die Reifeprüfung in den allgemein bildenden höheren Schulen (Prüfungsordnung AHS) StF: BGBl. II Nr. 174/2012 mit den Änderungen BGBl. II Nr. 264/2012 und BGBl.

II Nr. 47/2014. URL: https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20007845.

Russel, S. and Norvig, P. (2012). *Artificial Intelligence – a modern approach*. Pearson, 3rd edition.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575.

Schwill, A. (1997). Computer science education based on fundamental ideas. In Passey, D. and Samways, B., editors, *Proceedings of Information Technology – Supporting change through teacher education*, pages 285 — 291. Chapman Hall.

Selkowitz, R. I. and Burhans, D. T. (2014). Shallow blue: Lego-based embodied ai as a platform for cross-curricular project based learning. In *5th Symposium on Educational Advances in Artificial Intelligence*. Association for the Advancement of Artificial Intelligence, AAAI Press.

Spohrer, M. (2009). *Konzeption und Analyse neuer Maßnahmen in der Fort- und Weiterbildung von Informatiklehrkräften*. PhD thesis, Technical University Munich.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Torrey, L. (2012). Teaching problem-solving in algorithms and ai. In *3rd Symposium on Educational Advances in Artificial Intelligence*. Association for the Advancement of Artificial Intelligence, AAAI Press.

Turing, A. M. (1948). Intelligent machinery, a heretical theory. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, page 105.

Utting, M. and Legeard, B. (2006). *Practical Model-Based Testing: A Tools Approach*. Morgan Kaufmann, 1 edition.

Wollowski, M. (2014). Teaching with watson. In *5th Symposium on Educational Advances in Artificial Intelligence*. Association for the Advancement of Artificial Intelligence, AAAI Press.

Wollowski, M., Selkowitz, R., Brown, L. E., Goel, A., Luger, G., Marshall, J., Neel, A., Neller, T., and Norvig, P. (2016). A survey of current practice and teaching of ai. In *Proceedings of the 6th AAAI Symposium on Educational Advances in Artificial Intelligence*, pages 4119–4124, Phoenix, USA. Association for the Advancement of Artificial Intelligence, AAAI Press.

Internet sources

Aldebaran Robotics (2016). Discover Nao, the little humanoid robot from Aldebaran. URL: https://www.aldebaran.com/en/cool-robots/nao. Last accessed 2016-01-26.

BRG Kepler (2015). Bundesrealgynasium Keplerstrasse Graz. URL: http://www.brgkepler.at/www/. Last accessed 2016-01-26.

Bundesministerium für Bildung und Frauen (2016). AHS-Lehpläne Oberstufe Neu - Informatik. URL: https://www.bmbf.gv.at/schulen/unterricht/lp/lp_neu_ahs_14_11866.pdf. Last accessed 2016-01-26.

Futschek, G. (2012). Prolog der Informatik - Algorithmen Teil 1. URL: http://www.informatik.tuwien.ac.at/studium/studierende/prolog/algorithmen-teil1.pdf. Last accessed 2016-02-08.

Future of Life Institute (2015). Research priorities for robust and beneficial artificial intelligence - an open letter. URL: http://futureoflife.org/ai-open-letter/. Last accessed 2016-04-04.

Google Inc. (2016). Google self-driving car project. URL: https://www.google.com/selfdrivingcar/. Last accessed 2016-04-04.

Stangl, W. (2016). Lernziele – Taxonomien von Lernzielen. URL: http://arbeitsblaetter.stangl-taller.at/LERNZIELE/. Last accessed 2016-04-11.

The RoboCup Federation (2016). RoboCup. URL: http://www.robocup.org/. Last accessed 2016-01-26.

Appendix A

Exercises

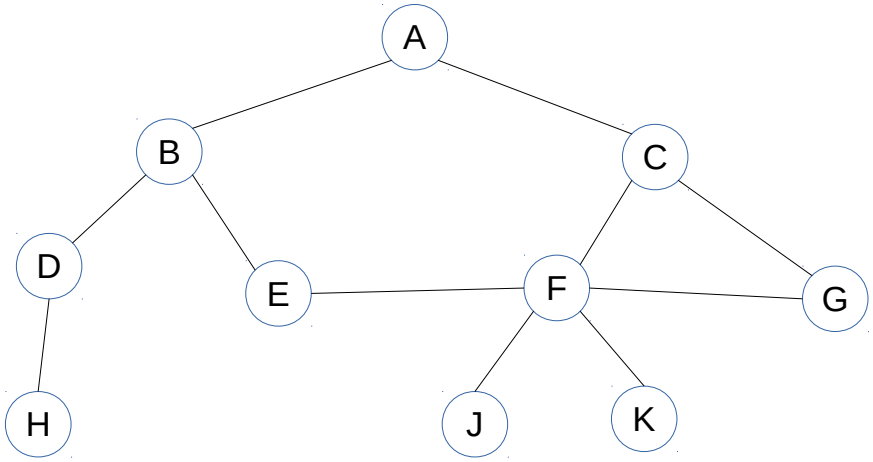
This Appendix contains the original PDFs for the interested reader. These had been distributed in printed form amongst the students during the lessons. The exercises were realised in form of group works of the students in which they had to find experimentally find solutions to the problems and discuss them together.

A.1 Exercises for Searching

The examples in this section were used for the exercises about problem solving by searching in trees. The students had to work in groups as it is described in Section 3.5 on page 30. Each student got his or her own sheet of paper on which they could write down their solutions. The following pages show the original (German) PDFs that had been given to the students.

Breitensuche

Lieber alle Möglichkeiten gleichzeitig probieren ...



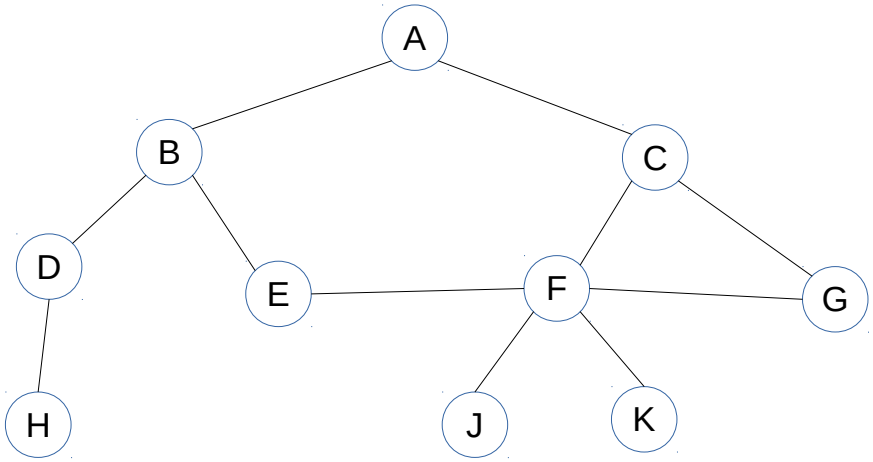
Algorithmus

```
Breitensuche (G, s)
// G … Graph, s … Startknoten (hier: Knoten A)

Lege neue, leere Schlange Q an
Markiere s als besucht
Q.add(s)
Solange Q nicht leer ist {
    k = Q.remove()
    Gib k aus
    Mit allen Nachbarn j von k in G {
        Markiere j als besucht
        Q.add(j)
    }
}

Queue:
```

Ausgabe:

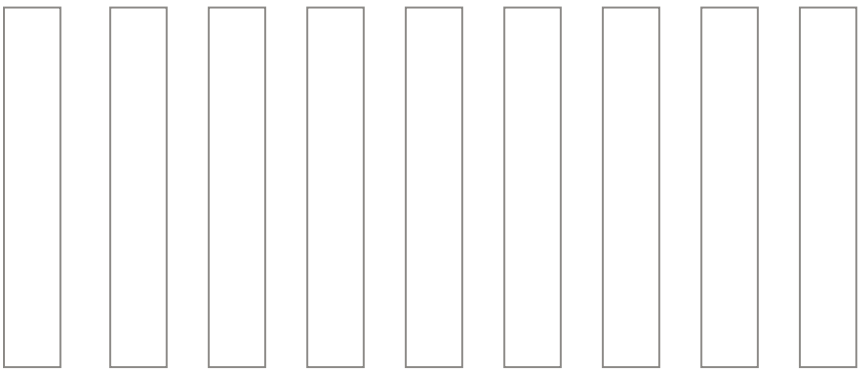
Tiefensuche

Besser mal einen Weg bis zum Ende probieren ...



Algorithmus

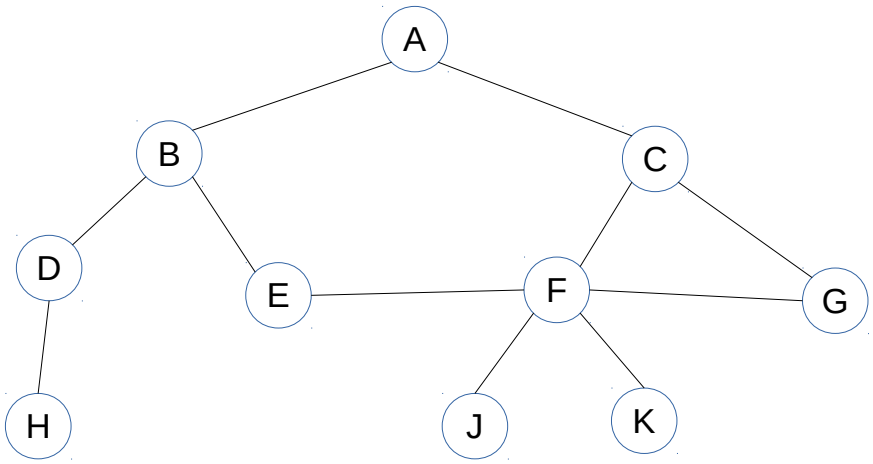
```
Tiefensuche (G, s)
// G … Graph, s … Startknoten (hier: Knoten A)

Lege neuen, leeren Stapel S an
Markiere s als besucht
S.push(s)
Solange S nicht leer ist {
    k = S.pop()
    Gib k aus
    Mit allen Nachbarn j von k in G {
        Markiere j als besucht
        S.push(j)
    }
}

Stack:
```



```
Ausgabe:
```

A*-Bestensuche

Erst mal genau überlegen, welchen Weg ich nehme ...



Algorithmus

Vorinformation: Der Algorithmus versucht immer einen optimalen Weg zu finden. Dazu vergleicht er die Kosten „f“ für verschiedene mögliche nächste Knoten. Diese Kosten bestehen aus

- den Kosten um vom Start zum einem Knoten zu kommen („g“) und
- den (geschätzten) Kosten um vom nächsten Knoten zum Ziel zu kommen („h“)

In unserem Beispiel werden beide Kosten zB als gemessene Strecke in [cm] angenommen.

```
A*-Suche (G, s, z)
// G … Graph, s … Startknoten (hier: Knoten A),
              z … Zielknoten (hier: Knoten J)
Lege neue, leere Felder „Offen“ und „Gesehen“ an
f = 0 + Luftlinie(s,z)
s.notiereF(f); s.notiereG(0)
Offen.add(s)
Wiederhole {
   besteWahl = Offen.EntferneKnotenMitKleinstemFWert()
   Wenn (besteWahl == z) dann
      gib zurück: „Weg gefunden“ (und beende)
   Gesehen.add(besteWahl)
   Mit allen Nachbarn j von besteWahl in G {
      Wenn Gesehen.enthältNicht(j) {
         g_tmp = g(besteWahl) + Luftlinie(j,besteWahl)
         Wenn Offen.enthältNicht(j) oder g_tmp < j.gWert() {
            j.notiereG(g_tmp)
            j.notiereRückweg(besteWahl)
            // f Wert in der offenen Liste aktualisieren
            f := g_tmp + Luftlinie(j,z)
            j.notiere(f)
            Offen.EinfügenOderErsetzen(j)
         }
      }
   }
}
bis Offen leer ist
Gib zurück: „Keinen Weg gefunden“
```

A.2 Exercise sheets for Planning

For the exercises about planning we used the following two examples. Both are scrambled versions of known planning problems as it is described in Section 3.6 on page 34. The sheet of paper on which the examples were printed had been cut into stripes to allow the students to arrange the individual actions in the correct sequence.

Example one represents the Monkey-Banana-Dilemma, example two is the spare tire problem from [Russel and Norvig, 2012]. The following two pages show the original (German) PDFs that had been given to the students.

Planungsproblem 1

Initial state: (Ta(A) ∧ Veell(Oool) ∧ Xotab(C) ∧ Naassabta(B))
Goal state: (Eahv(Nnaaabs))

Action (Brrrk(X, Y)
Preconditions: Ta(X) ∧ Veell(Oool)
Effect: ¬Ta(X) ∧ Ta(Y))

Action (Vrrzch(Bzzzktkt)
Preconditions: Ta(Bzzzktkt) ∧ Xotab(Bzzzktkt) ∧ Veell(Oool)
Effect: Veell(HHgi) ∧ ¬Veell(Oool))

Action (Knoimff(Bzzzktkt)
Preconditions: Ta(Bzzzktkt) ∧ Xotab(Bzzzktkt) ∧ Veell(HHgi)
Effect: Veell(Oool) ∧ ¬Veell(HHgi)

Action (Arrrgh(X, Y)
Preconditions: Ta(X) ∧ Xotab(X) ∧ Veell(Oool)
Effect: Xotab(Y) ∧ ¬Xotab(X) ∧ Ta(Y) ∧ ¬Ta(X))

Action (Ckckch(Bzzzktkt)
Preconditions: Ta(Bzzzktkt) ∧ Naassabta(Bzzzktkt) ∧ Veell(HHgi)
Effect: Eahv(Nnaaabs))

Planungsproblem 2

Initial state: (Ommm(Pppuink, Lxea) ∧ Ommm(Sssrape, Ktrnu))
Goal state: (Ommm(Sssrape, Lxea))

Action (Krrtnass(Sssrape, Ktrnu),
Precondition: Ommm(Sssrape, Ktrnu)
Effect: ¬Ommm(Sssrape, Ktrnu) ∧ Ommm(Sssrape, Drrognu))

Action (Krrtnass(Pppuink, Lxea),
Precondition: Ommm(Pppuink, Lxea)
Effect: ¬Ommm(Pppuink, Lxea) ∧ Ommm(Pppuink, Drrognu))

Action (Uoptn(Sssrape, Lxea),
Precondition: Ommm(Sssrape, Drrognu) ∧ ¬Ommm(Pppuink, Lxea)
Effect: ¬Ommm(Sssrape, Drrognu) ∧ Ommm(Sssrape, Lxea))

Action (Ssslaneesh,
Precondition:
Effect: ¬Ommm(Sssrape, Drrognu) ∧ ¬Ommm(Sssrape, Lxea) ∧ ¬Ommm(Sssrape, Ktrnu)
∧ ¬Ommm(Pppuink, Drrognu) ∧ ¬Ommm(Pppuink, Lxea))

Appendix B

Questions during the evaluation

B.1 Anonymous questionnaire for the self evaluation and the course evaluation

The following pages contain the original PDFs from the evaluations used after the last class lecture. Both, the self evaluation and the course evaluation had been done with one document. In Chapter 4 the translated version of this questionnaire can be found.

Evaluierung und Selbstevaluierung

Bitte bewerte die folgenden Aussagen, je nachdem wie stark sie für dich persönlich zutreffen, auf einer Skala von 1 … „trifft vollkommen zu“ bis 6 … „trifft überhaupt nicht zu“. Darunter kannst du bitte bei jeder Aussage eventuelle Anmerkungen dazuschreiben.

	1 … 2 … 3 … 4 … 5 … 6
1. Ich habe im Rahmen von „iRobot“ Neues gelernt. Anmerkungen:	O O O O O O
2. Ich habe das Gefühl, dass das Gelernte für mich von praktischem Nutzen ist. Anmerkungen:	O O O O O O
3. Der Lehrer hat sich mit meinem Wissensstand auseinandergesetzt und darauf aufbauend die LV gestaltet. Anmerkungen:	O O O O O O
4. Die Inhalte wurden verständlich dargestellt. Anmerkungen:	O O O O O O
5. Ich finde der Lehrer hat verständlich und nachvollziehbar erklärt. Anmerkungen:	O O O O O O
6. Ich konnte mich aktiv in den Unterricht einbringen. Anmerkungen:	O O O O O O

 iRobot

7. Der Lehrer ist auf meine Fragen eingegangen und hat sich gut beantwortet. Anmerkungen:	O	O	O	O	O	O
8. Ich fühle mich motiviert, mich mit den Themen aktiv auseinanderzusetzen Anmerkungen:	O	O	O	O	O	O
9. Der Lehrer verhält sich im Umgang mit den Schüler_innen korrekt bzw. respektvoll Anmerkungen:	O	O	O	O	O	O
10. Die eingesetzten Lehrmittel unterstützen die Erarbeitung des Stoffes sinnvoll Anmerkungen:	O	O	O	O	O	O
11. Ich würde gerne an einer Fortsetzung dieses Kurses im nächsten Semester teilnehmen. Anmerkungen:	O	O	O	O	O	O
12. Was sollte unbedingt beibehalten werden?						
13. Was sollte verbessert bzw. geändert werden?						

 iRobot

Nun geht es noch um eine Einschätzung deiner erworbenen Kompetenzen. Bitte beantworte die folgenden Fragen so ehrlich wie möglich. Es geht nicht um deine Beurteilung, sondern um die des Kursaufbaus und -inhaltes bzw. dessen Gestaltung. Kreuze bitte an: „Da fühle ich mir relativ sicher, das hat schon gut funktioniert." (Links), „Da bin ich mir noch etwas unsicher, das muss ich noch weiter üben." (Mitte) oder „Das kann ich gar nicht, da brauche ich Hilfe." (Rechts)

	Sicher / Unsicher / Hilfe!		
14. Ich kann grundlegende Datenstrukturen wie Stack und Queue anhand von Beispielen erklären und anwenden.	O	O	O
15. Ich kann grundlegende Datenstrukturen wie Graphen und Bäume anhand von Beispielen erklären und anwenden.	O	O	O
16. Ich kann „endliche Automaten" für einfache Steuerungen erklären und konstruieren.	O	O	O
17. Ich kann prinzipielle Eigenschaften von künstlicher Intelligenz erklären.	O	O	O
18. Ich kann einige grundlegende Agentenstrukturen auf die die meisten intelligenten Systeme aufbauen unterscheiden.	O	O	O
19. Ich kann die Unterschiede zwischen bzw. Vor- und Nachteile von reflexbasierten und modellbasierten Agenten erklären.	O	O	O
20. Ich kann Unterschiede zwischen statischen und lernenden Agenten erklären.	O	O	O
21. Ich kann erklären, wie man ein Labyrinth als Graph darstellen und aufbauen kann.	O	O	O
22. Ich kann ein einfaches Problem als Graph formulieren.	O	O	O
23. Ich kann die Unterschiede zwischen Graphen und Bäumen erklären.	O	O	O
24. Ich kann erklären, wie man ein Problem, das als Graph vorliegt, durch Suchen zu einer Lösung kommt.	O	O	O
25. Ich kann den Unterschied zwischen der sogenannten „informierten" und „uninformierten" Suche erklären.	O	O	O
26. Ich kann die Unterschiede zwischen Breiten- und Tiefensuche erklären und weiß von den jeweiligen Vor- und Nachteilen.	O	O	O
27. Ich kann die prinzipielle Vorgehensweise des A*-Suchalgorithmus verstehen und erklären.	O	O	O
28. Ich kann die Ideen der klassischen, logikbasierten Planung anhand von einfachen Beispielen erklären.	O	O	O
29. Ich kann einige Unterschiede und Gemeinsamkeiten zwischen Aussagen- und Prädikatenlogik aufzählen.	O	O	O
30. Ich kann einige maschinelle Lernstrategien aufzählen.	O	O	O

 iRobot

Gibt es etwas, das du uns noch mitteilen möchtest? Eventuell ein allgemeines Feedback?

Danke für deine Mitarbeit in diesem Kurs
und für deine Antworten! ☺

Schöne und erholsame Ferien!

 iRobot

Affidavit

concerning this thesis titled

DESIGN AND EVALUATION OF AN INTRODUCTORY ARTIFICIAL INTELLIGENCE CLASS IN HIGHSCHOOLS

"I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present diploma thesis."

Graz, May 3, 2016

(HARALD BURGSTEINER)