RoboCupJunior: a vehicle for enhancing technical literacy.

Elizabeth Sklar
Dept of Computer Science
Columbia University
1214 Amsterdam Avenue
New York, NY 10027 USA
sklar@cs.columbia.edu

Simon Parsons
Dept of Computer and Information Science
Brooklyn College
2900 Bedford Avenue
Brooklyn, NY 11210 USA
parsons@sci.brooklyn.cuny.edu

Abstract

This paper outlines the key components of *technical literacy*, explaining their relationship to classic computer science and artificial intelligence curriculum, and demonstrate how hands-on robotics can be an effective tool for teaching these topics. We describe an international educational initiative called *RoboCupJunior*, which was introduced in 1998 and has been growing rapidly in the last 4 years. We present research on the phenomenon of *educational team robotics* and explore the use of RoboCupJunior as a vehicle for promoting technical literacy.

Introduction

Advances in technology happen so quickly that new computer hardware and software becomes obsolete every 30 months (Gonzalez 2000). Comfort with one system only lasts so long before it is upgraded, and users must be retrained. So the key here is literally *literacy*. Memorizing a poem is no substitute for knowing how to read; literacy allows exploration of any new and unfamiliar text. Similarly, *technical literacy*, i.e., comfort with and understanding of technology, allows exploration of new and unfamiliar systems.

Computers, and various forms of automated machines such as robots, have infiltrated not only offices, schools and factories, but also homes and a wide range of everyday devices. While everyone in the next generations will not be engineers, everyone will need to assume a level of technical literacy well beyond that of their parents. "Technical literacy is quickly becoming as important as the ability to read." (Brussels 1998)

How can technical literacy be achieved? Education researchers and psychologists have repeatedly demonstrated the importance of hands-on experiences to promote learning (Gruber & Voneche 1977; Papert 1980). *Constructionist* theory states that we learn best when actively involved in building or constructing something, something that is external to ourselves, something that is personally meaningful (Papert 1991). Putting all the pieces together, the best way to learn about technology, to become comfortable with technology, to become technically literate, is to actually build something physical, something technical — *how about a robot?*

Once upon a time, you had to be an engineer to do robotics — a real engineer with an advanced degree in electrical and/or mechanical engineering and a big lab filled with computers, soldering irons, a drill press and a lathe. But today, you only have to be an eight-year-old child with a PC and a penchant for playing with LEGO. Just as the trend in computer hardware has evolved over the last 50 years from immense machines that took up entire rooms to powerful microprocessors that literally fit in the palm of your hand, robots have gotten smaller, faster and cheaper.

The result is that these little robot kits are showing up in classrooms all over the world. Creative instructors are finding ways to teach a myriad of science topics using these hands-on technologies and engaging kids of all ages. Tournaments are being organized around the robots, and the energy and enthusiasm displayed by participants is unsurpassed. Although the excitement is obvious, from an intellectual and pedagogical viewpoint, we wonder what the students are actually *learning* from working with robots.

Just because they are interacting with technology does not necessarily mean that they are learning something worthwhile. Yet this appeared to be the conventional wisdom as children began using computers in last two decades. Recently, researchers have questioned this stance. Jane Healy suggests that no more than 10% of the available software for children has any educational value (Healy 1998). Thomas Reeves reminds us that "fifty years of media and technology comparison studies have indicated no significant differences in most instances" (Reeves 1999). Tom Snyder warns that we are becoming "blinded by science" (Snyder 1994).

As a result, we and our colleagues have been examining the relationship between robotics and learning outcomes, attempting to formalize the notion of *educational robotics* — a term which we use to refer to the utilization of robots as a vehicle for teaching subjects other than specifically robotics. Currently, there is little or no work on integrating, testing and evaluating the use of robots as educational devices. Our work centers around fulfilling this need, and here we focus in particular on educational team robotics as a vehicle for enhancing technical literacy.

This paper begins by detailing some of the key components of technical literacy, explaining their relationship to

classic computer science and artificial intelligence curriculum and demonstrating how hands-on robotics can be an effective tool for teaching these topics. The second section describes an international educational initiative called *RoboCupJunior*, which was introduced in 1998 and has been growing rapidly in the last 3 years. We have been examining the phenomenon of *educational team robotics*, using RoboCupJunior as the basis for a 3-year study. From this work, we have identified several key elements that seem to make the initiative work, and we highlight these as factors in exploring the use of RoboCupJunior as a vehicle for promoting technical literacy. The third section outlines two undergraduate courses in robotics which are centered around RoboCupJunior activities and include results of course evaluations. The paper concludes with future directions and introduce the two newest RoboCupJunior projects.

Technical Literacy

We believe that technical literacy involves more than knowing about specific computer tools or material from a single scientific discipline. Technical literacy is about understanding topics like dynamical systems, state machines, search heuristics, knowledge representation, logic, uncertainty and planning. Obtaining a clear comprehension of these ideas has global importance today, in order to be able to take full advantage of the types of devices that are becoming commonplace — such as palm-top computers, mobile phones and VCRs. As well, technical literacy is important not only for becoming comfortable with technology itself, but also for accomplishing a variety of non-technical tasks where application of scientific method can offer improved process and results.

Here, we expand on our notion of technical literacy by enumerating some of these topics:

- **Dynamical systems.** The world is a dynamical place, which means that things change over time. The field of system dynamics deals exactly with this phenomenon and has been applied to K-12 education, bringing topics like feedback loops and computer modeling to bear in teaching children about systems (Forrester 1992). How does one construct a plan to accomplish a certain task? Once the plan is made, how can it be adjusted if unexpected changes in the environment cause the plan to fail? What if in the middle of executing the plan, external influences cause the task itself to be altered? These types of scenarios occur regularly in everyday life, and system dynamics offers methodology for coping with them effectively.
- **Markov processes.** A state machine provides a simple and general language for describing a process in which discrete changes to a situation take place over time. A Markov model is used to describe situations where more than one change could occur in a given state and probabilities are attached to each possible change. Students need the ability to analyze a problem and describe a solution as a series of steps in which cause and effect play a key role. The concept of a Markov chain can be a powerful modeling tool for describing not only the steps required for programming a robot, but also for example alternative routes for driving under various traffic conditions or multiple lines of argument in a legal debate.
- **Search heuristics.** What is a good technique for searching for something? Methodologies and heuristics from AI provide solid algorithms for conducting any kind of search. For example, a paralegal seeking a reference for a particular case could perform either breadth-first or depth-first search. Given the type of citation being sought, the available references and the cost of obtaining additional references, the appropriate and more efficient search methodology can be selected.
- **Knowledge representation.** Learning how to represent real-world data so that it can be stored and analyzed in a computer is an important skill. Children are being introduced to tables and spreadsheets during primary school, which are useful for representing simple relational data. However, complex relational and hierarchical data structures are also powerful concepts and allow representation of sophisticated data that does not lend itself to spreadsheets.
- **Logic.** Understanding logical operators is becoming more and more important. When performing a search on the Internet, what does it mean to search for "apple orange pear" — is this a logical conjunction or disjunction? Are you interested in articles containing the words "apple" and "orange" and "pear" or are you interested in articles containing either the word "apple" or the word "orange" or the word "pear"? Logical reasoning is also important— for example, when a government declares that "there is no evidence that beef is unsafe", this is not the same stating that there is evidence that beef is safe, and understanding the difference between these two statements can, literally, be a life and dead matter.
- **Uncertainty.** Just as the world is a dynamic place and we are often faced with making decisions in changing and unpredictable environments, we are also confronted with the need to make choices with incomplete information. Reasoning under uncertainty involves designing networks of probabilities, attempting to model the known elements in an environment and identify the unknown elements, drawing conclusions which may be modified as more information becomes available. Programming a robot to cross a busy street where there is no traffic light is much the same problem posed for a human in the same situation.
- **Planning.** Looking ahead and deciding what to do next, reasoning in an uncertain and dynamic world, modeling the environment, making decisions — all of these factor into planning. The notions of establishing contingency plans, of determining a heuristic to choose between these plans, of evaluating a plan after it has executed, can be useful in many applications.

Thus we have illustrated through examples that the concept of technical literacy is extremely powerful, both inside a computer science classroom and outside in the everyday world. The next section describes the educational team robotics initiative, RoboCupJunior, and details the types of activities encompassed by the program. This leads us to a

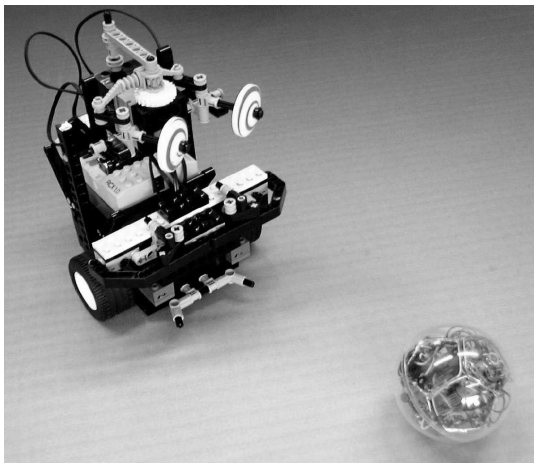


Figure 1: The first RCJ footballer.

discussion that ties the elements of technical literacy specifically to RoboCupJunior, documenting two undergraduate courses which integrate hands-on robotics and RoboCupJunior challenges.

RoboCupJunior

RoboCupJunior (RCJ) began as a research project, exploring the idea of robotic soccer using the LEGO Mindstorms platform (Lund & Pagliarini 2000) (see figure 1). The focus was on using robotics to teach young students about the field of Artificial Life (or ALife).

RoboCup-2000 in Melbourne, Australia marked the first international RoboCupJunior tournament (Sklar, Johnson, & Lund 2000), and took place in conjunction with the RoboCup tournament[1] as all subsequent international RoboCupJunior tournaments have. Over 100 students participated, ranging in age from 8 to 19 and coming from 25 schools around Australia, as well as from Germany and the USA. The blueprint for RoboCupJunior tournaments was developed, involving a curriculum-based, student-driven approach.

The second international tournament took place at RoboCup-2001 in Seattle, USA (Sklar, Eguchi, & Johnson 2002). Approximately 100 students, ages 7 to 23, from the area surrounding Seattle, as well as other American states, England, Germany and Australia participated.

In 2002, the third international RoboCupJunior tournament was held in Fukuoka, Japan. The initiative has exploded in popularity. For the first time, the event attracted teams from a wide geographical region (see table 1). Most teams were selected from regional or national tournaments. In Australia, a full national competition was held, with 5 states conducting eliminations leading to a national final. In all, over 200 teams and 500 students participated.

[1] www.robocup.org

Table 1: Countries represented at RoboCupJunior 2002. Fifty-nine teams from twelve countries came to RCJ-2002, totalling over 240 mentors and students.

country	number of teams		country	number of teams
Australia	8		Korea	5
Canada	1		Macao	2
Denmark	1		Norway	1
Finland	1		Slovakia	1
Germany	5		Thailand	4
Japan	29		USA	1

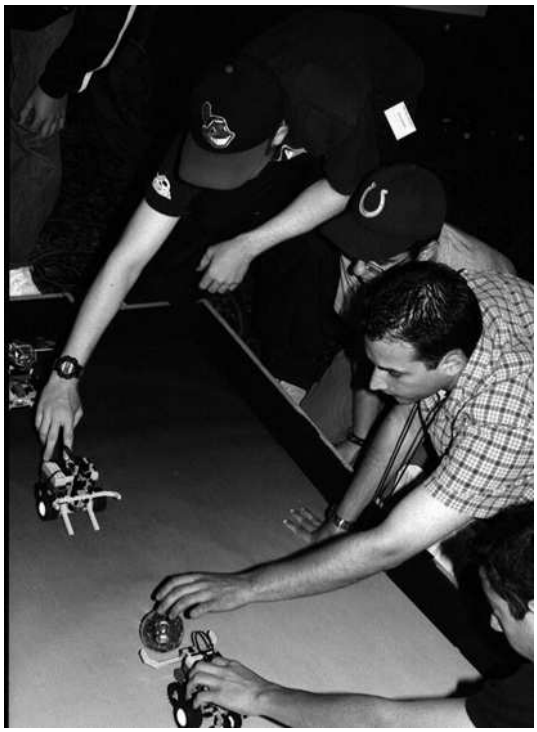


Figure 2: The RCJ soccer.

Challenges.

Three challenges have been developed for RoboCupJunior: *soccer*, *dance* and *line-following*.

In the **soccer** challenge, two teams of two robots each play soccer on a 4 foot by 6 foot field. The floor of the field is lined with a greyscale mat and the ball is an electronic device that emits infra-red (IR) light which makes it relatively easy to see (as shown in figure 2). The field and ball specifications were originally developed in Lund's lab in 1998. The rules of play are based on FIFA soccer rules and were adapted to robotic soccer following from the RoboCup F-180 (small) league rules. The tournaments use a round-robin elimination, where the teams are divided into groups and each team plays at least 4-5 games (depending on the number of participants). Final rounds succeed the round-robin.

Figure 3: Robot dance teams.

A one-by-one soccer game was also developed (indeed, this was the original instantiation of the game), but this version has been less popular, because the game play is less exciting. However, this model does give teams with only one robot the chance to participate. As an exhibition at RoboCupJunior 2002, a new *friendship game* was introduced. Teams were paired, each team supplying one robot, and the pairs participated in two-on-two games. In this way, teams that brought either one or two robots were able not only to experience the added complexity of the two-on-two game, but also to interact with other teams in a shared project.

For the **dance** challenge, students build robots that move to music for up to two minutes. Creativity is emphasized — robots (and sometimes even students) are dressed in costume. This is designed to be an entry-level event, since it is possible to participate using simple robots that only employ motors and no sensors. The event itself is exciting and innovative.

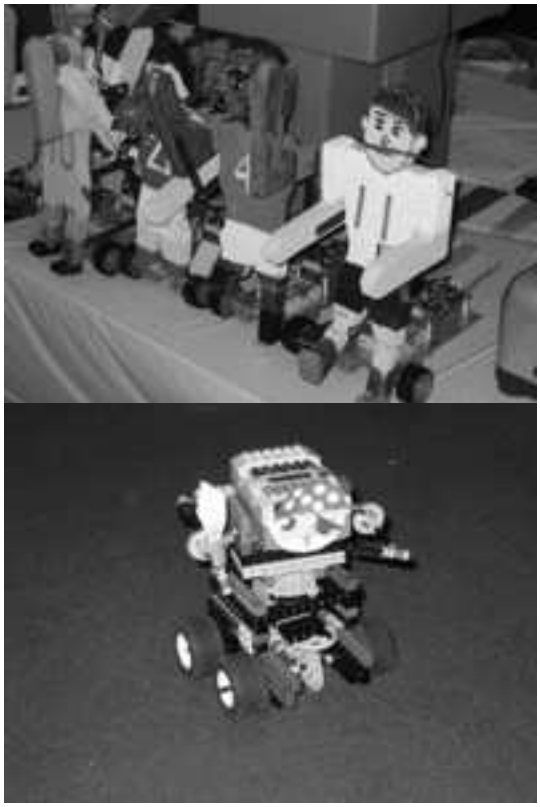


Figure 4: RCJ Dance 2002.

In 2002, the dance event showed tremendous progress. Twelve teams participated, each demonstrating unique and creative ways of combining technology with art and music. Some teams' routines told stories. Many teams shared their country's culture through traditional dances, music and costumes — worn by both robots and students. Several teams built robots out of wood, like puppets, dressed and decorated for the occasion.

The **line-following** challenge has undergone the most change. At RCJ-2000, it was a *sumo* event, designed as a middle-level challenge (see figure 5). Two robots followed wiggly black lines and competed for possession of a central circular region on the playing field. This was presented as a middle-level event; only one robot was needed for each team and the environment was essentially static. The only dynamic elements were the two robots; they had limited interaction and did not need to respond to each other, only to changes in their own location on the playing field.

In 2001, sumo was replaced by a rescue event. The **rescue** challenge is timed and one robot competes at a time. The background of the field is white, and the robot is required to follow a black line through a simulated disaster scenario. The scenario designed for 2001 was that of a burning building, where the robot was supposed to rescue three victims stranded on the roof. The black line turned in a maze-like

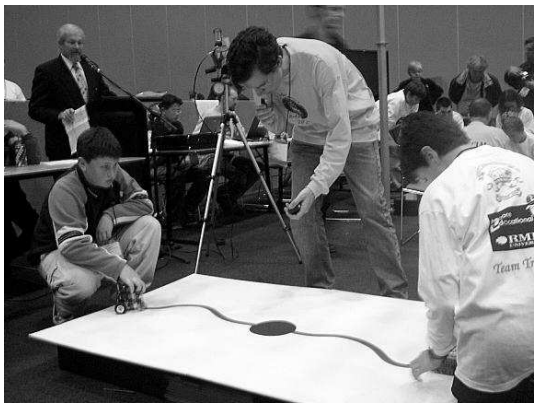


Figure 5: RCJ-2000 sumo.

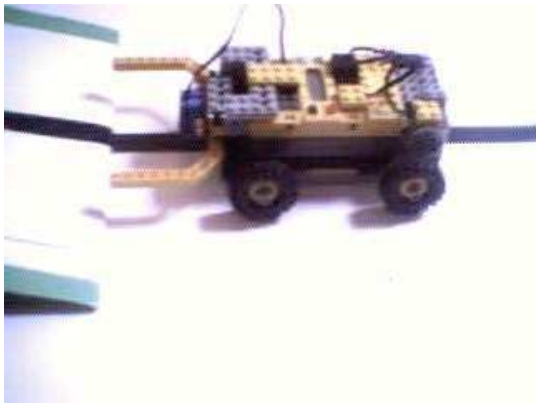


Figure 6: RCJ Rescue, 2001.

pattern, then went up a ramp to the roof, where the robot was supposed to push three toy people off the roof to a safety net below. While there were no dynamic elements, accurate control of the robot based on light sensor readings is essential and proved to be surprisingly difficult. As well, the uneven terrain with a change in pitch where the robot reached a ramp and a bump in the path between the flat field and the ramp made the course even more challenging

There was no line-following competition in RCJ-2002, but as discussed later in the paper, we are developing a new line-following event which will be piloted at RCJ-2003.

Studies.

RoboCupJunior gives us the opportunity to study the effects when groups of students participate in educational team robotics activities. We have conducted studies at RoboCupJunior in 2000 (Sklar, Johnson, & Lund 2000), 2001 (Sklar, Eguchi, & Johnson 2002) and 2002. Here we highlight the results.

At RCJ-2000, we interviewed about half of the teachers who entered teams, with the general stated goal of investigating the educational value of RoboCupJunior. This study revealed remarkable consensus of opinion amongst the teachers:

- RoboCupJunior fits in with existing robotics curriculum.
- RCJ is highly motivating for participants.
- The initiative advances both academic and personal development skills.
- RoboCupJunior teaches teamwork and tolerance of others.
- RCJ appears to attract girls into robotics as well as boys.

The RoboCupJunior competition itself is a motivating factor, particularly because:

- it is an international event,
- it imposes an absolute deadline (i.e., the date of the conference is fixed), and
- it gives young students an entry-level role in the complex and stimulating field of robotics research in an exciting context — alongside the senior RoboCup competitors, some of the top robotic scientists and engineers in world.

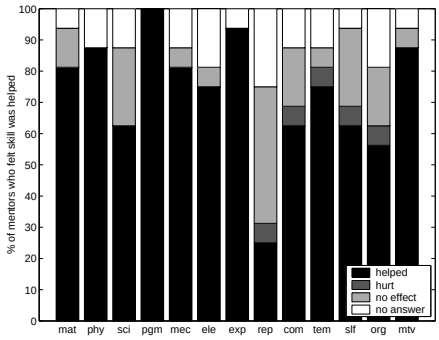
At RoboCupJunior 2001, mentors as well as students were interviewed. They were asked to consider 13 specific skills and indicate whether they felt their involvement in RoboCupJunior had helped or hurt each of these skills, or if there was no effect (see figure 7). The selection of the specific skills listed was based on the results of the study conducted in the previous year (Sklar, Johnson, & Lund 2000). The overall consensus is that all the skills named were helped more than they were hurt. Note that participants felt that reporting skills were helped less than other skills. This could be due to the lack of activities such as keeping journals and writing lab reports. Further emphasis on reporting as part of the tournament itself (i.e., posters and papers) will help promote development of this skill set.

It is interesting to compare the mentors' and students' skill ratings. Overall, more of the mentors consider that RoboCupJunior has positive effects than the students. It is likely more difficult for students to assess the effects objectively than it is for mentors. Also, it is harder for students to assess abstract skills, such as communication, self-esteem and organization, than it is for them to evaluate concrete skills, such as mathematics, physics and programming. Future studies will investigate more effective ways of asking students about abstract skills.

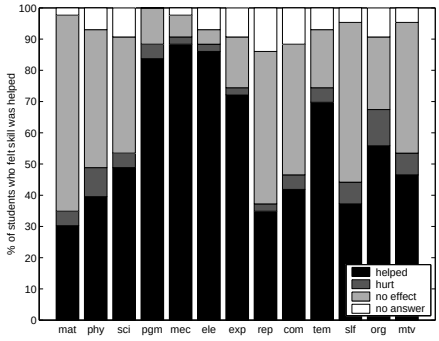
At RCJ-2002, again we studied mentors and students using both paper-and-pencil surveys as well as video-taped interviews (of students only). 57% of teams completed the surveys, totalling 104 responses. Here we share preliminary results from the students' surveys.

In this survey, students were presented with multiple questions about each of the thirteen skills examined in the previous study. They were asked, on a skill-by-skill basis, did RoboCupJunior and/or your robotics experience have a positive effect? The results are shown in figure 8. Note that the key is the same as in figure 7, with the addition of "pdv" for personal development, grouping questions regarding self-esteem and motivation.

We highlight a couple of interesting factors:



(a) mentors



(b) students

key:

mat	math
phy	physics
sci	general science
pgm	computer programming
mec	mechanical engineering
ele	electronics
exp	experimental skills
rep	reporting
com	communication
tem	teamwork
slf	self-esteem
org	organization
mtv	motivation

Figure 7: Effects on various skills (2001).
The bars illustrate the number of participants who indicated whether each skill was helped, hurt, etc. For example, 80% of the mentors indicated that they thought their students' math skills were helped through their preparation for RoboCupJunior; approximately 12% of the mentors indicated that they thought that the preparation had no effect on their students' math skills; and 8% did not respond to the question.

- 60% of the students indicated that they did not feel their math skills were affected by their participation in RoboCupJunior or their experience with robotics.
- There is a disappointing relationship between students' attention to the survey and their performance in the competition.

The main differences found in the three studies were related to technical implementation. In 2000, all teams used the LEGO Mindstorms platform. In 2001, other platforms were used: Fischer-Technik mobile robot (16% of teams), Tetrixx kit (4%), and Mindstorms (80%). In 2002, the Elekit SoccerRobo was added to the list. Also, some teams built robots themselves from basic components. Figure 9 illustrates some of the variations.

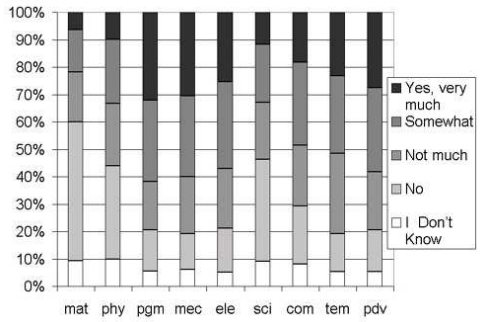


Figure 8: Effects on skills, 2002.

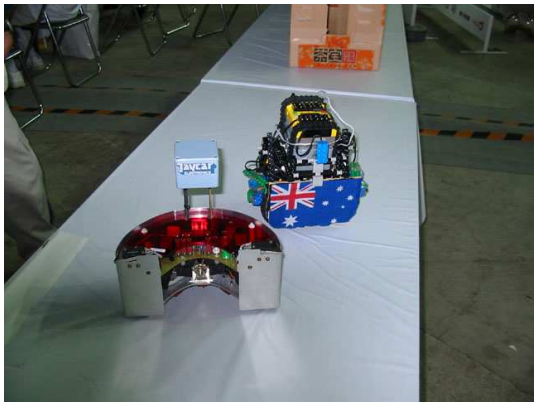


Figure 9: A variety of RCJ platforms.

The trends in motivational and developmental aspects were markedly similar from one year to the next. Indeed, the motivational aspects found in educational technologies in general (**?**; **?**; **?**; **?**) are clearly carried into educational team robotics activities. Most teams spent more than two hours for each preparation meeting, which suggests that robotics activities are challenging and attractive enough to make students focus on their work for long periods of time. The emphasis on teamwork in RoboCupJunior allows students with a variety of interests and abilities an opportunity to pick their own challenges while contributing to the progress of the whole, an experience that nurtures the varied and multiple intelligences (**?**) of each participant.

Robotics in undergraduate education.

Our goal is to take advantage of the motivational aspects of RoboCupJunior by bringing the activities into undergraduate classrooms. While undergraduates are not (currently) eligible to enter the competition itself, the three challenges can be used to demonstrate concepts taught in many computer science classes. As an example, we describe two such classes here where we have successfully used RCJ challenges as term projects. Then, we broaden the scope into general technical literacy and discuss two additional courses in which

robotics and these challenges will be used in the near future.

Artificial Intelligence.

This course is designed to give a broad understanding of the basic techniques in use today for building intelligent computer systems. Students learn about state-space representations, problem reduction, means-end analysis, and and-or graphs. They study search methods including depth-first, breadth-first and best-first search, as well as hill-climbing, divide and conquer, minimax and alpha-beta pruning. Predicate calculus is introduced, along with various methods of theorem proving. AI systems and languages, goals and contexts, are presented. Issues of knowledge representation, machine learning and concept formation are discussed.

The modern view of Artificial Intelligence (AI) (Russell & Norvig 1995) is that it is the study of intelligent agents — autonomous computing systems that perceive their environments and act upon them in a way that both responds to changes in the their environments and works towards underlying goals. The metaphor of intelligent agents is a way of bringing together the many strands of work carried out under the banner of AI and presenting them to students in a convincing way. For example, thinking of an agent exploring an environment is a natural way to introduce search techniques, and considering how agents must respond to changes in their environments clearly shows the advantage of behavioral-based reactive techniques.

Robots are prototypical agents that have to move around, and react to, their environments in pursuit of their goals. Indeed, it is hard to think of something that embodies the qualities that are required of intelligent agents better than robots do. As a result, it is highly appropriate to explore areas of a typical artificial intelligence syllabus using robotics projects. Some examples of the way in which robotic projects can be used are:

- **Behavior-based techniques:** In behavior-based approaches, agents follow simple stimulus-response rules, and complex behaviours can be constructed from them. Building a hierachy of responses to sensor input, robots built from the LEGO Mindstorms kit can navigate through complex environments as illustrated by student projects in a recent offering of the course (Parsons 2002).
- **Search techniques:** Search techniques are at the heart of traditional Artificial Intelligence. The large array of techniques can easily be explored using robots, which have to, for instance, navigate through a maze (whether delineated by walls or by a line). Successful use of LEGO robots in such student activities is reported by (Kumar 2001).
- **Learning techniques:** Learning is a major sub-field of Artificial Intelligence and of great importance to intelligent agents — intelligent machines will learn from their mistakes. Although learning techniques such as neural networks and reinforcement learning are too complex to implement directly on the LEGO hardware, they may be implemented on the simulator we are developing It is therefore possible to have projects in which a robot controller is learnt on a simulator and then downloaded into the LEGO robot.

From our preliminary experience in using robotics as a device to teach an undergraduate Artificial Intelligence course at Columbia University, it seems that the chance to program "real" agents is appreciated by students. Students were given a term project in which they had to program LEGO robots to perform a RoboCupJunior style line-following rescue task and to play a simplified version of soccer. Students were also given the option of an extra-credit project of building a dancing robot. When we offered this course in Spring 2002, two course evaluations were administered. One was an official evaluation done by the engineering school. The other was an informal paper-and-pencil survey given out in class.

The results of the engineering school's evaluation showed that 55% of 33 students gave the robotics project they undertook a rating of 5 (on a 5-point scale) for interest, and two-thirds gave it a rating of 4 or 5. 21% of the same cohort of students gave the project a rating of 5 for the amount learned during the project, and 58% gave it a rating of 4 or 5.

Introduction to Robotics.

This course looks at robotics from several aspects: technically, historically and socially. The course is designed for non-engineering students to gain a basic understanding of the field of robotics and the challenges facing the field today. Part of the course is spent reading and discussing classic material that relates to robots — including science fiction, psychology, cognitive science and education. The remainder of course takes a hands-on approach to introducing the basic concepts in robotics, focusing on autonomous mobile robots. LEGO Mindstorms robots are used, and students must complete two projects with them. First, they must build robots to execute a line-following task culminating in a maze contest. Second, they construct robots to play soccer and perform in a RoboCupJunior style two-on-two tournament.

The topics covered include the basics of building and programming with LEGO Mindstorms, using the Not Quite C (NQC) programming language (Baum 2000). The software (which is essentially a compiler) is freely available on the Internet[2] (Baum 2000). The robots are used as examples for the remainder of the topics, which introduce the general areas in robotics: effectors, sensors and control. The area of control is covered in more depth, discussing deliberative, reactive, hybrid and behavior-based architectures. Learning is also discussed.

Students were surveyed at the end of the course. 83% responded that the labs (i.e., building and programming the robots) was helpful for learning the material, whereas only 33% said that the reading was helpful. 75% and 67% responded that the two contests (maze and soccer, respectively) were valuable in helping them solidify and demonstrate their knowledge of the material.

The high school and younger students who enter RoboCupJunior do not participate as part of a computer science class. Yet the lessons learned by interacting with the robots are similar, and our contention is that more general technical literacy topics can be taught effectively through the

[2] `http://www.enteract.com/~dbaum/nqc`

initiative. We illustrate this notion by outlining two courses, one in programming and one in technology.

Introduction to Computer Programming and Computer Science.

This course provides an introduction to the field of computer science as a science of abstraction. Students learn about the basic elements of computers and computer programs. They learn how to write effective computer programs (in C, C++ or Java) and how to create models for reasoning about and solving problems. They learn about implementing abstractions using simple data structures and algorithms.

The first computer programming course students take is often extremely daunting. In college, students are used to the lecture style of a class and are often loath to put their fingers on the keyboard and really work out a homework assignment. But in order to be good programmers, students need to learn to *tinker*. The addition of robotics brings out the natural "tinkerer" — students tinkered with LEGO as children, so it becomes easy to extend this mentality from physical robot-building to on-line writing of programs.

There are several programming interfaces available for use with the LEGO Mindstorms platform. NQC provides a C-like interface to the Mindstorms microprocessor and so is one appropriate choice for use in an introductory course in C. There are also Java interfaces to the Mindstorms.

For new programmers, the material being acquired is challenging. Some concepts can be very difficult to grasp and so being able to actually see the result of small changes in software on a real, physical (or simulated) robot can be extremely effective. Here are some examples.

- **Data storage and representation:** The notion of storing data in a computer is abstract, but knowing how data is stored is important in order to be able to design effective classes in object-oriented programming and to manipulate data correctly and efficiently. The small robot kit has relatively little memory and students will need to learn how data is stored in order to make full use of the robot's memory.
- **Branching:** Branching refers to a decision point where there are several options for what to do next. Students have trouble understanding the syntax and semantics of branching statements. With a robot, they can really see how a small change in syntax will modify the semantics of what the robot does.
- **Looping:** Looping refers to doing something multiple times, either a fixed number of times or until a certain condition becomes true. Comprehending these notions is difficult for a new programmer, but a robot can easily demonstrate the difference between "move forward for 5 seconds" versus "move forward until you hit a wall".
- **Modularization:** Some students may be familiar with creating macros — saving frequently-used groups of statements that can be invoked via a single user-defined command. But modularization goes beyond that, particularly in the design phase of coding where students need to learn to think about their program before writing any code. Being able to structure their program in modules is a first step toward understanding the object-oriented paradigm, the backbone of Java and C++. In behavior-based robotics, modules are created to perform simple behaviors such as "go to the light". Through this paradigm, students have an intuitive environment for defining behaviors and thus experiencing their first attempt at designing object-oriented software.
- **Exception handling:** The notion of an interrupt driven interface is quite an advanced concept, but one that must be understood by students who are learning Java. Having a robot makes this a natural lesson. For example, consider obstacle avoidance, one of the classic robot behaviors. The algorithm basically says: "go forward; but if you hit an obstacle, go around it."

Introduction to Technology.

This course provides a very general introduction to technology, designed for non-technical students. Topics include mathematical reasoning, problem solving, the design of algorithms and computer hardware, as well as hands-on experience with applications such as spreadsheets, databases, and the World Wide Web. The use of robots as a physical, hands-on medium can be very effective not only for helping students understand these concepts but also for engaging students who are typically disengaged from these types of concepts.

Many of the modules for this course are adapted from the expansion of modules in the *Introduction to Computer Programming and Computer Science* course described above, but here focus more on concepts at a higher-level than the programming course, which also concentrates on the specific syntax and semantics of a particular programming language. As well, these modules will be structured to help students connect the new concepts with their everyday lives and their everyday experiences with technology. Some examples are:

- **Programming concepts.** There are some basic concepts that occur in almost all programming environments, for example branching and looping. Understanding these concepts and the relationship between the two can help you program a VCR or set the time on a digital watch. Concepts like "point-and-click" and "select-and-set" fall into this category. In both cases, you perform a repeated action (pointing or selecting) and the device being programmed waits until you perform a decision action (clicking or setting).
- **Fundamental algorithms.** Generally defined as "computational procedures", algorithms are used for programming computers to achieve a variety of tasks. It has been realized that many seemingly unrelated tasks may be successfully accomplished by following similar computational procedures. As well, one task may be completed using different algorithms, each having different strengths; and an appropriate algorithm may be chosen based on desired performance metrics. Knowledge of these types of methodologies can be helpful for simple daily tasks like filing and balancing a checkbook.

- **Logical thinking.** As discussed above, the ability to both understand logical statements and to reason about what follows from them is an important part of technical literacy. These are also skills that are practiced when programming robots—there are clear differences between the behaviour of a robot that stops when it detects light AND bumps into an object, and one that stops when it detects light OR bumps into an object. Similarly, it is clear that just because a robot cannot detect a bend in the line that it is following (there is no evidence of a bend) it does not mean that there is no bend.

These, then, are the undergraduate programs that we anticipate using as part of our continuing investigation into the use of robotics to promote technical literacy. We are also extending the reach of the RoboCupJunior activities which, we hope, will give further opportunities to use robotics in this way, both for the 14-17 year age group targeted by RoboCupJunior to date and undergraduates.

Current and future work.

There are two aspects to our extension of RoboCupJunior, the U-league and RoboCupJunior Rescue.

The U-League

As things stand, the structure of RoboCupJunior is intended to give school children a series of more complex challenges. The dance competition is intended to be an entry level event with no need for the use of sensors. The rescue event is intended to be an intermediate level event with the use of sensors in a static environment. The soccer events are intended to be more complex, involving the use of sensors in a dynamic environment. While this progression seems, as far as anecdotal evidence is concerned, to work well (although some competitors at least feel that dance is by far the most creative of the challenges), we feel there is problem with it as it stands.

The problem is that some of the most involved competitors, who have competed in the soccer events for several years in a row, find themselves falling through a gap in the challenges. Some find themselves ruled out of the competition because they are leaving high school to go to college. Others feel that the soccer challenge is no longer sufficiently interesting due to the limitations imposed by the sensors on the robot kits and the number of robots on a team.. Either way, RoboCupJunior is losing some of its best competitors and given that one of the aims of RoboCupJunior is to encourage the next generation of robotics researchers, this is a grave loss.

It is possible, of course, that these "graduates" of RoboCupJunior might step up to the senior tournament, and indeed some have said that this is something they would like to do. However, it is a big step up from programming two LEGO Mindstorms to the nearest alternative in the senior league—the small size robots—where the expense of the hardware and the engineering expertise required to develop it prevents many universities taking part.

As a result, we are working on a new RoboCupJunior challenge, one that is within the reach of both high schools and universities without the budget to compete in the small size league. This challenge is currently known as the U-league.

Although the details of the U-league remain to be worked out, the basic form of the competition is decided, and is as follows[3]. Teams will feature 4 robots of a standard type and roughly the same dimensions as the small-size league robots. The pitch will be slightly smaller than that used by the small-sized league in 2002 (where teams were composed of 5 robots). Teams will use standard vision software (the RoboCupVideo Server[4] developed by Jacky Baltes) fed from a common camera, and commands will be fed to the robots by a communication mechanism, based on infra-red broadcast, that is common to both teams in a game.

The reason for these choices is to allow competitors to concentrate on developing good soccer playing programs without having to concentrate too much on other issues. The choice of pitch size and number of robots enriches the environment to make it more interesting without making the cost of acquiring a team too high. The use of a common video feed is intended to simplify the task facing the teams—they don't need to develop their own vision system, and teams won't suffer from not getting the prime position for their video camera (as seems to happen in the small-size league). Finally, the use of a common communication platform again simplifies the development task, and the use of infra-red (which can be shielded by thin partitions) overcomes the problem of radio interference between teams which is also a problem in the small-size league.

RoboCupJunior Rescue

The RoboCup Rescue competition[5] promotes the development of search and rescue robot teams that can respond to disasters by locating human victims and thus aiding traditional emergency services. The arena for the competition is a sequence of three "rooms", developed by NIST (the US National Institute for Standards and Technology), which contain a variety of terrain and a number of simulated victims. The rooms are colour-coded from yellow through orange to red, becoming progressively more difficult to navigate, and having simulated victims that are progressively more difficult to find. The great advantage of the NIST arena is that is provides a standard modular enviornmnet for Rescue competitions. Our current aim is to develop a similar modular environment for a RoboCupJunior Rescue challenge.

Although, as is the case for the U-League, the details of the challenge are yet to be finalised, the outline is decided[6]. The challenge will be in part a line-following exercise, and the robots will have to traverse different types of terrain. This will include different surfaces (such as wood, linoleum and carpet), and different gradients, in particular gradients

[3] For the most current description of the U-League see `http://www.robocupjunior.org`.

[4] `http://sourceforge.net/projects/robocup-video`

[5]

[6] For the most current description of RoboCupJunior Rescue see `http://www.robocupjunior.org`.

that connect different levels of the arena. Robots will also have to recognise "victims", and these will advertise their presence using infra-red light (just as the simulated victims in the main Rescue competition are warmer than their surroundings). The arena will also be modular, and participants will be encouraged to bring their own practice field to competitions where they will be combined to make up a large competition arena.

Summary

This paper has described some of the varied aspects of the RoboCupJunior tournament and, in particular, its role as a vehicle for the development of technical literacy. In addition to explaining what we mean by the term "technical literacy" and describing some of the history of RoboCupJunior, we have focussed on three main issues.

The first of these is our work to identify and quantify the impact of project-based robotics work on the technical literacy of RoboCupJunior participants. We presented results obtained from the analysis of a survey of participants at RoboCupJunior in 2000 and 2001 (described in more detail in (Sklar, Eguchi, & Johnson 2002)), and the preliminary results obtained from a similar survey carried out at RoboCupJunior in 2002.

The second issue that we discussed was that of using project-based robotics in undergraduate curriculum. To illustrate this issue we described four courses, which we have either already taught or plan to teach in the near future, that make use of robotics to develop aspects of technical literacy.

Finally, we described our ongoing work to broaden the appeal of RoboCupJunior. This involves the development of two new challenges—the U-League soccer challenge, which is intended to fill the gap between the current RoboCupJunior soccer challenge and the small-size league, and RoboCupJunior Rescue, which is intended to offer the same kind of challenge to Junior participants that RoboCup Rescue offers to participants in the main tournament.

References

Baum, D. 2000. *Dave Baum's Definitive Guide to LEGO Mindstorms*. APress.

Brussels. 1998. A call to action to bridge the it skills gap: The need for public/private partnership. In *Summit on employment and training in the information society*.

Forrester, J. 1992. System dynamics and learner-centered-learning in kindergarten through 12th grade education. Technical Report D-4337, MIT.

Gonzalez, A. 2000. Digital divide closes — but schools aren't ready. *USA Today* April 26.

Gruber, H. E., and Voneche, J. J., eds. 1977. *The Essential Piaget*. BasicBooks.

Healy, J. 1998. *Failure to Connect: How Computers Affect Our Children's Minds — and What We Can Do About It*. New York: Simon & Schuster.

Kumar, A. 2001. Using Robots in an Undergraduate Artificial Intelligence Course: An Experience Report. In *Proceedings of the 31st ASEE/IEEE Frontiers in Education Conference*.

Lund, H. H., and Pagliarini, L. 2000. Robocup jr. with lego mindstorms. In *Proceedings of ICRA2000*. New Jersey: IEEE Press.

Papert, S. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. BasicBooks.

Papert, S. 1991. Situating constructionism. *Constructionism*.

Parsons, S. 2002. Artificial intelligence, coms 4701. `http://www.cs.columbia.edu/~sp/4701-2.html`.

Reeves, T. 1999. A research agenda for interactive learning in the new millenium. In *Proceedings of the World Conference on Educational Multimedia, Hypermedia & Telecommunications (EdMedia99)*.

Russell, S., and Norvig, P. 1995. *Artificial intelligence : a modern approach*. Englewood Cliffs, N.J.: Prentice Hall.

Sklar, E.; Eguchi, A.; and Johnson, J. 2002. Robocupjunior: learning with educational robotics. In *Proceedings of RoboCup-2002: Robot Soccer World Cup VI*.

Sklar, E.; Johnson, J.; and Lund, H. 2000. Children learning from team robotics: Robocup junior 2000 educational research report. Technical report, The Open University, Milton Keynes, UK.

Snyder, T. 1994. Blinded by science. *The Executive Educator* March.