Educational Robotics in Algorithm Concretization

Javier LÓPEZ
Department of Computer Science, University of Joensuu, PO BOX 111 FIN-80101 Joensuu Finland, jgonza@cs.joensuu.fi, http://cs.joensuu.fi/edtech

Niko MYLLER
Department of Computer Science, University of Joensuu, PO BOX 111 FIN-80101 Joensuu Finland, nmyller@cs.joensuu.fi, http://cs.joensuu.fi/edtech

Erkki SUTINEN
Department of Computer Science, University of Joensuu, PO BOX 111 FIN-80101 Joensuu Finland, sutinen@cs.joensuu.fi, http://cs.joensuu.fi/edtech

KEYWORDS: *algorithms, concretization, robotics, sorting*

ABSTRACT: *Algorithm concretization with robots combines hands-on robotics and traditional algorithm visualization techniques to assist diverse learners to understand the basic idea of a given algorithm. In this way, we bring algorithms into the real physical world where students can touch the data structures during the execution. We present a proof-of-concept implementation with a few sorting algorithms. Moreover, we have carried out an evaluation with 13-to-15-year old children who used the concretization to comprehend an algorithm. The results indicate that algorithm concretization can enhance their learning process. Our implementation experience and the analysis of the experiment give new directions to the further development for both the educational robotics and the learning environment. Our aim is to build a new learning environment that helps students to understand algorithms and to acquire programming skills through concretization with robotics.*

1 INTRODUCTION (ROBOTICS AND ITS APPLICATIONS. HISTORICAL VIEW)

Usually in the first year of Computer Science studies, one of the main difficulties for the students is the understanding of algorithms. The conventional approach of teaching algorithms, inherited from former education, consists mainly of verbal explanations along with the use of blackboard and / or slides, a rather "static" point of view in contrast to the idea embodied in algorithm: how data (input) is transformed into outputs over time. Thus it seems natural to show how they evolve with film-like tools [BAECKER, R. M. 1983]. This idea is based on what is called *algorithm visualization* whose aim is, apart from another description of algorithm's activity, to seek interaction with the students. [BEN-ARI, M., MYLLER, N., SUTINEN, E., & TARHIO, J. 2002] [FLEISCHER, R., & KUČERA, L. 2002]. Nevertheless, most of the visualizations available are constrained within computer screens, and it is difficult to find a suitable balance between animation and interaction [FALTIN, N. 2002].

Nowadays, with the dropping of robot prices, and specifically with Lego™ Mindstorms Robots release, robotics has been adopted in the teaching of Computer Science. Robots provide a sort of fascination to people [NARAYAN, S., TOMPKINS, J. 2003] which increases their level of engagement in the learning of programming, networking, distributed systems, etc. and a novel viewpoint for teaching. However, there have been negative results [FAGIN, B., MERKLE, L. 2002]: the usual approach in a introductory programming course is the teaching of the basic *concepts* of programming tied to a programming language. But robots generally use their own programming language with its own features. That leads to a situation where the specific programming language constraints the general programming language for the instruction of programming concepts. Besides, there is a lack of experience in teaching a programming course with robotics and it is complicated to integrate this new experience into the traditional learning environment.

Our approach to the robots is somewhat different: We think that robots can be used to *concretize* algorithms; their behavior is emulated by using robotics or other real world objects. In addition, concretization makes easier to link the abstract knowledge of the usual Computer Science teaching to everyday issues so that the students can apply a whole set of already known resources (i.e. problem solving skills) to compose new knowledge and hence to foster their learning [RESNICK, M., BERG, R., EISENBERG, M. 2000]. In this way, concretization can be easily adapted to different learners and cultural contexts. When students design a concretization of an algorithm they need to analyze it more carefully in order to express the key features of the algorithm, but they do not have to focus on implementation details. While robots move in the real world representing the algorithm, students get immediate *visual* or sensory feedback and gain insight into different features of the algorithm, such as its efficiency. In this fashion, students can compare among different algorithms and enlarge the assimilation of them into a higher maturity level [LAWHEAD, P.B., DUNCAN, M.E. et al. 2003].

At this stage we have implemented a few sorting algorithms that can be concretized by the robots. Besides, with this work, we are developing a spatially and behaviorally oriented framework in which students can interact with robots and, at the same time, learn the basics of algorithms watching how robots behave with the algorithm designed by the students themselves; thus students would not need to program the algorithm with a high level programming languages but they could concentrate on the main aspects of the algorithm (*invariants*) [HUNDHAUSEN, C., and DOUGLAS, S. 2002].

We have carried out a preliminary evaluation with 13-to-15-year-old students. Through the test we gather information about how students understand the operation of a sorting algorithm by means of questions and practical tasks. The analysis of the results shows that students learn how one algorithm works, although further tests and analysis must be done to determine how the concretization should be best utilized.

2 FROM ALGORITHM VISUALIZATION TO ALGORITHM CONCRETIZATION

2.1 INTRODUCTION TO COMPUTER SCIENCE INSTRUCTION

We have claimed before that one of the main difficulties that novice students in Computer Science face is the understanding of algorithms. It is somewhat surprising because current students have been living in a more and more computer-based environment, so maybe they might grasp easier how algorithm works. But it does not seem to happen this way.

Students indeed are living among computers. But their experience is mainly focused on how computer programs *interact* (i.e. video games, web browsers, people) and particularly on how the students interact with them. The idea of *interaction* relies on the concept of an immediate *feedback*. Besides this, the computer programs are *concurrent*: apart from the running of a given computer program (more precisely, a thread), other things are happening at the same time (i.e. a word processor). Likewise, students are visually oriented: TV, movies and game consoles are widespread around the world. Eventually, the structure of the text-based information has dramatically changed with Internet and the use of hyperlinks and their multiple ways of connecting concepts (opposite to a linear structure of concepts). To sum up, the students' current point of view about Computer Science is *dynamic* [ANDREA, L. 1997].

However, all these changes are not yet present in the paradigm of Computer Science instruction: teachers mainly use verbal explanations combined with blackboards or slides, which consists of text-based information and in a rather *static*, poorly interactive fashion. As we have stated before, a natural approach to algorithms directs attraction to show how the information (data flow, control flow) evolves over time.

Sometimes students feel discouraged because they sense that what they are studying is "far from reality": they don't know the potential uses of that knowledge or how to apply that knowledge. Concretization makes the identification of this link easier. Students are more easily involved and motivated and their learning will be fostered [JADUD, M. 2000].

2.2 ALGORITHM ANIMATION

Algorithm animation illustrates the behavior of an algorithm by producing an abstraction of the *data* (how data evolves) and the *control flow* (a representation of *all* possible sequences of events) of the algorithm. The main focus has been on visualization of algorithms constrained to computer screens. The results of the research have been questionable. There are studies that support as well as studies that do not support the use of visualization in algorithm teaching [FLEISCHER, R., KUČERA, L. 2002].

The key feature for the research that supports visualizations has been the level of engagement of the students which induces to achieve better results. With our work we want to transfer the abstract concepts from computer screens to the physical world. Concretization of algorithms seems to provide a higher level of interaction and engagement potential for the students, allowing them a more hands-on style of learning [KERREN, A. and STASKO, J.T. 2002]. Besides, concretization gives the students an immediate feedback which allows them a deeper insight of algorithm's activity and allows them to concentrate on the main features of the algorithm (*invariants*), and not in the programming details. This also elaborates on the common idea of interface in Computer Science education. Instead of interacting with keyboards or computer screens, students can interact with their ideas in a different way: they can even *touch* them [LÓPEZ, J., MYLLER, N., SUTINEN, E. 2004].

2.3 ROBOTICS IN EDUCATION

As the price and complexity of robots have decreased in the last years, their use in Computer Science education has grown in different subjects such as AI, distributed and parallel programming, etc. [SCHUMACHER, J., WELCH, D., and RAYMOND, D. 2001] [KLASSNER, F. 2002]. The central idea has been that robotics could motivate students to learn but the results have been contradictory [FAGIN, B., and MERKLE, L. 2003] [WOLZ, U. 2001]. On one hand, there is the novelty and fascination of this new tool; on the other hand, the lack of experience and integration into a traditional learning environment. Another approach has been to simulate the robot's behavior through computer software and after that introduce either real robots or another programming language [FAGIN, B. 2003] [PATTIS, P. 1981].

Our approach is somewhat different. We try to make the abstract subject, such as algorithms, more concrete through robots which behave according to the given algorithm. Students, for now, watch and interact with the robots only affecting the external factors, such as their placement or appearance. This is closer to visualization of the algorithms where students can mainly affect the input sets or appearance of the algorithm. With the framework we are developing we want to go further in this approach in order to enable the students to concretize their own algorithms. For instance, within a GUI they can drag & drop an image representing the robot to control its location. Then the students can design a high level behaviour by sending and receiving messages from one robot to another

3 ALGORITHM CONCRETIZATION WITH ROBOTICS

3.1 TECHNICAL OVERVIEW

In this prototype phase, we decided to use a LEGO^{TM} Mindstorms RCX unit as a base for the robots. This gave us more freedom to easily modify and program the robots with a LEGO^{TM} robotics kit. The RCX units can send and receive signals through infrared (IR) transceivers. The concretized algorithms were developed using NQC (Not Quite C) and the LeJOS (Lego Java OS) API to evaluate the ease of use of these systems. Although we have used these environments for this first system, we are not restricted to their solely use only: any robot with a programmable interface can be used, such as Sony ^{TM} AIBO robots or cricket technology [RESNICK, M., BERG, R., EISENBERG, M. 2000].

At this point, we decided to concentrate on data flow oriented visualizations rather than concretizing the control flow of the algorithm. This is partially due to the fact that the latter would involve too many details for a novice student.

There were two different kinds of options to implement the communication and computation between the robots. The first (*centralized*) possibility was to compute all the commands in the control unit and then send them to the robots. The other (*decentralized*) option was to distribute the same behavior to all robots and depending on the placement they would act differently. We introduce briefly both approaches and then explain our decision.

The first idea was to split up the algorithm itself and the associated behavior: a control unit implements the algorithm (e.g. sorting algorithm) with communication commands inserted within it. Those commands tell the rest of the robots what to do and when; these commands are closely related to the interesting events in algorithm visualization [KERREN, A. and STASKO, J.T. 2002]. Robots communicate with each other while the control unit keeps track of what is happening in order to achieve synchronization.

The second possibility was a decentralized approach, where a copy of the algorithm would be running on every robot. Therefore, none of the robots would have control on the overall behavior but all the robots should be rather autonomous. This approach introduces new kinds of problems of distributed computing and parallel algorithms that need to be solved; these are far from our original aim. However, this could lead into new possibilities to visualize parallel or distributed algorithms.

We decided to use the first approach, centralized computation, because it is more feasible with robots with many limitations, especially in sensing. With this approach, the control can be kept simple. The main problem is reduced to the design of a protocol between the control unit and the robots and the synchronization of the messages.

We use one of the RCX units as the control unit commanding the robots in order to be able to execute the algorithm concretizations without an additional PC. The protocol was designed to be as simple as possible. The robots were given a unique *id,* to which only that robot reacts. That *id* represents the location of a robot in an array. After the control unit makes a connection to a robot by sending its *id*, that robot is given a role and it will act according to the given role. The robot sends a message to the control unit after finishing the acting of the role. Then the control unit assigns new roles to other robots and the algorithm's execution goes on (see Figure 1).

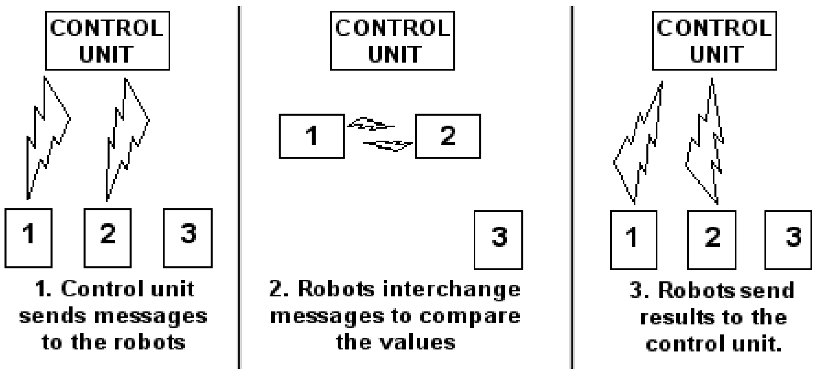
CONTROL UNIT
1
2
3
1. Control unit sends messages to the robots
CONTROL UNIT
1
2
3
2. Robots interchange messages to compare the values
CONTROL UNIT
1
2
3
3. Robots send results to the control unit.

Figure 1 – Different stages during to event of comparison.

To sum up, each relevant operation (i.e. interesting event) in algorithms can have a new semantics, more concrete, less abstract, which symbolizes or represents the activity of the algorithm by means of robots' behaviour. For instance, in a sorting algorithm, comparison between two variables can be concretized as an event where two robots move towards each other, communicate their values and rearrange themselves if necessary. This corresponds to the visual semantics of the operation in algorithm visualization.

3.2 THE EXAMPLE ALGORITHM AND IMPLEMENTATION DETAILS

In this first version of software, we have implemented two algorithms: selection sort and bubblesort. Here we briefly explain the technical issues concerning the implementation of the bubblesort algorithm. The idea of bubblesort consists of advancing through an array and finding the first minimum element then the second and so on. Every time two items are not sorted, their values are swapped.

```
for (j = 1; j <= vector.length; j++) {
  for (k = j + 1; k <= vector.length; k++) {
    if (vector[j] > vector[k]) {
      temp = vector[j];
      vector[j] = vector[k];
      vector[k] = temp;
    }
  }
}
```

Figure 2 – A possible implementation of bubblesort algorithm.

The concretization of the algorithm goes as follows: the control unit executes the algorithm similar to Figure 2 and on the interesting events it calls the robots. The array of items is concretized as a line of an arbitrary number of robots (see Figures 1 and 3). In this algorithm, the point when the control unit calls the robots is in the comparison between the elements of the array, in the if-condition (line 3 in Figure 2). The control unit calls a specific robot by its *id* and waits for its answer.

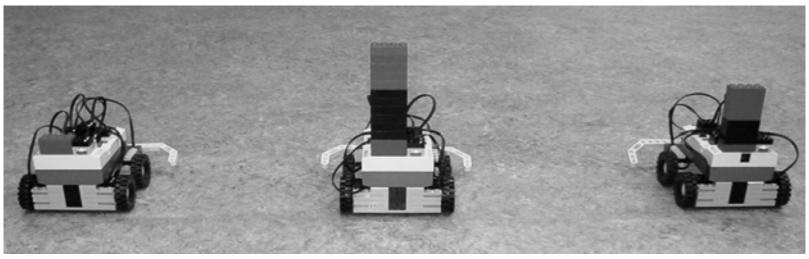
Figure 3 – Robots in the line before the sorting.

When taking the viewer into account, there will be a robot on the left and right hand side during the comparison, with different movements. Thus, we differentiate between these two different behaviors or roles: the *left role* and the *right role*. Furthermore, every robot can be assigned with the left or right role at any step of the algorithm. After a robot has been called with its *id*, the control unit assigns the correct role to it. Once being called and given a role, robots move forward and turn according to their role, either to the right or left in order to be compared (see Figures 1 and 4). Every robot has a *weight* or value that represents an array item. In that moment, the robots must compare their weights in order to know if they are sorted or not. The robot on the left communicates its value, as expressed by the external property such as height or weight, to the robot on the right and it compares the value with its own value. If the values are already in the right order, i.e. the robot on the left has a value less or equal than the robot on the right, a FALSE message is sent back and they return to their original locations. If the values are not sorted, i.e. the robot on the left has a greater value compared to the robot on the right, they exchange their *id*s and their locations afterwards. Finally, in both cases, when robots turn back to go to the line, they send a DONE message to the control unit.

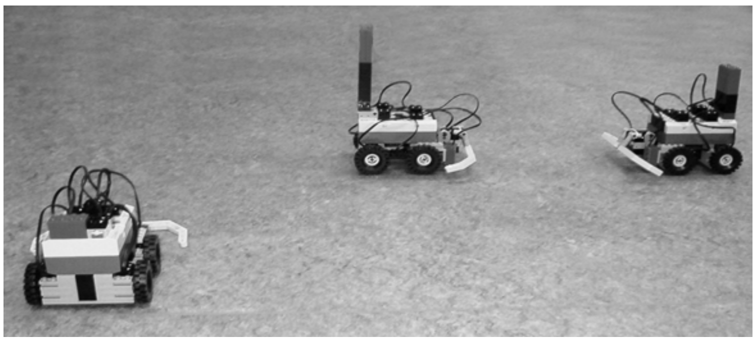
Figure 4 – Robots' locations during the comparison state. Robot in the middle has the left role and robot on the right has the right role.

After receiving two DONE messages, the control unit can continue to the next round in the iteration of the inner loop. If the control unit calls the right robot and does not receive any answer, that absence is interpreted as the end of the inner loop. Taking a look to the Figure 2, when *j* is equal to *k*, the algorithm is finished and the control unit notifies the robots by sending an ACK message that will terminate their execution. Figures 5 and 6 illustrate the protocols between the control unit and the robots.

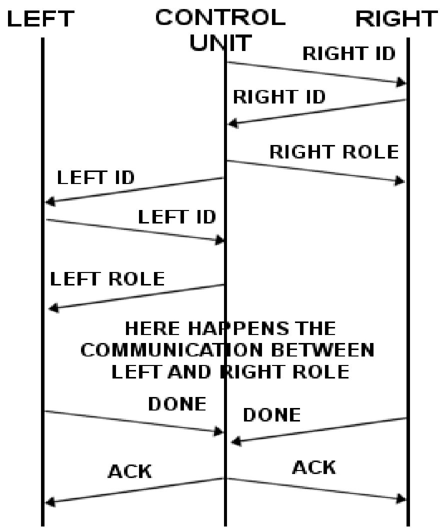

Figure 5 – Protocol between the control unit and the robots in the bubblesort algorithm.

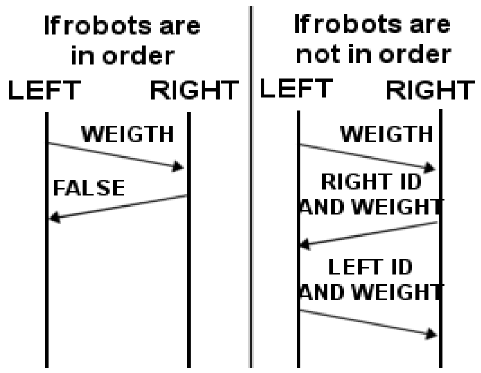

Figure 6 – Protocol between robots in the bubblesort algorithm.

With this design, *concurrency* and *interaction* among robots are showed, and a holistic, visual feedback is given to the novice student. Apart from this, there are different kinds of information that the control unit and the robots should have. The control unit should have implemented the algorithm and know all the commands that are available from the robots. Additionally, it must know when to send those commands to the robots during algorithm execution. On the other hand, the robots should know their *id*, the commands and their *behavior* depending on the given commands. This is also a concretization of the concept of *object*: data and methods, data (*weight*) and behavior.

3.3 IMPLEMENTATION EXPERIENCES

The implementation of the system was straightforward as far as the programming part was concerned. However, physical restrictions and differences between the RCX units and its components such as motors introduced new kinds of problems. In particular, it was difficult to achieve movement control. There were several physical factors to take into account when considering elementary movements and precise turning: the power of motors, the different response times depending on the battery power and friction from the floor just to mention a few. This led us to store physically related information, such as the distance between robots and the speed for backward and forward movement and turning around into the robots.

Another issue was the tuning of the synchronization times for sending and receiving the messages. It would help if robots had another means for communication than infrared ports because infrared communication requires line-of-sight to work properly. Moreover, the environment needs to be free of other infrared interferences because IR transceivers are quite sensitive to noise and the communication is very slow.

Java and LeJOS API were more applicable in this context. The richness of LeJOS API made the development of the system easy. However, we do not think that first or second year students should do their own concretization with Java language because of the low level implementation requirements. Thus they need a separate environment to work with, otherwise most of the effort is put into the programming and not into the design of the concretization.

There were no major problems when implementing the system with NQC. However, we think that NQC is a restricted language compared to Java and can cause problems in a later development. Especially, if we change to another robotics system it is more likely that a Java API is available for that system.

4 EVALUATION

In this section we introduce the empirical evaluation and the preliminary results from the experiment done with robotics concretization. We were using a single sorting algorithm, bubblesort. We used different methods of teaching during the experiment. We collected only qualitative data. In this section we describe the lesson and how the students answered the questions.

4.1 METHOD AND MATERIALS

The experiment was carried out with five 13-to-15-year-old students in an after-school computer club [ERONEN, P.J., SUTINEN, E., VESISENAHO, M., VIRNES, M. 2002]. All of them knew the basics of programming but only one of them had studied the bubblesort algorithm before. The empirical data was collected with a questionnaire and by video taping the lesson.

First, students were introduced to the problem of sorting with five boxes in a row having different sized items inside of them, presenting an unordered array. Students were asked to sort the items.

Secondly, the concretization of the bubblesort algorithm was introduced to them with four robots. After the initial view of the algorithm, they were asked to predict how the algorithm proceeds in four different situations not presented before. All situations concerned the swapping of the places in the array. After that they were asked to explain the procedure of the algorithm. They could explain the predictions and the algorithm either by writing or drawing. They were then given the correct answers. Finally, they were asked to sort the items in the boxes according to the introduced bubblesort algorithm and try to describe ways to make the algorithm and the presentation with robots better.

4.2 DISCUSSION OF THE RESULTS

At the beginning when students were asked to sort the items in the boxes they used an ad hoc method to do it. They did not think about the constraints that an algorithm normally has. When they were given the constraint of just opening two boxes at once the sorting was closer to the sorting algorithm but still some ad hoc methods of determining which box to open were used and the process was not systematical.

During the concretization, the students watched eagerly how the robots work according to the bubblesort algorithm. After that, students were shown four different scenarios of the sorting with robots and they were asked to explain the next step of the algorithm. All students answered correctly to three questions. In one of the questions two students were wrong. All students used drawing.

When asked to describe the algorithm two students gave the correct explanation. One student wrote the first round of the inner loop correctly and then just said that it would continue in the same way so we cannot say for sure if he had understood the algorithm. The two other students gave either vague or incorrect explanations.

In the end, the students were asked to sort the items in the boxes according to the bubblesort algorithm. The algorithm was repeated step-by-step so that each of the students made the next step of the algorithm in turns. They were eagerly advising each other how to proceed. They were discussing the algorithm with each other and they repeated the procedure of the algorithm correctly in a new situation. This shows that at least some of the students understood the algorithm correctly.

We also asked the students what they would like to enhance in the current algorithm and system. One of the students gave a suggestion that the computation robot should get hold of all the weights of the robots and then just pick up the smallest amount of robots to make the sorting faster. Even though this proposed method does not make the actual sorting algorithm faster but the concretization of it, it is in a line with two other students who said that the robots should be faster which is true in both communication and movement. Four of the students liked the experience with robots and one did not explain her feelings in any way.

5 CONCLUSION AND FUTURE WORK

We have introduced a novel way to teach algorithms through algorithm *concretization*. This approach combines algorithm visualization and robotics to make algorithms *alive* in the real world. Thus students can construct new knowledge by observing the real world objects. This connects their previous knowledge to the data structures and algorithms at a more concrete level and promotes learning in a *constructivist* way [BEN-ARI, M. 2001]. Another supported method is learning-by-doing because concrete matters make the abstract concepts such as the data structures and algorithms easier to grasp.

Algorithm concretization with robotics can give many kinds of possibilities to design algorithms in a creative way and get new insights to the most vital parts of the algorithm and develop the algorithms further through them. This can for instance lead to a classroom activity where students in teams could try to harm a certain algorithm and the winner team is the one that can find the most vulnerable part of the algorithm. On the other hand, this kind of naturally distributed environment gives interesting possibilities to concretize parallel and distributed algorithms and let the students build their own concretizations of them. Moreover, these algorithms can be more carefully analyzed in real world settings than with a software simulator.

As stated by Hundhausen and Douglas [HUNDHAUSEN, C., and DOUGLAS, S. 2002], expert-made visualization materials do not necessarily promote learning any better than materials made by students. This leads to the idea of developing an environment where students could develop their own concretizations. This should be made easier than just coding a new algorithm and the behavior of the robots through languages like Java or C. To accomplish this, a behaviorally and spatially oriented language and a graphical user interface through which students could assign movements to each role and robot should be designed. This can be achieved by standardizing the interface to the robot's controls leaving the appearance and functionality of the robot still modifiable.

At this stage, we have not yet done a large scale evaluation of the current system in a classroom setting. However, we have shown the concretizations to small groups of students that have been eager to interact with the robots. The results have been promising and given us clear directions how to proceed with our work. Thus, one of our future plans is to do a proper evaluation of the system and teach algorithms by concretization.

Learning algorithms through concretization is an approach that needs to be developed further to find new means to illustrate the execution of the algorithms. This work is the starting point for concretization tools that can make students' learning of algorithms explicit and interesting. The added value of concretization compared to visualization is its hands-on character which has an especial appeal to a certain type of student. Furthermore, concretization through robots supports diverse learners and learning styles or even different *cultural contexts* because robots' behavior and appearance are easily adaptable for the current situation. In this way each student can utilize the tools in a way appropriate to him/her.

REFERENCES

ANDREA, L. Reconceptualizing Computation: Radically Rethinking CS1 [online]. September 1997, [cited 2004-2-23]. Available from Internet: <URL: http://faculty.olin.edu/~las/2001/07/www.ai.mit.edu/ people/las/papers/cs101-proposal.html>

BAECKER, R. M. Sorting out Sorting. [Videotape] 30 minutes, presented at ACM SIGGRAPH '81 and excerpted in ACM SIGGRAPH Video Review #7. Los Altos, CA: Morgan Kaufmann, 1983.

BEN-ARI, M. Constructivism in Computer Science Education. In *Journal of Computers in Mathematics & Science Teaching 20(1)*. 2001, 45 - 73

BEN-ARI, M., MYLLER, N., SUTINEN, E., and TARHIO, J. Perspectives on Program Animation with Jeliot. In *Software Visualization International Seminar, Dagstuhl Castle*. Berlin: Springer, 2002, 46 - 57. LNCS 2269.

ERONEN, P.J., SUTINEN, E., VESISENAHO, M., VIRNES, M. Kids´ Club – Information technology Research with Kids. In *Proceedings IEEE International Conference on Advanced Learning Technologies,* 2002, 331 – 333.

FAGIN, B. Ada/Mindstorms 3.0.: A Computational Environment for Introductory Robotics and Programming. In *IEEE Robotics and Automation 10(2)*, 2003, 19 - 24.

FAGIN, B., and MERKLE, L. Measuring the Effectiveness of Robots in Teaching Computer Science. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*. 2003, 307 - 311.

FAGIN, B., and MERKLE, L. Quantitative analysis of the effects of robots on introductory Computer Science education. In *Journal on Educational Resources in Computing (JERIC)*, *Volume 2, Issue 4*. 2002, 1 - 18. ISSN:1531-4278.

FALTIN, N. Structure Constraints in Interactive Exploratory Algorithm Learning. In *Software Visualization International Seminar, Dagstuhl Castle*. Berlin: Springer, 2002, 213 - 226. LNCS 2269.

FLEISCHER, R., and KUČERA, L. Algorithm animation for Teaching. In *Software Visualization International Seminar, Dagstuhl Castle*. Berlin: Springer, 2002, 113 - 128. LNCS 2269.

HUNDHAUSEN, C., and DOUGLAS, S. A language and system for constructing and presenting low fidelity algorithm visualizations. In *Software Visualization International Seminar, Dagstuhl Castle*. Berlin: Springer, 2002, 227 - 240. LNCS 2269.

HUNDHAUSEN, C., DOUGLAS, S., and STASKO, J. A Meta-Study of Algorithm Visualization Effectiveness. In *Journal of Visual Languages & Computing, 13(3)*. 2002, 259 - 290.

JADUD, M. TeamStorms as a Theory of Instruction. In *Proceedings of the 2000 IEEE International Conference on System, Man & Cybernetics*. 2000, 712 - 717.

KERREN, A. and STASKO, J.T. Algorithm animation Introduction. In *Software Visualization International Seminar, Dagstuhl Castle*. Berlin: Springer, 2002, 1 - 15. LNCS 2269.

KLASSNER, F. A case study of LEGO Mindstorms' suitability for artificial intelligence and robotics courses at the college level. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education.* Northern Kentucky, USA, 2002, 8 - 12.

LAWHEAD, P.B., DUNCAN, M.E., BLAND, C.G., GOLDWEBER, M., SCHEP, M., BARNES, D.J., HOLLINGSWORTH, R.G. A road map for teaching introductory programming using LEGO© mindstorms robots. In *Working group reports from ITiCSE on Innovation and technology in computer science education*. Aarhus, Denmark, 2003, 191 - 201. ISSN:0097-8418.

LÓPEZ, J., MYLLER, N., SUTINEN, E. Sorting out sorting through Concretization with Robotics. Accepted for publication in *AVI - Advanced Visual Interfaces 2004.* 2004.

NARAYAN, S., TOMPKINS, J. Using Robotics to Enhance Learning in Introductory Computer Science Courses. In *ACMSE Proceedings 41*, 2003, 412 – 415.

PATTIS, P. *Karel the Robot: A Gentle Introduction to the Art of Programming*, *2nd ed.* John Wiley and Sons, 1981.

RESNICK, M., BERG, R., EISENBERG, M. Beyond Black Boxes: Bringing Transparency and Aesthetics Back to Scientific Investigation. In *Journal of Learning Sciences, 9(1),* 2000. 7 – 30.

SCHUMACHER, J., WELCH, D., and RAYMOND, D. Teaching introductory programming, problem solving and information technology with robots at West Point. In *ASEE/IEEE Frontiers in Education Conference*, *F1B vol.2, October 10-13, 2001*. 2001, 2 – 7.

WOLZ, U. Teaching Design and Project Management with Lego RCX Robots. In *Proceedings of the 32th SIGCSE Technical Symposium on Computer Science Education*, *February 21-25, 2001, Charlotte, Carolina, USA*. 2001. 95 - 99.