Robin the Robot: The first steps to creating a robotic learning companion

Lizsandra Zuniga
Summer Research at Arizona State University, Tempe, AZ, USA
zunigl0@sewanee.edu

Abstract

Technology has become a crucial part of everyday life. This paper presents the first steps to creating Robin, a robotic learning companion, that will make programming more interactive for children (ages 8-12) . These steps include observing summer camps that teach programming, creating modules, creating production rules, and the beginning of implementing the code. In the future, social human-robot interactions such as those created by Robin might improve learning by motivating students and giving them feedback on their learning tasks.

Introduction

Due to their physical embodiment, robotic learning companions have the potential to create more effective social interactions, enhance motivation, promote learning, and provide a scalable educational experience [6, 5]. For example, Kory successfully used a DragonBot for early language development [5]. Robin will provide these experiences to children learning to program.

Robin is a continuation of a previous learning companion called Quinn, an algebra learning companion, which contained a voice-adaptive speech interface, and was socially responsive [6]. Comprised of a iPod-Touch fixed on a LEGO® Mindstorms® NXT robot, Quinn also exhibited lively facial expressions and speech when interacting, and neutral expressions otherwise [6, 7]. A typical LEGO® Mindstorms® EV3 contains motors, light and touch sensors [8].

Robin will contain all the qualities Quinn had, but will now be for programming, instead of algebra. Robin will use the next generation of LEGO® Mindstorms® EV3. This next generation contains an improved EV3 brick increasing its efficiency[4]. The Lego Mindstorm kit comes with its own graphical programming language that is easy for children to learn [8] .

We believe that Robin will be more beneficial and helpful in a programming setting than a math setting. Robin will be used to provide feedback and motivation for children learning how to program LEGO® Mindstorms® EV3. Since it is for a programming setting, *FIRST* (For Inspiration and Recognition of Science and Technology) LEGO league programs were observed to gain a better understanding of what challenges to create. Then modules, or the challenges, were created. Followed by production rules, and the execution of the code. This paper will discuss the experience in further detail.

FIRST LEGO League Program

FIRST is an international organization that expose children to STEM (science, technology, engineering, and mathematics) fields at an early age. Every Tuesday and Thursday a team of undergraduates including myself attended the *FIRST* LEGO league program based at Arizona State University in Tempe. There the team investigated how children approached programming problems and which errors were commonly made. This experienced allowed me to compile notes for the possible modules that would be feasible for a child to complete in one day.

Creating the Modules.

The objective was to create five modules that could be completed within a day. To start off, a comparison had to be made between *Robotic Graphical Introduction to Programming LEGO® MINDSTORMS EV3*, a robotic simulation environment made by Carnegie Mellon Robotics Academy, and the powerpoint used in the *FIRST* LEGO league program [9]. I worked through both examples and compiled notes on each. From the notes, similarities were found for the challenges presented in *Robotic Graphical Introduction to Programming LEGO® MINDSTORMS EV3* and the powerpoint used in the *FIRST* LEGO league program. Using bold font the module was named, such as **Moving Forward**, and below was a description of how the challenges varied between the two platforms. For example, for **Moving Forward** the difference was the distance (one used 50cm, the other used 85cm). Two sets of five modules were created. One set flowed very nicely, and would prepare the students for the last module, while the other was the same with the exception of one. Based on the observations of the *FIRST* LEGO league program, the different modules would be more difficult but is a good way to observe problem solving skills in children. The chosen module set included **Moving Forward**, which programmed the robot to move forward, **Retrieving an Object**, in which the robot retrieved an object by lowering its arm and returning to its starting position, **Turning**, which programmed the robot to undergo a 90 degree right turn , **Turning Continued**, which programmed the robot to turn to a destination, and **Sensors**, which programmed the robot to sense a wall and turn when sensed.

Creating Production Rules

Production rules, which relate task goals and states to actions and consequences, are based on a theory that cognitive skill or thinking consists in large part of units of goal related knowledge [2]. Production rules vary between different disciplines, but do have common features [1]. They consist of a condition and action pair presented as an If (condition) and Then (action) statement. For example, **Moving Forward**:

If The goal is to move forward 85cm
And the left wheel has to be +u
And the right wheel has to be +v
And one rotation = x
Then Conclude number of rotations is y and y = 85/x

The initial production rule was an algebraic expression because the actual numbers were not yet available. There are a total of four variables and three conditions that have to be met for a successful module. In the initial draft the overall goal to move forward 85cm, is first stated, followed by three conditions, that state the right and left wheel have to have a positive number rotation (*+u, +v*), and one rotation is x (distance of rotation in cm). I later discovered that there were more variables to consider to create more accurate production rules, such as, left wheel speed, right wheel speed, and number of rotations. Finally we have the action statement that concludes that *y* is the number of total rotations, and *y* is 85cm divided by x (distance of one rotation in cm). I had difficulties making production rules for the other four modules because they required more than one movement. I learned that the overall goal can be decomposed into subgoals, for example for the modules that required more than one movement, the overall goal was stated followed by subgoals that outlined each movement and concluded that if all the subgoals were met then it was complete [1]:

If The goal is to retrieve an object by lowering the arm and returning to its starting position
- If The the sub goal is to raise the robot's arm
  And the robot's arm is in the down position
  And the number of rotations has to be positive
  And the speed has to be negative
- Then Conclude rotation is x and speed is -z
- If the subgoal is to move forward 85cm
  And the left wheel speed has to be +a
  And the right wheel speed has to be +b
  And the rotation has to be positive
  And one rotation = c
- Then Conclude number of rotations is d and d = 85/c
- If The sub goal is to lower the robot's arm
  And the arm is already in the up position
  And the number of rotations has to be positive
  And the speed has to be positive
- Then Conclude rotation is y and speed is +w
- If the subgoal is to move backward 85cm
  And the left wheel speed has to be -e
  And the right wheel speed has to be -f
  And the rotation has to be positive
  And one rotation is g
- Then Conclude number of rotations is h and h= 85/g

Then Conclude all the subgoals have been met

Production rules are capable of constructing sequences that represent correct solutions of the problem [2]. Essentially, correct actions to the solutions will praise the student, while the students will be guided to the correct procedure for incorrect solutions[2].

The Code

The code consisted of Java files and XML or Extensible Markup Language, which encodes a description of the document's storage layout and logical structure making it readable for humans and machines[3]. LEGO® Mindstorms® EV3 creates a XML document when the file is saved,

enabling the Java to read the file. The LEGO® Mindstorms® EV3 contains elements with attributes such as:

<ConfigurableMethodCall Id="n2" Bounds="112 34 194 91" Target="MoveTankDistanceRotations\.vix">

The target, "MoveTankDistanceRotations" is the action block that controls the motor for the wheels. Within the previous element there are other elements with attributes that contain the left wheel speed, right wheel speed, and rotations. The Java file used the LEGO® Mindstorms® EV3 XML document by creating a file object from the XML document. The Java file contains several for loops within one another. The first for loop traverses through the element "ConfigurableMethodCall" and creates a node object for this "ith" iteration. Then casts the node object to an element object. Lastly, a NodeList is created from the "ConfigurableMethodTerminal" nodes that are within the "ConfigurableMethodCall":

```
for (int i = 0; i < nList.getLength(); i++) {
  Node cmc_node = nList.item(i);
  Element cmc_element = (Element) cmc_node;
  NodeList cmtList =
cmc_element.getElementsByTagName("ConfigurableMethodTerminal");
```

Within the for loop there is another for loop that also creates a node object for the "jth" iteration, casts a node object to an element object, pulls the "ConfiguredValue" value and stores it in a string, and creates a NodeList object from the "Terminal" node that is contains within each configurable method terminal element:

```
for (int j = 0; j < cmtList.getLength(); j++)  {
  Node cmtNode = cmtList.item(j);
  Element cmtElement = (Element) cmtNode;
  String possibleKeyValue = cmtElement.getAttribute("ConfiguredValue");
  NodeList terminalList = cmtElement.getElementsByTagName("Terminal");
```

The last for loop iterates through the the "terminalList", creates a node object from the terminal node, casts a node object to an element, and checks the "Id" value of the "terminalElement":

```
for (int k = 0; k < terminalList.getLength(); k++)  {
  Node terminalNode = terminalList.item(k);
  Element terminalElement = (Element) terminalNode;
  String aspect = terminalElement.getAttribute("Id");
```

The for loop also contains a series of if-else statements, which created the first coding problem. The second module contained a series that involved raising the EV3's arm, moving forward, lowering the the arm, and moving backwards. In the XML document both moving forward and moving backward actions are under the "ConfigurableMethodCall" "Target" element as the attribute "MoveTankDistanceRotations\.vix". The arm actions are both under "MotorDistanceRotations\.vix". Because they are both under the same attribute, the specific feedback is not allowed in the form of a print output. For example with the moving forward action, the if-else statements should have contained a condition checking that both left and right speeds were positive values. There also should have been another statement checking that the rotations were in the given range. If everything was correct the output would have been "Success, you completed this module! ", but if something was wrong, such as having negative

values, it would have returned a statement such as “The robot is moving in the wrong direction”. Since both moving forward and moving backward actions were under the attribute “MoveTankDistanceRotations\.vix” the Java file would output both statements, rather than giving one message.

The solution at the time was to create if-then statements that incremented and tallied “moveTankPositiveCount” (moving forward), “moveTankNegativeCount” (moving backward), “positiveArmCount” (lifting the arm), and “negativeArmCount” (lowering the arm) and ended with following statement:

```
if (totalCount == 4)  {
  if (moveTankPositiveCount == 1 && moveTankNegativeCount == 1
      && positiveArmCount == 1 && negativeArmCount == 1)
  {
        System.out.println("Success");
  }
  else
  {
    System.out.println("Fail");
  }
 }
```

This code is a first step towards implementing the production rules that enable feedback specific guidance for children during their programming experience. At the moment, the output simply states whether or not they passed the overall module. If they “failed,” it would not give any guidance. However, a solution for this issue is underway.

The Conclusion

This paper discusses the first steps of the Robin project. Although the initial code failed to accomplish its goal, it provided valuable insight that enables the team at ASU in Tempe to continue to work on the ongoing Robin learning companion.

The Acknowledgements

I would like to thank the DREU (Distributed Research Experiences for Undergraduates) program for this wonderful research opportunity. I would also like to thank my mentor Dr.Walker for guiding me and providing all the resources necessary for the project. I would like to thank Nichola Lubold for letting me work on her innovative Robin project. Thanks to Nick Martinez for working alongside me through this project. Thanks to the 2 Sigma Learning Lab for providing an inspirational work environment. Lastly I would like to thank Arizona State University in Tempe, the CRA-W (Computing Research Association for Women), the CDC (Coalition to Diversify Computing), and the National Science Foundation.

Citations

[1] Anderson, J. R., Boyle, C, F., Corbett, A. T., & Lewis, M. W. "Cognitive Modeling and Intelligent Tutoring." In *Artificial Intelligence and Learning Environments,* (pp. 7-49). 1990.

[2] Anderson, J. R., Corbett, A, T., Koedinger, K. R., & Pelletier, R. "Cognitive Tutors: Lessons Learned." *The Journal of the Learning Sciences,* 4(2), 167-207, 1995.

[3] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F., & Cowan, J. "Extensible Markup Language (XML) 1.1 (Second Edition)", 2006. [Online]. Available: http://www.w3pdf.com/W3cSpec/XML/2/REC-xml11-20060816.pdf

[4]Green, A. A. "Lego® Mindstorms ev3 and Lego engineering conference", [Online]. Available: https://community.education.lego.com/legodev/attachments/legodev/LE_LessonPlanLibrary_TKB/17/1/LEGO%20EV3%20lo-res.pdf.

[5] Kory, J. M., Jeong, S., & Breazeal, C. L. "Robotic Learning Companions for Early Language Development." In *Proceeding ICMI '13 Proceedings of the 15th ACM on International conference on multimodal interaction,* (pp. 71-72). ACM, 2013.

[6]Lubold, N., Pon-Barry, H., & Walker, E. "Effects of Voice-Adaption and Social Dialogue on Perceptions of a Robotic Learning Companion." In *Proceeding HRI '16 the Eleventh ACM/IEEE International Conference on Human Robot Interaction, 2016 IEEE,* (pp. 255-262). IEEE, 2016.

[7]Lubold, N., Pon-Barry, H., & Walker, E. "Naturalness and Rapport in a Pitch Adaptive Learning Companion." In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop*, 2015.

[8] Schep, M., McNulty, N. " Use of Lego mindstorm kits in introductory programming classes: a tutorial ." *Journal of Computing Sciences in Colleges,* 18(2), 323-327, 2002.

[9]Schoop, R. "Fostering Innovation through Robotics Exploration (FIRE)", [Online]. Available: www.dtic.mil/dtic/tr/fulltext/u2/a622795.pdf.