Using coding toys to understand equality

Lise E. Welch
Utah State University, USA
<lise.welch@usu.edu>

Jessica F. Shumway
Utah State University, USA
<jessica.shumway@usu.edu>

Jody Clarke-Midura
Utah State University, USA
<jody.clarke@usu.edu>

Victor R. Lee
Stanford University, USA
<vrlee@stanford.edu>

The authors show how a toy robot was used with a number line in a Year 1 class to explore equality and missing addends. They conclude that coding toys can be used to integrate mathematics and programming in engaging and innovative ways, to support young children in their learning.

Robot toys can be used to promote STEM subjects, particularly coding and mathematics. The lesson described in this article illustrates how a robot toy was used with a number line to explore equality and missing addends in Year 1. The lesson progression models how activities can be sequenced to build on developing understandings while supporting the objective of balancing equations and equality. Extension opportunities and adaptation ideas are included to address technology availability and learners' needs at varying levels.

Toy robots can be used in elementary classrooms for STEM activities. These robots and similar programming activities have the potential to deepen content knowledge across traditionally siloed content areas, such as programming and mathematics (Attard, 2012; Clements & Battista, 1989; Fessakis, Gouli, & Mavroudi, 2013). This potential can be realized when pedagogical decisions are carefully considered to leverage the affordances of the toys or tools to enhance the mathematical learning for students (Vatalis, 2018). This article describes how a robot toy, designed to teach coding to young children, was used in a mathematics lesson to support a Year 1 student's understanding of the equal sign and help the student identify missing whole numbers in equations.

The equal sign and solving for unknowns in equations

Understanding the equal sign's function is one of the most fundamental understandings in algebra. It is common for students to misunderstand the equal sign as signaling the result of an arithmetic operation (Fischer et al., 2019). By making sense of the meaning of the equal sign early in mathematics, students can be more successful in their mathematics journey. For example, students will not be able to balance equations or develop a robust understanding of number operations without understanding the meaning and purpose of the equal sign. The primary objectives of the lesson in this article were to reinforce a relational understanding of the equal sign (i.e., = meaning the same value on both sides) and to learn to determine the unknown in simple addition and subtraction problems to balance equations (ACMNA015).

Robot toys as an engaging context for using the number line

There are a growing number of coding toys on the market, such as Primo's *Cubetto*, Fisher Price's *Code-a-pillar*, and Learning Resource's *Botley the Coding Robot* (see Figure 1). One of the most basic of these robot toys is a screen-free robot called Code & Go *Robot Mouse* (see the mouse toy in Figure 1). This toy, produced by Learning Resources, has four arrow buttons on its top that allow the user to program forward, backward, rotate right, and rotate left. The three circular buttons include Go, Clear, and Action/random movement. The robot's forward and backward movements are consistently 13 cm long. *Robot Mouse* is programmed by entering a sequence of directional commands before pressing Go. The robot pauses between commands so children can count

Figure 1. A sample of coding toys. The robot mouse (bottom right) is one of many possible robotics toys that could be used with this lesson.

Figure 2. Top view of Robot Mouse and its buttons.

each movement. These pauses promote one-to-one correspondence of command-to-movement and are ideal for young children as they learn to count, model, and order numbers on a number line (ACMNA013). Other robot toys with directional commands like the *Robot Mouse* could be adapted for this activity.

The following lesson is divided into two activities. Activity 1 is an introduction to programming a *Robot Mouse*. Activity 2 is about using two robots along with a number line to balance an equation and consists of four progressively challenging steps. The computer science (Computer Science Teachers Association, 2017) and Australian Curriculum (Australian Curriculum, Assessment and Reporting Authority, 2015) descriptors referenced within the body of this article are included at the end of the article for reference.

Activity 1: Programming Robot Mouse

Everly's teacher introduced *Robot Mouse* and explained that the robot can be given instructions by pressing its buttons. Everly was introduced to the forward and backward buttons and explored these codes on her own (see Figure 2). The teacher explained that the right circular button needed to be pressed before starting a new program; this 'Clear' button made the robot forget what had been previously programmed (1A-CS-02).

Everly tinkered with the robot by building simple programs to see how far Robot Mouse could travel, experimented with how to make the robot rotate (1A-AP-12), and explored how many movements it took from a starting point to an ending point (1A-AP-14). For example, when she entered the sequence of codes, forward-right-forward-right-forward-right, she saw that the robot moved as if to draw a square.

Activity 2: Using robot mice to balance equations

The following materials were gathered and placed as follows:

- a vertical number line (1–20) with a 13-centimeter space between each number
- two *Robot Mice* positioned parallel to the number line facing in the positive direction with their noses at 0
- two expression cards positioned below the mice with the equal sign between (see Figure 3).

This activity was broken down into steps and scaffolded to build upon Everly's understanding.

Step 1: 7 = __. The equation 7 = __ was placed below the mice (see Figure 3).

"The mouse above the 7 wants to travel seven spaces." Everly's teacher explained. "What number should we put on the right so that the equation is equal and both mice travel an equal distance?"

Everly thought for a moment. "Umm, 8?"

"Alright," her teacher responded, "Let's put an 8 on the other side of the equal sign and see what happens."

The teacher pressed the forward button 7 times for the left mouse and Everly pressed the forward button 8 times for the right mouse. Inputting these codes gave the robot instructions to move forward 7 or 8 movements. "Wait, am I doing 7 or 8?" asked Everly. She seemed unsure about her initial response and was starting to rethink her answer.

"Let's try 8, like you chose, and see what happens," her teacher assured her.

After inputting 8 forwards and a "Ready, set, go!" the Go buttons were pushed to execute the programs and both mice scooted along the number line.

"One, two, three …" Everly sung as she scooted alongside the robots, counting each movement as they traveled. "Six, seven, wait, huh?" Everly watched as one robot stopped at 7 and the other stopped at 8, as shown in Figure 4.

"What happened?" inquired her teacher.

"They're not the same!" giggled Everly.

"They're not the same? Does that mean that 7 and 8 are equal or not equal?" The teacher asked.

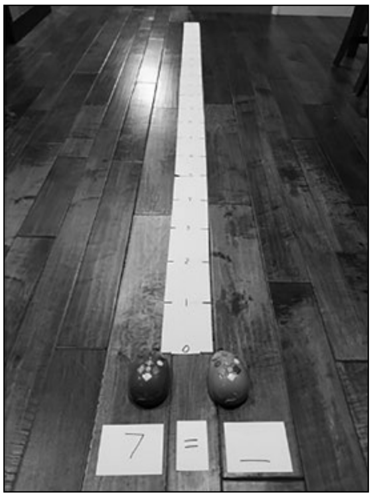

Figure 3. Activity set-up with robots, vertical number line, and cards.

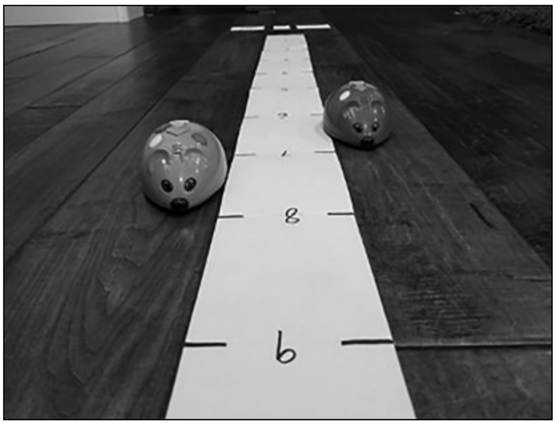

Figure 4. The robots stopped at different places on the number line indicating 7 and 8 are not equal.

"They're not equal! One stopped at the 7 and the other one stopped at the 8. This one," she said, pointing to the mouse on the right, "went one farther than the other one. They're not equal!"

"Hmm." Mused the teacher. "We tried 7 = 8 and that's not equal. Should we change the 8 to something else?" This discussion prompted two related concepts: 1) the concept of equality in mathematics, and 2) the concept of debugging in programming (1A-AP-14). Everly was able to recognize that her mouse's movements did not equal the same amount of movements as the other mouse, and the teacher suggested a change to the equation, which in turn would entail debugging the program in order to make the equations equal.

Everly decided that 7 = 7 was correct because 7 is the same as 7. The mice were reset to the zero line and each was programmed with seven forwards. Everly was elated when both mice stopped at 7. Everly understood how the mice could be used to test if equations are equal.

Step 2: 10 = __ + 4. The equation 10 = __ + 4 was then placed below the mice. "Hmm," Everly murmured, "5, 6, 7, 8, 9, 10," counting on her fingers. "I think it's 6," she stated.

The teacher placed a 6 in the equation and instructed Everly to program the right mouse with 6 plus 4 while the teacher programmed 10 into the left mouse. They then executed their programs and Everly counted as they went along, delighted when the mice simultaneously stopped at the 10.

"Did your choice of 6 make the equation equal?" The teacher asked.

"Yes! They both stopped at the 10, so they are equal."

Reinforcing the integrated programming learning, the teacher exclaimed, "Yes, which means you created a program that got your mouse to the same spot on the number line as my mouse!"

Step 3: 17 = __ + 9. The equation 17 = __ + 9 was positioned below the mice. Numbers larger than ten were more difficult for Everly to calculate. Everly suggested 5, but had difficulty justifying why she thought 5 belongs in the blank. Everly programmed the 5+9 while her teacher programmed 17, then they pressed Go. When the mice stopped in different places, Everly was puzzled.

To build problem solving perseverance, the teacher more explicitly emphasized a computer science process called debugging. She stated, "This means you have the chance to debug this program! Programmers fix bugs in their program if something doesn't work. Let's find our bug. Using a 5 didn't make the equation equal. What number might make it equal instead?"

Everly counted on from 14 where the right mouse left off. "One, two, three. Maybe three?" This is common when students are learning to debug a program (Silvis et al., 2021). They try to solve the problem from where the robot has landed (i.e., debugging from the end) rather than consider the whole program. The teacher moved Everly's mouse back to zero and stated, "If my mouse gets to 17, where is your mouse supposed to land for the equation to be equal?" After Everly replied correctly (17), the teacher continued, "How far do we know the mouse will get?" Everly thought for a moment before pointing to the 9 on the number line. The teacher placed the right mouse on 9 and asked, "How much farther does the right mouse need to travel to equal the distance of the left mouse?" Everly counted on from 9 to 17 and replied that it needed to travel 8 more spaces. The number 8 was placed in the equation and both mice were programmed. Everly hopped with delight when both mice stopped at 17. The teacher and Everly successfully repeated the activity with 19 = __ + 8.

Step 4: 6 = 13 – __. By this point, Everly understood how to make an equation equal and that the robots could be used to check if the two sides were equal. If they were, the mice would end at the same number on the number line. If they were not, she could use the equation with the number line to debug the program. The final equation, 6 = 13 – __, was set in place.

Everly thought aloud, "That mouse (on the left) will end on 6, so the other one has to also." She counted up the number line to 13 then back to 6. "Maybe 7?" Everly reasoned, "If this one (the right mouse) wants to get to 6 too, it needs to go backward seven times."

"We have only used addition so far, how could you program a subtraction problem into Robot Mouse?" queried the teacher.

"I used forward for addition, so I can use backwards for subtraction because subtraction is taking away and if I move backwards then I'm taking away."

Everly programmed the right mouse with 13 forward and 7 backward commands while the teacher programmed the left mouse with 6 forwards before they pressed go. Everly counted the number of backward movements, her voice raising in anticipation. "It worked, they're equal!"

Extensions and adaptations

This lesson could be adapted to explore more complex problems relating four whole numbers rather than just two or three. Additionally, equations that elicit relational thinking, as opposed to solving for one side, are an important possible extension (Carpenter et al., 2003). Table 1 describes these extension possibilities.

Although this lesson was written and enacted for a single student, it could easily be adapted for differing situations (see Table 2).

Conclusion

Integrating mathematics and programming can be an engaging and innovative method to support these two subjects. In particular, using robot coding toys as a context for learning equality concepts and using the meanings of the equal sign to solve for unknowns was beneficial to Everly's journey in making sense of equations. As she solved for unknowns in equations, Everly had a visual representation of equality as she watched her robot's movements compared to her teacher's robot's movements along the number line. She used the robot's movements to check for equality, and when the movements were not equal, she determined there was a bug in the program and the equation. Not only did these activities develop her understanding of equality and solving for unknowns, but they also provided experiences with foundational programming concepts such as the understanding the robot's codes (arrows connected to directional movements), sequencing codes (arrows) to create a program, and debugging (mistakes in the counting of arrows or incorrect number of arrows).

Table 1. Equations for lesson extensions.

Extension	Possible Equations	Justification
A	7 + 5 = 3 + __ __ + 6 = 9 + 5	Two Addition This requires students to consider the equivalent whole of the complete expression to the incomplete expression.
B	10 – 6 = __ – 3 9 – __ = 8 – 2	Two Subtraction This requires students to compare the equivalent difference of the complete expression to the incomplete expression.
C	10 – 4 = __ + 5 7 – __ = 3 + 1	Addition and Subtraction This requires students to attend to each operation as they compare the complete and incomplete expressions.
D	6 + 7 = __ + 6 __ + 2 = 5 + 2 8 + __ = __ + 2 9 + 1 = 8 + __ 5 + __ = 3 + 2 + 5	This extension fosters opportunities for relational thinking so that students do not over generalise the meaning of the equal sign as an "answer."

Table 2. Ideas for adapting this lesson.

Small-group instruction	Students may work in small groups to brainstorm solutions. Each student may be assigned a role to encourage participation. For example, one student may be the button pusher, one can justify the group's answer using manipulatives, one can justify the group's answer by drawing a picture, and one student may set up the problem and reset the robots.
Whole-class instruction	The class sits around the set-up with drawing materials and small manipulatives. The activity is discussed as a whole class and students work as partners to determine a solution and prove it using manipulatives and/or paper and pencil. The teacher then chooses pairs to describe their thinking and enact it with the robots.
Without a robot	The teacher could act out the 'robots' using two figurines on a number line under a document camera. The activities could be modelled by the teacher or students playing the parts of moving the figurines.

Curriculum content standards in this article

Year 1 Australian Curriculum descriptors

- ACMNA013 Recognise, model, write, and order numbers to at least 100. Locate these numbers on a number line.
- ACMNA015 Represent and solve simple addition and subtraction problems using a range of strategies including counting on, partitioning and rearranging parts.

Computer science standards

- 1A-AP-11 Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.
- 1A-AP-12 Develop plans that describe a program's sequence of events, goals, and expected outcomes.
- 1A-AP-14 Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.
- 1A-CS-02 Use appropriate terminology in identifying and describing the function of common physical components of computing systems (hardware).

References

Silvis, D., Clarke-Midura, J., & Shumway, J. (2021, June 8). *Objects to debug with: How young children resolve errors with tangible coding toys.* Annual Meeting of International Society of Learning Sciences, Bochum, Germany.

Attard, C. (2012). Teaching with technology: Exploring the use of robotics to teach mathematics. *Australian Primary Mathematics Classroom*, *17*(2), 31–32.

Australian Curriculum, Assessment and Reporting Authority. (2015). *Australian Curriculum: Mathematics*. https://www.australiancurriculum.edu.au

Carpenter, T. P., Franke, M. L., & Levi, L. (2003). *Thinking mathematically: Integrating arithmetic & algebra in elementary school.* Portsmouth, NH: Heinemann.

Clements, D. H., & Battista, M. T. (1989). Learning of geometric concepts in a Logo environment. *Journal for Research in Mathematics Education, 20*(5), 450–467.

Computer Science Teachers Association and the Association for Computing Machinery. (2017). CSTA K-12 Computer Science Standards, Revised 2017. http://www.csteachers.org/standards

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 year old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87–97.

Fischer, J.-P., Sander, E., Sensevy, G., Vilette, B., & Richard, J.-F. (2019). Can young students understand the mathematical concept of equality? A whole-year arithmetic teaching experiment in second grade. *European Journal of Psychology of Education*, *34*(2), 439–456.

Highfield, K. (2010). Robotic toys as a catalyst for mathematical problem solving. *Australian Primary Mathematics Classroom, 15*(2), 22–27.

Vatalis, E. (2018). I can locate it! Teaching location with the assistance of digital technologies. *Australian Primary Mathematics Classroom, 23*(3), 8–11.