Toward Mutual Theory of Mind as a Foundation for Co-Creation

Bobbie Eicher, Kathryn Cunningham, Marissa Gonzales, Sydni Peterson, Ashok Goel
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30332
{beicher3, kcunningham, mgonzales9, speterson33}@gatech.edu, goel@cc.gatech.edu

Abstract

Despite increasing emphasis on the importance of helping people to understand how computers work, success in teaching new programmers to properly understand languages and how they behave has been limited. Our work focuses on how the cognitive science concept of *theory of mind* could be applied to better teach computational thinking. We achieve this by creating a prototype that builds a theory of mind for the user's misconceptions in the context of beginning programming, and then leverages that as a tool for helping the user to build a more accurate mental model of the prototype itself.

Introduction

Human-computer co-creativity requires deep human-computer collaboration that goes beyond the usual notion of turn taking. Instead, co-creativity requires human-computer collaboration that emulates the depth of human-human collaboration. Cognitive science theories of human-human collaboration suggest that the collaborating humans have a theory of each other's minds, a theory that ascribes and expresses intents and beliefs to each other. Further, each human's actions on a team are predicated on his/her, perhaps incorrect, theory of teammates. We posit that deep human-computer collaboration will need to emulate human-human collaboration in that both the human and the computer will need a theory of each other's minds.

In the learning sciences, there has been significant work on how a computer - an AI agent - may be given, or might acquire, a theory of a user's mind. This is sometimes called a user model that captures the user's goals, current beliefs, trajectory of learning, etc. However, there has been relatively little work on how a human may be given, or might acquire, a theory of a computer's or a computer program's mind. Naturally this brings us close to human understanding of how a computer program works, including misconceptions in the understanding.

Background

Theory of mind

Theory of mind was a concept first introduced by Premack and Woodruff, in the context of experiments on chimpanzees. The authors define theory of mind to mean "[that] the individual imputes mental states to himself and to others (either to conspecifics or to other species as well)" (Premack and Woodruff 1978). Neurotypical adult humans are capable of examining a situation involving other humans, understand that distinct individuals may have separate belief and knowledge about a situation, and drawing conclusions about kinds of behaviors and responses that may result from those differences (Premack and Woodruff 1978).

Computer science education

Arguments that Computer Science (CS) should be central to a liberal arts education go back decades, but the field nonetheless faces poor retention, high failure rates, and declining diversity. Studies indicate that introductory programming courses fail to teach many students how to independently write code to solve small problems (Utting et al. 2013). Students may complete such a course 'successfully' and yet still be unable to even read and understand short pieces of code (Lister et al. 2004).

Research argues that much of the struggle originates in misunderstandings of the *notional machine*, the process a computer uses to process code in a particular language or paradigm (Sorva 2013). Students must construct a complex mental model of the notional machine in order to understand program execution. Better technologies are required for understanding what is happening in this model built by the students, and how it differs from the correct model, so that errors can be corrected.

Importance to collaboration

Research indicates members of a group have the ability to effectively understand the mental state and motives of other members, and is predictive of the collective intelligence for said group, to a degree not explained solely by assessing the individual intelligence of each member (Woolley et al. 2010). This work was initially heavily based on tests like "Reading the Mind in the Eyes," which is specifically designed to measure an individual's capacity for "mentalising," an alternative term for theory of mind (Baron-Cohen et al. 2001). Experiments found a positive correlation between the scores on this test and the collective intelligence of a group, which supports the idea that the most effective collaboration happens between parties that each have a strong theory of mind for the other.

Thus far much of the focus on theory of mind and collaboration has focused on individuals interacting in the same physical space, measuring the ability to read face-to-face interactions like eye movements and expressions. However, research indicates it is equally predictive in on-line settings where group members never see each other and cannot rely on physical cues (Engel et al. 2014). Based on this, we have reason to believe that humans are capable of developing useful mental models in interactions mediated via technology.

The notional machine

The notional machine was first proposed as a way of explaining how humans understand computers in 1986 (Du Boulay 1986) and has become a key framework for understanding how humans connect to programming languages and the computer itself as they learn to code (Sorva 2013). A given notional machine is an abstraction of the true operation of a computer as it processes code, and its precise nature will vary depending on the language and paradigm involved. One of the key challenges for beginning programmers is to develop an understanding of the notional machine specific to the work they're doing, and often language they're learning, as to effectively interact with the computer and direct its behavior (Sorva 2013).

One of the ways this is made explicit in the learning process is to have students trace the action of code and attempt to understand what individual lines and operations of code will achieve when executed (Sorva 2013). Even those students who successfully make it through a course, however, may not gain a strong enough grasp of the notional machine to perform this task accurately (Lister et al. 2004).

Misconceptions about variable assignment

Misconceptions about assignment statements in the procedural programming paradigm are some of the most well-studied types of misconceptions. In typical introductory computing courses, variable assignment is a fundamental concept that is learned early and which underlies other concepts throughout the rest of the semester. Misconceptions about variable assignment are also well-constrained, since they are less complex syntactically and require fewer steps to execute as compared to more complex programming structures like loops and selection.

We began with a set of misconceptions about assignment statements identified by Linxiao Ma as prevalent among beginning programmers in the Java language (Ma 2007) and then narrowed this list to a smaller set we felt would be likely to occur in Python as well, and would be suitable for implementation in a prototype (see Table 1).

Our approach

We have approached this problem with the hypothesis that the notional machine can be understood as a special case of theory of mind. The student, therefore, has a theory of mind for how the computer operates. Our tool is an attempt to give the computer the ability to develop a theory of mind about the student and to use that in collaboration with the student to correct misconceptions.

Table 1: Misconceptions modeled by our system

Name (in Ma 2007)	**Action during variable assignment** (e.g. `x = y`)
Mr 2	The value in the variable on right-hand-side is given to the variable on the left-hand-side, and then the content of the original variable on right-hand-side is erased (i.e. the variable still exists, but no content is in the variable)
Mr 4 ***(Correct)***	The value in the variable on the right-hand side is copied into the variable on the left-hand-side
Mr 6	The value in the variable on left-hand-side is given to the variable the right-hand-side, and then the content of the original variable on the left-hand-side is erased
Mr 8	The value in the variable on the left-hand-side is copied into the variable on the right-hand-side
Mr 10	Variables swap values
Mr 99 (not from Ma 2007)	Assignment occurs correctly, but in subsequent assignment statements, updating one variable in the assignment statement will update the other variable.

It develops a precise understanding of a student's beliefs and misconceptions about assignment statements. To do this, the program has a set of common misconceptions that it can model in addition to the ability to the correct version of the execution. The agent can use these models to determine what correct or incorrect model a specific student is likely using, and alert them appropriately while also providing visualizations that contrast what they're expecting against the correct behavior. In this way, it can help the student develop an accurate understanding of how the code behaves.

Our tool

We developed a GUI tool that allows students, teachers, and researchers to create and interact with the mistaken models of the notional machine in a variety of ways.

The tool functions by first taking the input (several lines of code) and converting the code into an intermediate representation of what function each line serves. It then interprets the intermediate representation and computes over it, tracking the active variables and their current state. Based on this, it generates visualizations of both potential misconceptions and the behavior of the correct notional machine.

Execute mode

Functionality The primary interface is 'execute mode', where a user inserts program code, selects a programming language and a misconception, and runs the code according to the corresponding model of the notional machine.

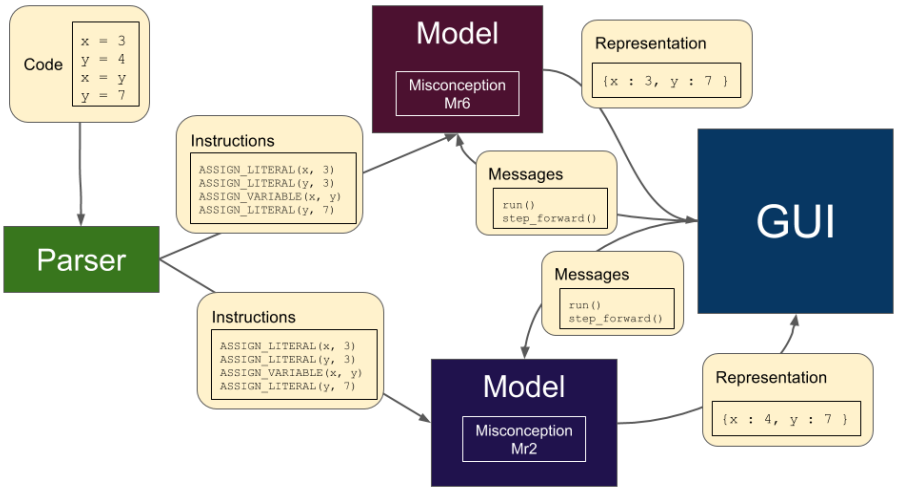

Figure 1: Model creation and execution process

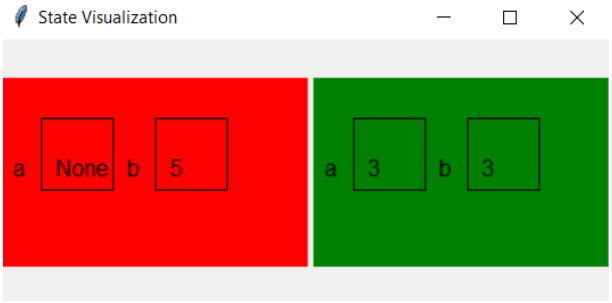

Figure 2: Visualization of program execution

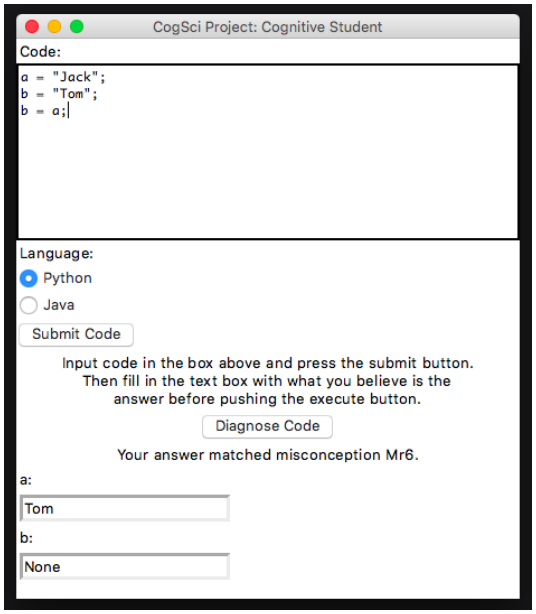

Figure 3: Interface for diagnosing a misconception

Application Execute mode is a straightforward way to see what the expected output would be during cognition with a specific misconception. Knowing what student misconceptions are most common has been shown to be associated with more effective teaching (Sadler et al. 2013). Introductory programming teachers can use this mode to generate expected answers for code they will demonstrate in class, for a variety of misconceptions. Teachers can also use this mode to create effective "distractor" answer options for multiple choice questions.

Visualization mode

Functionality The visualization setup uses a combination of variable names and boxes to create a side-by-side display of both the model requested and the correct model for easy comparison. The red background in this figure represents the mistaken case, and the green side represents the correct execution of the code. A user can click through the code line-by-line to see exactly what is happening and how the two interpretations differ at each step.

Application Visualization mode can be used by students to better understand how their thinking about the notional machine is different than the correct understanding of the notional machine. Students are forced to simultaneously activate of correct and incorrect conceptions, a process which is associated with knowledge revision (Van Den Broek and Kendeou 2008). Current visualization systems, where only the correct action of the notional machine is demonstrated, no not make such co-activation explicit.

Diagnose mode

Functionality In this mode, a user can input the expected variables and values for a snippet of code and get feedback on whether their understanding is correct or they may be misunderstanding the way that assignment statements work.

Our tool achieves this by iteratively running each of the misconception models and comparing the results to the stu-

dent's responses, to see if they match any given misconception or for the correct model.

Application Our diagnosis mode shows how our tool creates a theory of mind for the student it is working with, by matching its cognitive model of some given misconception to the cognitive model posed by a student via a code snippet.

Discussion

The ultimate goal of this work is to make both computers and humans better at creating and maintaining a model of one another's knowledge and processing of information. Programming is an inherently creative act, but many students subjectively experience it as mechanical and dull because of the difficulty in getting past initial issues of syntax and understanding the notional machine (Guzdial 2003).

Recent work on nanotutors takes a similar functional approach towards learning complex concepts (Goel and Joyner 2017): identify a concept to be learned, identify possible misconceptions, create an exercise that leads to different answers corresponding to these, create a nanotutor to interpret the various answers and provide feedback accordingly. However, the nantotutor's theory of mind of the student is only implicit in the way that humans designed the exercises, not an attempt to build a meaningful collaboration between the student and software.

We plan to continue building a more complete version of this tool for use in an introductory computing course to determine what impact this approach has on student progress. We hope that this visual and collaborative approach, that offers an opportunity to see exactly how a misconception differs from the correct understanding, will make it easier for beginning programmers to understand and correct their errors and to work with the computer more effectively.

Limitations

Our prototype only works with a very specific set of potential misconceptions, and these are currently concerned only with assignment statements. More work would be needed to identify the specific misconceptions which may present themselves in more complex programming constructs, and to model those in a similar fashion.

The current implementation of our tool doesn't have special handling for the possibility that a single student may possess multiple misconceptions (which could either appear in their attempt to decipher a single piece of code, or over multiple code samples).

Conclusion

Research on collaboration has made it clear that having the ability to use theory of mind to understand a partner or team leads to better results. Humans routinely engage in creative collaboration with computers, but they rarely do so within a framework that allows the computer itself to actively engage in the communication and in building a shared understanding of the task and how commands should be carried out.

This project is only a first small step in the direction of building computer programs that can collaborate with humans to build mutual understanding and agreement as they carry out their tasks. Nevertheless, this paradigm for building the interaction could form the basis of much more complex tools that enable creative collaboration between humans and machines.

References

Baron-Cohen, S.; Wheelwright, S.; Hill, J.; Raste, Y.; and Plumb, I. 2001. The reading the mind in the eyes test revised version: A study with normal adults, and adults with asperger syndrome or high-functioning autism. *Journal of Child Psychology and Psychiatry* 42(2):241–251.

Du Boulay, B. 1986. Some difficulties of learning to program. *Journal of Educational Computing Research* 2(1):57–73.

Engel, D.; Woolley, A. W.; Jing, L. X.; Chabris, C. F.; and Malone, T. W. 2014. Reading the mind in the eyes or reading between the lines? theory of mind predicts collective intelligence equally well online and face-to-face. *PLoS ONE* 9(12):1 – 16.

Goel, A., and Joyner, D. 2017. Using AI to teach AI. *AI Magazine* 38(2).

Guzdial, M. 2003. A media computation course for non-majors. *SIGCSE Bull.* 35(3):104–108.

Lister, R.; Seppälä, O.; Simon, B.; Thomas, L.; Adams, E. S.; Fitzgerald, S.; Fone, W.; Hamer, J.; Lindholm, M.; McCartney, R.; Moström, J. E.; and Sanders, K. 2004. A multi-national study of reading and tracing skills in novice programmers. In *ACM SIGCSE Bulletin*, volume 36, 119–150.

Ma, L. 2007. *Investigating and improving novice programmers' mental models of programming concepts*. Ph.D. Dissertation, University of Strathclyde.

Premack, D., and Woodruff, G. 1978. Does the chimpanzee have a theory of mind? *Behavioral and Brain Sciences* 4(4):515–629.

Sadler, P. M.; Sonnert, G.; Coyle, H. P.; Cook-Smith, N.; and Miller, J. L. 2013. The influence of teachers knowledge on student learning in middle school physical science classrooms. *American Educational Research Journal* 50(5):1020–1049.

Sorva, J. 2013. Notional machines and introductory programming education. *Trans. Comput. Educ.* 13(2):8:1–8:31.

Utting, I.; Tew, A. E.; McCracken, M.; Thomas, L.; Bouvier, D.; Frye, R.; Paterson, J.; Caspersen, M.; Kolikant, Y. B.-D.; Sorva, J.; and Wilusz, T. 2013. A fresh look at novice programmers' performance and their teachers' expectations. In *ITiCSE Working Group Reports*, 15–32.

Van Den Broek, P., and Kendeou, P. 2008. Cognitive processes in comprehension of science texts: the role of co-activation in confronting misconceptions. *Applied Cognitive Psychology* 22(3):335–351.

Woolley, A. W.; Chabris, C. F.; Pentland, A.; Hashmi, N.; and Malone, T. W. 2010. Evidence for a collective intelligence factor in the performance of human groups. *Science* 330(6004):686–688.