Using Robots in Undergraduate AI Courses at Small Universities

Kenneth Moorman
Transylvania University
Lexington, KY 40508
kmoorman@transy.edu

Dee Parks
Appalachian State University
Boone, NC 28608
dap@cs.appstate.edu

Abstract

Artificial intelligence courses at universities that do not have focused AI tracks may suffer from a lack of student interest unless they are tied to some application of AI that students find appealing. Robots have been used successfully to draw students into AI courses, but their use can take time away from material that instructors feel should be covered in a general course. We present an approach that utilizes small-scale robotics to teach traditional, core AI concepts. The approach is somewhat generalizable and does not rely on any particular textbook. The methodology was used at two universities in the spring of 2009 to good student response. After presenting those results, we discuss ways in which we intend to modify the approach in future semesters.

Introduction

As the field of computer science grows, it becomes increasingly difficult to cover all topics, especially at universities with small programs. Computer science programs tend to rely on the ACM curricula guidelines to create a good balance of courses. Unfortunately, one area that often gets left out is artificial intelligence (AI); in the 2008 revision to the ACM curriculum, for example, AI is allocated only ten core hours (ACM/IEEE 2008). To treat the subject in more depth, many programs will offer an AI course as an elective. However, it is often difficult to attract students to such a course, especially at smaller schools. Although AI pervades many other areas of computer science (and other fields as well), the undergraduate student may have to be "sold" on this.

The question then becomes how to market the course. An AI class is generally aimed at upper-level undergraduates, although an interdisciplinary approach can attract younger students. Some universities take an engineering approach while others follow a software engineering path. Many larger universities use an overview course as a launching point for follow-up courses on more focused topics. Unfortunately, the options at a smaller university are limited. With one terminal course, the problem becomes how to present a wide-range of material in an appealing and effective way.

Another issue is the choice of textbooks. The most popular text is Russell and Norvig's *AI: A Modern Approach* (AIMA) (2003). A new edition of this text came out in November, 2009. Without question, this book is exceptional in its treatment of the field. Unfortunately, both the size and cost make it somewhat less suited for universities where students may be taking one terminal AI course. Only a fraction of the material can be covered in such a course, making students reluctant to invest in the text. It is also possible that the flavor of the course may not lend itself to both the rigorous and broad treatment of the field in Russell and Norvig. For example, the first author has traditionally taught the AI course from a cognitive science perspective which attracts as many non-majors as computer science students. He has found other texts to be more approachable. While the second author's AI course is aimed at junior and senior majors, even they find the amount of material in the text somewhat overwhelming.

To overcome these problems, we incorporated robots as a core component of the AI course. This is not, on the surface, an original idea; however, the particular goals we want to achieve and the use of robots in this pursuit is atypical. We redesigned our courses to achieve the following:

- Present a solid core of the AI discipline, suitable for future work either in special topic undergraduate courses or graduate courses.
- Make the topics independent of the textbook chosen.
- Integrate robotics not by making the course into a robotics class but by making robots the platform on which to explore those core techniques.

Both of the authors have taught their respective courses multiple times with numerous textbooks and areas of focus. In the spring semester of 2009, the two classes were taught for the first time using this robot-driven model. While there were some issues that arose, both authors feel confident that this approach is a significantly useful technique for teaching artificial intelligence as either a cross-disciplinary AI course at a small liberal arts university or as a stand-alone elective course in a typical computer science undergraduate program. Both intend to continue teaching their courses using this approach although we do acknowledge that there were problems in our first pass.

The AI course

Our first goal is to present a solid foundation in artificial intelligence. The topics that need to be covered in such a

course can be debated; that debate will largely be driven by the professors' personal biases and their attempts to overcome those. The ACM guidelines list a number of topics including basic search techniques, advanced reasoning forms, robotics, and perception. Again, we are bound by the style of class at a smaller university and by the students who take the course. The ACM suggests a fair amount of time be spent on *logic*, for example. Unfortunately, we have both found that our students are under-prepared to deal with the complexities of logic beyond the basic level. So we cover fundamentals of logic but make a conscious decision to focus most of our attention on other areas.

After considering the ACM guidelines as well as our own AI courses over the past several years, we created a set of three core ideas to present—*problem solving*, *learning*, and *communication*. These were defined broadly enough to allow us both flexibility in deciding what to include and what to leave out. Furthermore, the three concepts fit nicely into a model of ever widening "agent-ness."

First, an intelligent agent in a non-static environment needs to be a problem solver. Methods to achieve this can vary widely based on the complexity of both the environment and the agent. While exploring this area, students are introduced to the idea of a purely reactive entity, using either the subsumption model (Brooks 1986) or field technique (Arkin 1989). Students are reminded of search techniques taught in previous courses (such as *Data Structures*). The initial discussion quickly moves to the area of informed searching, which leads to an overview of planning. Given the time constraints, advanced planning techniques are not given much consideration. Instead, the class moves towards case-based reasoning as the final model of problem solving.

Second, if we want the agent to be dynamic (i.e., able to respond differently to the environment over time as experience is gained), learning is needed. Since we would like the entities we build to adapt their own behavior, a discussion of various learning techniques is appropriate. Students are given an overview of both supervised and unsupervised methods. Symbolic approaches (e.g., version spaces and decision trees) and subsymbolic ones (e.g., perceptrons, neural networks, and Bayesian techniques) can be presented. Once more, our goal is to lay a foundation. A student coming out of this course should be conversant with the core ideas and primed to go on to graduate work if desired.

We start with a static agent in a changing world and move to a learning agent in a dynamic world. The last step considers groups of agents via communication. This ranges from low-level concepts like simple syntactic parsing through higher levels of discourse understanding. Issues of ambiguity can be introduced along with techniques for handling them. In a historical approach to the issue, scripts and plans are discussed as well as corpus-based techniques. Advanced ideas from communication can be discussed if time allows, such as non-verbal comprehension or emotional understanding.

As desired, the three areas do not tie us to a specific textbook. One author used Russell and Norvig in her course, but the other taught from Nilsson's *Artificial Intelligence: A New Synthesis* (1998). Furthermore, a number of texts have been utilized over the years, and we see no reason this approach could not be adapted to any of them.

In addition to using different texts, we approach the class from fairly different perspectives. The first author prefers to teach the course using a cognitive science approach. His course fulfills an upper-level computer science elective but is also taken by interested students from other disciplines such as philosophy. psychology, or education. The class is cross-listed as a philosophy course. As a result, there is a wide mix of students. While it is a computer science course, the emphasis is on using the computer programming elements to improve comprehension of the concepts. Since some students have no programming experience, scaffolding is provided to allow all students to contribute to the finished products. During the spring semester of 2009, only five out of thirteen students were computer science majors. The others were a mixed group, with two from psychology, and one each from cognitive science, chemistry, English, business administration, philosophy, and physics.

The second author teaches the course as a traditional elective; it is the only AI course and is offered every spring. It is popular with the number of students typically 24 to 28. *Data Structures* is a prerequisite course, and all AI students have taken at least three prior courses in which they do extensive Java programming. During the first half of the spring 2009 course, students attended one lecture and one lab per week in which they worked with the robots. The second author intended to use the same assignments as the first during this period, but timing issues prevented her from assigning either the learning or natural language understanding projects. She will use those in spring 2010.

The robots

As our courses are not intended to be classes in robotics, we do not want the students to spend significant time on the engineering elements. Further, neither institution can invest a great deal of money in the robots themselves. We also want the groups to be able to take the robots with them after class and to be able to use them outside of a formal lab setting. As Kay (2010) pointed out, a potential key to success of robotics in the classroom is precisely whether students can take the kits back to the social side of a campus. Lego Mindstorms seem an obvious choice due to their low-cost, durability, and portability.

Lego released the NXT model of their popular robotics system in 2006 as an improvement over the original RCX model. The NXT computer contains an Atmel 32-bit ARM processor running at 48 MHz. The unit runs with 64 KB of RAM, as well as 256 of non-volatile storage for programs and data (LEGO Group 2006). In addition to the light and touch sensors, the NXT includes a sonar module for proximity sensing. The new motors are both more precise than the older ones and boast an included odometer which gives decent reports. Finally, both authors added compass sensors to their NXT kits.

Although the included NXT-G graphical language is extensive, we chose to use the replacement Lejos NXJ firmware (Bagnall 2007). This firmware installs a Java virtual machine on the robot, includes extensive libraries, per-

mits detailed communication between a host PC and the robot, and is open source. The developers are active and respond quickly to bug reports and suggestions for improvements. Finally, our students are either already conversant in the Java language or can work with someone who is. Therefore, the overhead of this approach is much less than the NXT-G language choice. This is important since we want to use robots with as little overhead as possible.

For the robotic activities prior to the final project, we provide them with a straightforward design of a tricycle-base differential drive robot known as ZippyBot (Perdue 2007). ZippyBot's third "wheel" is actually one of the plastic balls included with the NXT. We find that of all the designs we have tried, ZippyBot is able to make the most precise 90 degree turns and still travel in a relatively straight path.

Assignments

Other work

As mentioned previously, there are numerous examples of courses taught using robots. A variety of projects are used in these courses based on the goals of the course. As Dodds, *et al* point out in their overview, an important element when considering robots is "...how well they fit into the overall syllabus of course topics" (2006). It is not our desire to simply graft a robotics project onto our courses; it is imperative that the assignments be tightly coupled to the existing content. This eliminates many potential approaches. For example, the well-known RoboCup is often used as a springboard to assignments (Dodds et al. 2006; Heintz, Kummeneje, and Scerri 2001). We also avoid robot-centric projects, such as the excellent ones used by Greenwald, *et al* involving localization and obstacle detection (2006). Further, it is not our goal to have assignments which are just as effective without robots; for example, Talaga and Oh (2009) leverage off the extensive tools provided by AIMA to have robots navigate and play board games. The closest model to ours comes from Kumar which presents a traditional AI course that makes use of robotics (2001; 2004) in a fashion similar to our approach.

Group based

Students in both courses work in small groups. In the first author's course, this allows each student to contribute their own skills to the projects and offsets any weaknesses students have due to their respective majors. A team consists of at least one computer science student and one non-major. Since programming from scratch is not an important part of the course, the groups are required to produce short papers (5 to 7 pages) detailing the design choices made during each project, the testing methodology used, and what discoveries were made. The strengths of all are utilized by this approach. On the pedagogical level, it permits the groups to reflect on what they learn from the projects and from one another.

Because of the relatively large number of students in the second author's course and the expense of the robot kits, students work in pairs. All programming and testing is done in these pairs and, in almost all cases, both members of the pair receive the same grade on the assignments. Students self-select their partners and, for the most part, their choices work well. Occasionally, some rearrangements are done.

Length

We assign three focused projects and a more open-ended term project. This allows one project from each of the three themes of the course and has the benefit of permitting the groups to have two or three weeks for each assignment and still have time to work on the term project. Since the courses are taught in the spring semester, it makes logistical sense to shift to the term project about the time of spring break.

Materials

Each assignment is designed to allow the robots to navigate their way through a maze. The two authors take slightly different approaches to the physical construction of the maze. The first author uses 2x4 wooden segments as bases along with foam board for the walls. The second author connects 14 inch sections of 2x8 wooden planks with specially made connectors (straight, L-shaped, and T-shaped). Both maze designs are easy to reconfigure while being sturdy enough that an accidental bump by a robot does not displace a wall.

In order to ease the students into the robotic programming, the first author provides a class file in Lejos. The second author provides a specification for the class but requires the students to create it. For both, the use of the abstraction permits the groups to focus on higher level problems rather than the robotic interface. Some of the methods can be seen in Figure 1. These simply include an interface to the robot that helps to abstract away from the physical elements.

MazeRobot (float diam, float trkWidth, float unit) The constructor for the class receives the wheel diameter, the distance between the wheels, and the length of one maze unit. The *unit* simplifies movement by restricting the robot to moving one space at a time.

void setUpSensors () Sets up the compass.

void moveOneUnit () Move the robot forward.

boolean safeMoveOneUnit() Move the robot forward one unit, but stop and return *true* if the touch sensor is pressed.

void turnBackToZero () Turns the robot back to the 0 heading.

void moveArbitraryDistance (float distance) A dangerous function, this moves the robot an amount of space you specify.

void turnLeft (), void turnRight() Turns 90 degrees to the left or right.

int getSonarValue () Returns the distance to a detected obstacle.

Figure 1: Sample methods provided to students

Problem solving

In order to get the students familiar with the ZippyBot, Lejos, and with robotics in general, the first assignment is a straightforward implementation of two searching techniques. First, the groups implement a depth first search (DFS). The robot is given a simple text file representing the current state of the maze. Using DFS, the robot finds a path from the current location to the goal. A portion of the description from the assignment handout follows:

> **Search first, then run** For our first exercise, you will be provided with a text file representing the physical maze. Your robot will find a solution to the maze using the traditional DFS technique and will then execute the solution. The text file contains a matrix representing the maze. In the matrix, a '0' represents an open space, a '1' represents a wall, an 'S' represents the starting location, and a 'G' represents the goal. Your program should consider possible moves one at a time, following a pathway until there are no possible next moves. At that point the program *backtracks*, returning to the most recent prior state from which a different decision can be made, and tries that move. Backtracking also occurs if the algorithm determines that the next move will create a *cycle*.

Students are often annoyed by their inability to get the robot to travel precisely the amount of distance indicated or to make precise 90 degree turns. We address these issues in the lecture and class discussion. Naturally, the longer the robot navigates through the maze, the worse its performance becomes due to an accumulation of these errors. This is the motivation for the second element of the problem solving assignment—the reactive approach to maze running.

From the assignment:

> **Search while running** Unfortunately for ZippyBot, the DFS approach has some problems. Our robots don't make precise turns and find it difficult to stay in the center of the pathways. We know that we can find our way out of a maze by keeping our left hand on the wall (or our right). This technique works in any maze that doesn't have teleportation cells.
>
> In order to use this idea with the robot, you need to access the sensors. In particular, the robot needs a way to determine if a passageway opens up. If so, the robot needs to turn in that direction. If the robot runs into a wall in front of it, it should turn around 180 degrees and come back down that passageway.

This approach permits the robot to navigate with no advance planning while reacting to dynamic situations. The assignment is designed to launch a discussion of *reactive robotics*; both authors felt it was successful in spring 2009. Note that while we stay away from specific robot discussions such as which drive configuration is best or what sensor fusion is, we do introduce basic robot concepts through the natural flow of the assignments.

The first assignment also demonstrates an important point made by Kumar (2004)—robot projects in an AI course should build on the natural strengths of the platform. Problem solving in a more abstract fashion or even in something like the classic 8-tile puzzle game (Klassner 2002) would not be as successful in our classes since these are not tasks that an NXT robot could easily engage in. The maze task allows the strengths of the platform to be utilized.

Learning

We considered several possibilities for a robotic learning assignment. For example, it is popular to teach neural networks using robot kits (see (Imberman 2003) for one such approach). However, we decided to use the more symbolic approach of decision trees (Quinlan 1987). The theory behind decision trees is a little easier to present to students and permits interesting class discussions of information theory and entropy.

The project begins with the robot randomly attempting to find its way through the maze and storing traces of its paths. We place a black square at the goal so the robot can use its light sensor to determine success. The robot attempts to run the maze several times and transfers the traces to a host computer, thus building a library of positive and negative training cases. The students then use a decision tree program to learn the proper moves for each junction in the maze.

From the assignment:

> **Introduction** This project allows you to explore the use of *induction trees*, in particular an extension of Quinlan's famous ID3 program. The goal is to see if ZippyBot can inductively learn a basic path through a fairly simple maze.
>
> **Description** The robot will be placed in the maze at the starting location. Your code will cause it to make 10 moves or less. Each move will be chosen at random. The sequence will stop when 1) an illegal move is attempted (indicating a fail) or 2) the goal is reached. At that moment, the robot will wait for a button press. This will enable you to connect Zippy to the computer so that it can upload its path. You will repeat this process to gather more test cases.
>
> When all the test cases have been assembled, you will use C4.5, a decision tree program, to attempt to learn a proper series of actions to get the robot through the maze. If this is successful, you will load this path into Zippy and watch as he unerringly arrives at the goal.

Natural language understanding

The goal of the final assignment is to give commands to the robot using natural language. As with the learning project, we examined a variety of techniques, including Roger Schank's work with *conceptual dependency* (1975) and the program *Micro-ELI* (Birnbaum and Selfridge 1981); ultimately, we decided to use a simple Java-based implementation (Martin 2009) of the Earley parser (Earley 1970). This chart-based parser is an efficient (cubic time) handler of context free languages. Since the commands for navigating a maze are straightforward, nothing more powerful is needed.

The assignment is to create a grammar of navigational commands; for example, a group may want the robot to *go forward 3 units, turn left, go forward 5 units, play a song*. The PEP parser produces a parse tree which is then fed into a second program on the host computer. This creates a series of commands to issue to the robot as a standalone Lejos program. After that program is downloaded to the robot, students can watch the robot follow their directions.

Term project

The goal of the final project is to allow the students to relax a little with a more open-ended assignment while at the same time incorporating the concepts taught up to that point. Students are not required to make use of earlier assignments but many do. They submit a prospectus for the work, turn in regular progress reports, and present their project both in class and to the larger campus community. In addition, students write a detailed paper discussing all aspects of their work. Papers are structured using ACM journal specifications. In spring 2009, projects ranged from a self-parking car to a smart house controlled by the NXT brick to an instantiation of Frogger in which the NXT dodged older RCX robots running on tracks. One project that did make direct use of the earlier assignments involved a robot that accepted a natural language set of commands and combined that with a reactive framework to allow the robot to respond to obstacles in the intended path.

Results

Students at both locations provided similar feedback after the spring semester. The overall view can be summarized from this response: "Using robots for the projects was a good idea. It added another level of fun to the projects. However, there is a lot of overhead along with the robot." Notice that every aspect of the robots is not positive for the students. We continually stress that even when the robot is correctly programmed, conditions outside the students' control can cause problems. Walls can and do fall down during demonstrations. Batteries die. The compass sensor is notoriously bad due to a large number of electrical devices in the science buildings. As instructors, we see these occurrences as teaching moments. Projects in the real world suffer from these kinds of distractions, and we are not deterred from our use of robots by dealing with such annoyances.

The groups had varying degrees of success with the projects. All groups at both schools were able to get the searching assignment completed successfully, even though this assignment involved the most original programming. The learning assignment proved to be the most difficult of the three. Students became more frustrated with this assignment due to the effect of accumulated errors from the robot's motion through the maze.

Students in the second author's course found the lab assignments conceptually simple but difficult at run-time due to multiple difficulties with both the robots and the environment. In addition to the errors already described, the maze had to be constructed in the hallway prior to the 75-minute lab and taken down immediately after. Students had to finish their programming during lab in order to demonstrate their robot's success, but this was often impossible with thirteen groups of students using the single maze in the hallway. A few times it was necessary to use a second class period to finish a lab. In spring 2010 the second author intends to move labs to evening hours and use all class periods for lectures and discussions.

- The robots were very helpful once one figured out how to use them. By using the robots for the projects, they gave good hands-on experience.
- Robots helped but took away from the focus/reason behind the assignments. The work that came with the robots presented more of a challenge than the material and theories behind it.
- The ZippyBot was a lot of work, but he was fun and it did help to visualize the techniques in question.
- The hands-on approach to learning AI with robots was very interesting and useful.

Figure 2: Selected student responses to course

Student reflections

Course evaluations were generally positive. Overwhelmingly, students felt that the robots added to the experience. Some of the student evaluations of the course can be seen in Figure 2. It is important to note that a number of students commented on the frustrations they felt. We realize that our approach is not problem-free. The fact that a "perfectly" programmed robot can fail to accomplish its task is not something that can be ignored. However, we feel that students need to confront such issues—very few projects in the post-college world are as clean as our in-class assignments. Things will go wrong even with code that is "proper" by some internal metric. We see our work with the robots as an opportunity to present this reality.

In the section of the spring 2009 evaluations that asks for suggestions for improvements, the first author saw comments that suggested more scaffolding was needed and that the groups might be better with fewer members. Interestingly, while one person advocated reducing the number of assignments, two others asked for more projects. Other suggestions for improving the course can be seen in Figure 3.

Faculty reflection

Both authors are happy with the outcomes so far. In particular, the students in the first author's course were more engaged than in prior (non-robotic) terms. Enthusiasm was high; this is significant given the increased level of frustration. Rather than being put off by this, students were willing to "work through" the frustrating elements due to the level of excitement created by working with the robots. Finally, the performance of the students was not significantly different from previous semesters; we are confident that the robots did not negatively impact student learning.

- Reduce to one project. Do massive term project encompassing problem solving, learning, and communication.
- Maybe partner groups (two people). I also think the robots may not be the best method because it is hard to program them.
- Give mini-checkpoints over the course of projects so students feel more compelled to work earlier and ask for help.

Figure 3: Student suggestions for improvements

In spring 2010, we plan several changes. As noted earlier, the second author will move the lab sessions to the evening to overcome some of the noted scheduling issues. This will enable better time management and allow her to get through all three assignments. The first author plans to offer a more formal set of lab sessions to provide additional scaffolding. Although the level of discomfort felt by the students cannot be completely eliminated, more formal lab sessions will lessen the students' uncertainty. Finally, we are intrigued by an idea offered by Kumar (2004) where groups may submit an unedited video of their robot "in action." This would greatly lower some of the frustrations felt by students who stated that their projects worked fine the night before.

Conclusion

Although we experienced problems with our approach in spring 2009, we are happy with our decision to use robots to teach core AI concepts. Students were enthusiastic and eager to begin each new assignment. Papers were well-written and demonstrated that the students had mastered the core ideas. Term projects were well-received and students were excited about their presentations. Both authors are looking forward to an improved implementation in spring 2010 and will report on its success at a future venue.

References

ACM/IEEE. 2008. Computer science curriculum 2008: An interim revision of CS 2001. `http://www.acm.org/education/curricula/ComputerScience2008.pdf`.

Arkin, R. C. 1989. Motor schema-based mobile robot navigation. *The International Journal of Robotics Research* 8(4):92–112.

Bagnall, B. 2007. *Maximum Lego NXT: Building Robots with Java Brains*. Variant Press.

Birnbaum, L., and Selfridge, M. 1981. Conceptual Analysis of Natural Language. In Schank, R. C., and Riesbeck, C. K., eds., *Inside Computer Understanding*. Hillsdale, NJ: Lawrence Erlbaum Associates. chapter 13.

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

Dodds, Z.; Greenwald, L.; Howard, A.; Tejada, S.; and Weinberg, J. 2006. Components, curriculm, and community: Robots and robotics in undergraduate ai education. *AI Magazine* 27(1):11–22.

Earley, J. 1970. An efficient context-free parsing algorithm. *Communications of the Association for Computing Machinery* 13(2):94–102.

Greenwald, L.; Artz, D.; Mehta, Y.; and Shirmonhammadi, B. 2006. Using educational robotics to motivate complete ai solutions. *AI Magazine* 27(1):83–95.

Heintz, F.; Kummeneje, J.; and Scerri, P. 2001. Using simulated RoboCup to teach AI in undergraduate education. *Frontiers in Artificial Intelligence and Applications* 66:13–21.

Imberman, S. P. 2003. Teaching neural networks using Lego-Handyboard robots in an artificial intelligence course. In *SIGCSE '03: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, 312–316. New York, NY, USA: ACM Press.

Kay, J. S. 2010. Robots in the classroom ... and the dorm room. *J. Comput. Small Coll.* 25(3):128–133.

Klassner, F. 2002. A case study of LEGO mindstorms'™suitability for artificial intelligence and robotics courses at the college level. In *SIGCSE '02: Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, 8–12. New York, NY, USA: ACM.

Kumar, A. N. 2001. Using robots in an undergraduate ai intelligence course: An experience report. In *31st ASEE/IEEE Frontiers in Education Conference*.

Kumar, A. N. 2004. Three years of using robots in an artificial intelligence course: lessons learned. *J. Educ. Resour. Comput.* 4(3):2.

LEGO Group. 2006. LEGO Mindstorms NXT hardware developer kit. `http://mindstorms.lego.com/Overview/nxtreme.aspx`.

Martin, S. 2009. Pep source and binaries. `http://www.ling.ohio-state.edu/~scott/`.

Nilsson, N. J. 1998. *Artificial Intelligence: A New Synthesis*. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Perdue, D. J. 2007. *The Unofficial Lego NXT Inventor's Guide*. San Francisco, CA, USA: No Starch Press.

Quinlan, J. R. 1987. Simplifying decision trees. *International Journal of Man-Machine Studies* 27:221–234.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Pearson Education, Inc., second edition.

Schank, R. C. 1975. *Conceptual Information Processing*. Amsterdam: North-Holland.

Talaga, P., and Oh, J. C. 2009. Combining AIMA and LEGO mindstorms in an artificial intelligence course to build real world robots. *J. Comput. Small Coll.* 24(3):56–64.