Blending Everyday Movement and Representational Infrastructure: An Interaction Analysis of Kindergarteners Coding Robot Routes

Deborah Silvis, Utah State University, deborah.silvis@usu.edu
Victor R. Lee, Stanford University, vrlee@stanford.edu

Jody Clarke-Midura, Jessica Shumway, Joseph Kozlowski
jody.clarke@usu.edu, jessica.shumway@usu.edu, jkozlowski@aggiemail.usu.edu
Utah State University

Abstract: If we want to design for computer science learning in K12 education, then Kindergarten is the place to start. Despite differing formats, early childhood coding tools rely heavily on a similar representational infrastructure. Grids and arrows predominate, introducing a nested series of directional symbols and spatial skills children must learn in order to code. Thus, learning to think computationally is entangled with learning to think spatially and symbolically. This paper analyzes video of Kindergarten students learning to use robot coding toys and examines how they navigated programming's representational infrastructure. We found that children drew on conventional notions of how objects move, creating a "conceptual blend" for programming robot routes (Fauconnier & Turner, 1998). We argue that coding in Kindergarten requires mapping a series of correspondences from the domain of everyday movements onto the resources available in the representational domain.

Introduction

Kindergarten is traditionally a time when playing while learning is not only permitted, it is planned. It is not surprising then, that designs for early computer science education increasingly involve entry points like coding robot toys (Bers, 2018). When it comes to early robotics, there is a growing number of commercially available toys that educators can use to introduce young children to computational ideas in a playful way (Hamilton, Clarke-Midura, Shumway & Lee, 2019). Many of these operate like the turtle from the LOGO programming language and are manipulated using forward and backward and right and left turn commands. The tangible components, cute construction, and sounds and lights embedded in these toys are engaging, and children have many intuitive ways of using them to find their way into coding. Things become more complicated when these same toys are deployed in tasks and embedded in a curriculum. The demands of basic robot coding tasks, coupled with the toys' design constraints, bely the simplicity of such toys. This paper examines what happens when Kindergarteners first learned to code a robot's routes and asks how deceptively simple concepts get structured by the material ecology of coding robots.

To begin, we observe that young children have had many experiences with objects moving around in everyday life. They see people walk, run, and jump. They observe other living things like animals or insects moving at various speeds, along curvilinear paths, and through the air. They sit (sometimes for many hours) in moving vehicles; or while stopped in traffic, they may observe other vehicles driving by or turning a corner. Prior to coding robot toys, children draw on a series of conventions that pertain to mobility of objects in the human world. The first is that many common objects appear to move continuously, not incrementally. Although people technically take individual steps as they walk a path and cars stop at red lights, while in motion, these trajectories are experienced as fluid movement, not repeated stops and starts. However, most robot coding toys traverse a straight gridded path one space at a time.

A second everyday motion convention is that when people are instructed to turn in some direction, this often can signify a single move of rotation-plus-translation, as in the situation of rounding a corner. However, the turn arrow in robots' programming language signifies just a rotation in place- either left or right; translation or displacement to an adjacent square on the grid requires an additional forward arrow command. Third, when navigating between two points, people and things are often capable of a range of headings from 0-360 degrees, creating "shortest distance" diagonal pathways from the point of origin to the destination. Children's everyday environments like classrooms and playgrounds are experienced as "open" spaces and allow for curvilinear paths, whereas robot toys used for learning coding are often confined to a gridded mat that circumscribes their wayfinding. Robot toys' turn capabilities may only be limited to 90 degree increments, and their fixed wheel axels prohibit lateral movement or "side-stepping."

Children draw from two sometimes conflicting domains to learn how robots move: the domain of everyday movement and the representational system of the coding environment. Programming a robot route means creating

a structure for the conventions of robot movements that is based in experiences of everyday movement but that conforms to the symbolic order of the coding task. In what follows, we present two examples of cross-domain mapping that took place when children coded robot routes. These two cases are not meant to exhaust the possible spatial dynamics at work in these learning configurations, but rather to illustrate how children blended inputs from their own experience of linear motion and turning with the inputs in the symbolic domain when they worked to code a robot to navigate to some point on a grid. We argue that coding occurred in a blended space, where running the program required running the blend. We further suggest that this blend then generated new understandings about the representational system itself.

Theoretical framework

Conceptual Blending theory explains how new conceptual structures are formed by projecting elements from separate, sometimes disparate domains, or mental spaces (Faucconier & Turner, 1998). Conceptual domains constitute mental spaces insofar as they structure conventional meanings that can be projected or mapped across reference frames. A classic example of conceptual blending is trashcan basketball, a game in which people construe their trashcan as a basketball hoop and a balled-up piece of paper as a basketball as they toss the "ball" into the "basket" (Coulson, 1999). The enactment of trashcan basketball is a blend from two separate domains or input spaces: the domain of throwing away trash and the domain of playing basketball. By mapping elements like "ball" onto "trash," "hoop" onto "can, "take a shot" onto "throw away" (as well as other salient characteristics shared by the two counterpart domains), trashcan basketball is born. According to Fauconnier and Turner, who first described concept formation in terms of blending, blends are creative, generative, and yield new ways of thinking not possible in either source domain. Through conceptual integration, blends can also be joined to one another in networks. When a blend operates to simulate new action or generate further computation, we are said to "run the blend."

Blending theory has been particularly attuned to how people develop new concepts for spatial dynamics, action, and motion, as canonical examples like "trashcan basketball" and "the 1853 regatta" attest (Coulson & Fauconnier, 1999; Fauconnier & Turner, 1998). In the regatta example, a catamaran in 1993 called the *Great American II* was set to outpace the record of a clipper ship in 1853 called the *Northern Light* that had run the same course. Stating that "at this point, *Great American II* is 4.5 days ahead of *Northern Light"*, sets up a "cross-space mapping that links the two trajectories, the two boats, the two time periods, positions on the course, and so on" (Fauconnier & Turner, 2002, p. 63). Projecting elements from one journey onto elements from the other yields a conceptual blend for a historic competition between the two boats. A number of other felicitous examples in the blending theory literature informed the present analysis. We drew on canonical blends like the "computer virus", to map a correspondence between machine and biological processes (Coulson, 1995); the "skiing waiter" to combine different styles of motion (Fauconnier & Turner, 2002); the "riddle of the Buddhist monk" to resolve spatiotemporal discontinuities in path-taking (Fauconnier & Turner, 1998); the "ship of state" to describe navigation through abstract representation (Grady, Oakey & Coulson, 1997); and "queueing" as a mix of linear space and forward trajectory (Hutchins, 2002). These very different examples speak to a consistent conceptual phenomenon—blending—that we believe is at work when learning to program robot routes. We elaborate a blend for "programming robot routes" to explain how children brought their knowledge of how things move around in the everyday world into correspondence with a symbolic system of arrows and gridded space.

Canonical blending examples like these mask the mundane quality of most blends. Blending concepts from one domain with those of another is a ubiquitous mechanism that explains language development, perception, and human ritual (Fauconnier & Turner, 2002). For example, the common language form for a Y-expression, where "X is the Y of Z" (i.e. the boss of the company, or the father of Sally) prompts for blending, projecting conventional relationships between X and Y into a blended space where the meaning of Z emerges and gets elaborated. It is therefore unsurprising that everyday activities—like using symbols to give directions—should also involve blending. Routine as our examples from Kindergarten classrooms may be, there is something novel about the environment in which they are situated. Programming robots using directional syntax provides a new configuration of conceptual spaces for children to learn to navigate. While navigating itself may be phylogenetically old hat for humans, "the child is on the path of both acquiring networks that lie behind what its culture offers and developing new ones through blending of inputs" (Fauconnier & Turner, 2002, p. 216).

Following recent empirical work with young children using learning technologies to support classroom learning, we see conceptual blending as a socially constructed means of young children simulating complex computational ideas through public interactions (Enyedy, Danish & DeLiema, 2015). We build from this perspective to suggest that, while blending theory incorporates young children's conceptual development, *part of what is at stake in any blend is the integrity of source inputs themselves*, and for young children these are still being stabilized. What Hutchins (2005) called "material anchors" for conceptual blends are sometimes assumed

to be stable sources of meaning, easily accessed and projected to the blend. However, meaning-making can come unmoored when the source input itself is still under construction. In what follows, we show how a particular blend for navigational programming is both a conceptual achievement for Kindergarten coders and resources for further meaning making.

Study design and data analysis

Primary methods used in this study were design-based research (Brown, 1992) and participant observation. The teaching and learning described in this paper took place as part of a larger study, which is set in the Rocky Mountain West region of the US and was designed to investigate young children's computational thinking with coding toys. In the larger project, we are conducting design studies where we pilot curricular tasks with small groups of three to five children in kindergarten classrooms. All lessons are taught by the research team, video recorded, and analyzed through design memos and discussions that lead to re-design of the lessons. In this paper, we focus on a single classroom and analyze what took place when 15 children, spread across four groups, learned to program a specific robot called BeeBot (https://www.terrapinlogo.com/beebot.html).

Through iterative, daily adaptations of structured coding tasks, we aimed to teach children how to program robots like BeeBot to follow simple paths. These tasks were typically framed as stories, where the robot needed to go on some adventure, travel to a series of locations, avoid obstacles placed in its path, or learn to do some special "move" along the way. For example, in the introductory lesson, children must program BeeBot to stop at the pond for a drink on the way home to the hive. In these coding tasks, the material ecology included a large floor grid, a collection of arrow cards that serve as codes or commands, a template for organizing these codes to build a program that we call a "program organizer", and a mechanical robot (Figure 1). The program organizer and arrow cards were optional designs integrated into the "representational infrastructure" to support children organizing and recalling sequences of code (Hall, Stevens & Torralba, 2002).

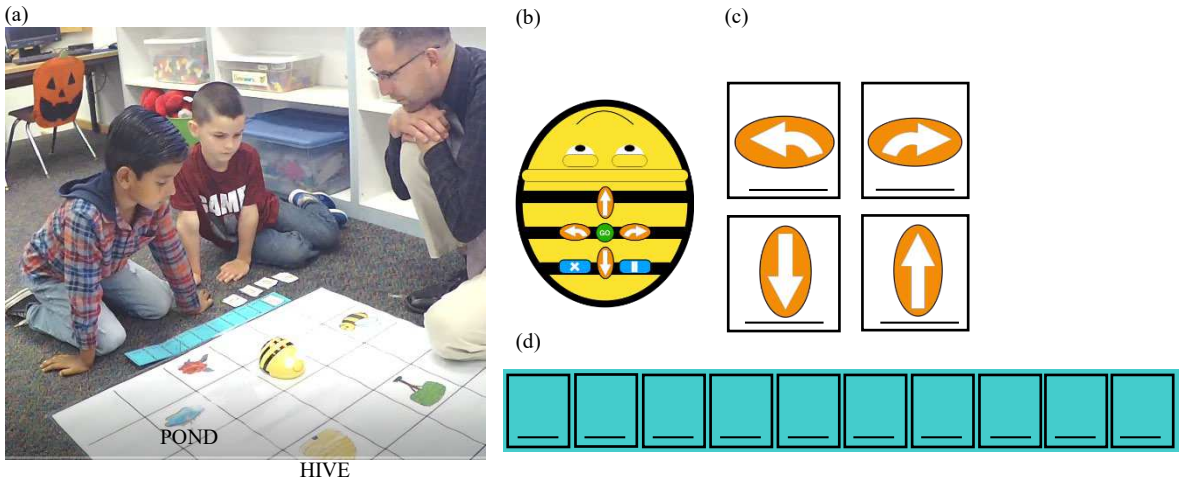

Figure 1. The material ecology of the BeeBot tasks. *(clockwise from left)* (a) the task configuration (b) BeeBot's buttons (c) directional arrow cards/codes (d) program organizer.

All of these lessons were video recorded. Four hours of data were first reviewed and content logged (Jordan & Henderson, 1995). Then rounds of open coding and three group video viewing sessions were conducted by our research team. Themes and categories emerged related to how children were learning to program robot routes, which corresponded with a number of dimensions of robot movement: conventions for *linear motion*; *turning*, *heading*, *topography*, *speed*, and *journeys*. We focus the following analysis on separate instances of how children learned conventions for two of these dimensions: linear motion and turning. Linear motion and turning are critical elements of robot movements that children needed to learn, and the two instances represent typical ways in which children in the four groups tended to go about this learning.

We conducted interaction analysis of the talk-in-interaction that took place during BeeBot lessons in two different groups of children (Jordan & Henderson, 1995). Specifically, we took "the person-in-interaction-with technology" (Hutchins, 1995, p. 155) as the unit of analysis across eight separate lessons (two per group). Iteratively re-viewing video data, we asked analytic questions regarding the use of the robot, people's talk about the direction arrows and grid, and the dynamics of these interactions as they temporally unfolded across the task space (Goodwin, 2018). As we analyzed key moments, we iteratively generated multimodal transcripts, ultimately producing figures that mapped onto the analysis and were also mappings of it (Figure 3,4). In other words, we

imagined the grid space as both a shared topography for the organization of real time activity and retrospective analysis. These diagrams display how programming robot routes was an achievement built of talk, body position, gestures, gaze, and (touching) materials ordered by a representational system that drew its meaning from children' everyday experiences of movement.

Analytic findings

We found that children' understanding of everyday movement conventions prefigured their use of the representational system as a resource for learning to program a robot. Programming took place in a blended space that drew on elements of these two domains: the mental space of everyday movement conventions and the task's representational system. On the one hand, objects in the world have particular ways of moving along paths; people plan routes based on these conventions. On the other hand, robot movements operate according to a different set of parameters organized by a set of representations which must be learned. Programming involved blending these two inputs by selectively projecting relevant elements of the two domains and "running the blend" to generate more robot routes in future tasks. Here, programming means planning and inputting instructions or arrow commands that BeeBot was to follow along a route to some pre-specified grid location. Elaborating their new concept for programming robot routes also enabled children to project meaning back to the inputs, imbuing the representational domain with new significance and "anchoring" the available materials.

We identified a number of key domain elements shared by everyday movements and by the representational system that were involved in children learning how to program robot routes: *linear motion*, *turning*, *heading*, and *topography* as well as *speed* and an element we call "*journeying*" which has to do with how long trips are broken up into legs or stops along the way. We elaborate how two of these—linear motion and turning—contributed to the programming blend. Figure 2 illustrates how the blend is composed of selective projections of characteristic elements from the domain of everyday movement and their counterparts in the tasks' representational domain. Whereas in the everyday domain, linear motion is experienced as continuous and represented by mundane instances of people walking around or vehicles driving, in the representational domain, motion is experienced as incremental and ordered by a one-to-one correspondence between arrows, buttons, and movements. And whereas in the everyday domain, turning is experienced as a simultaneous rotation and translation and represented by common phenomena like birds' arcing flight paths or a car "rounding a corner," in the representational domain, turning is ordered by a rotation in a grid space and a separate translation to an adjacent space and represented by arrow symbols. As it turns out, ambiguity of the symbol system itself is part of what needed to be resolved to run the blend and to run the program. Next, we analyze what this looked like in two cases that illustrate blends for programming routes that involved linear, forward motion and turning.

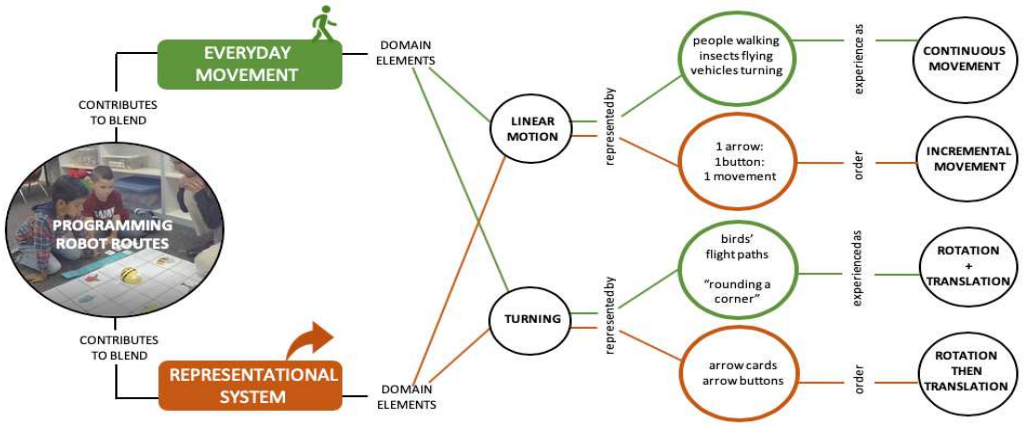

Figure 2. Conceptual blend for programming robot routes, highlighting two shared domain elements that get mapped to the blend: linear motion and turning.

From continuous to incremental motion

A good portion of the first BeeBot lesson was spent learning what each of the four directional buttons makes the robot do (see Figure 1b). What might be taken as a "straightforward" introduction to symbols and actions is complicated by Kindergarten children's divergent understandings of what "straight" possibly means and how "forward" gets represented by a robot and a bunch of arrows. In fact, children had a range of ways of describing the command that made BeeBot move forward. Some called it a "forward" or a "straight." For others, it signified "up" or it made it "keep going." That a single forward arrow called the robot to make a single movement forward

across one grid unit was something that needed to be learned, and learning this convention meant blending inputs from the symbolic resources of the task (i.e. the arrow buttons, grid, codes, and program organizer) with the domain of everyday movements.

When their coding teacher Mr. Koz initially asked Cory and Eli to predict the result of programming one forward arrow into BeeBot, Cory told him it would go to the strawberry image three spaces forward and one to the left, while Eli guessed it would land on a square adjacent to the strawberry (Figure 3ab, Route 1). Their initial concept for programming a forward arrow drew heavily from the everyday movement domain (i.e. continuous motion, landmarks as obstacles, etc.). Salient features of the representational domain, such as the strawberry, were mapped onto these everyday conceptual elements and selectively projected to Cory's blended space. For Eli, the grid lines rather than grid images, weighed on his estimate of forward motion (Figure 3ab, blends). When they ran the program, they saw that the BeeBot went forward one space. On their second test of the forward arrow, Mr. Koz again asked them for their predictions. This time, Cory, no longer recruiting conventions for destinations but still projecting continuous motion into the blend, modified his estimate, touching a square only two spaces beyond BeeBot's starting position. Eli agreed with Cory (Figure 3ab, Route 2). They then ran the program, and Eli conceded "I guess it only goes one." Mr. Koz finally solicited a third prediction, and Eli pointed to the space directly in front of BeeBot. However, Cory stuck to his prior predictions and pointed to a square two spaces ahead (Figure 3ab, Route 3). After running the program a final time, Mr. Koz asked Eli how he knew it would only go one, and Eli concluded "because it always does."

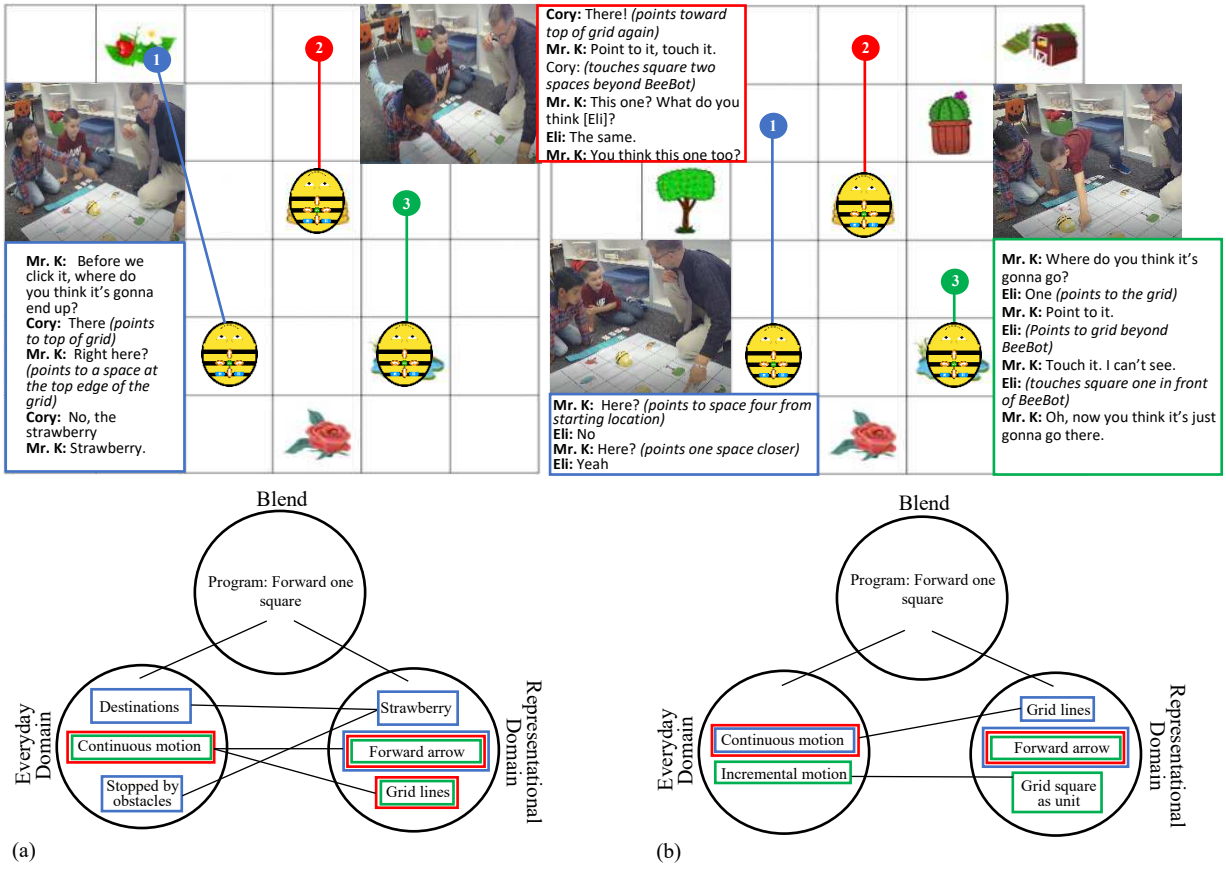

Figure 3. (a) Cory's predictions for programming one forward arrow on three trials *(top)* and selective projection of domain elements to the blend *(bottom)*. (b) Eli's predictions for programming one forward arrow on three trials *(top)* and selective projection of domain elements to the blend *(bottom)*.

Learning what a robot "always" does given a particular instruction is part and parcel of learning to code, because algorithms are essentially conventions for programming action (Bers, 2018). Like language or any other social and material system of meaning, programming conventions are not inherently meaningful but require considerable conceptual effort and time. Cory remained unconvinced by repeatedly running the program and continued to believe that BeeBot could progress according to continuous, rather than incremental motion.

Although Cory had begun in the second trial to project information about the significance of grid lines to the programming blend, Eli was eventually recruiting another set of resources, projecting the symbolic convention "one arrow: one grid square" from the representational system available in the task. For him at least, the representational domain now contributed new meaning to the programming blend, meaning which itself got constructed through repeatedly running the program.

Making (sense of) turns

One of the most perplexing parts of learning to code robots is figuring out how they turn. Robots' lateral movement requires a turn command (causing it to rotate in place) and then a forward command (to translate to an adjacent square). Only then will it leave it's starting position and turn into a square to the left or right on its path. However, based on knowledge of how things move around in the everyday domain, children intuitively believe that when they want a robot to make a left or right turn to an adjacent square, it needs only a single turn arrow. The convention of "curving turns" that applies to vehicles, airborne insects and birds, and our own embodied experience of "making a turn" are deeply entrenched; many children in our Kindergarten research sites persist in this belief even after multiple experiences coding different robots over periods of weeks.

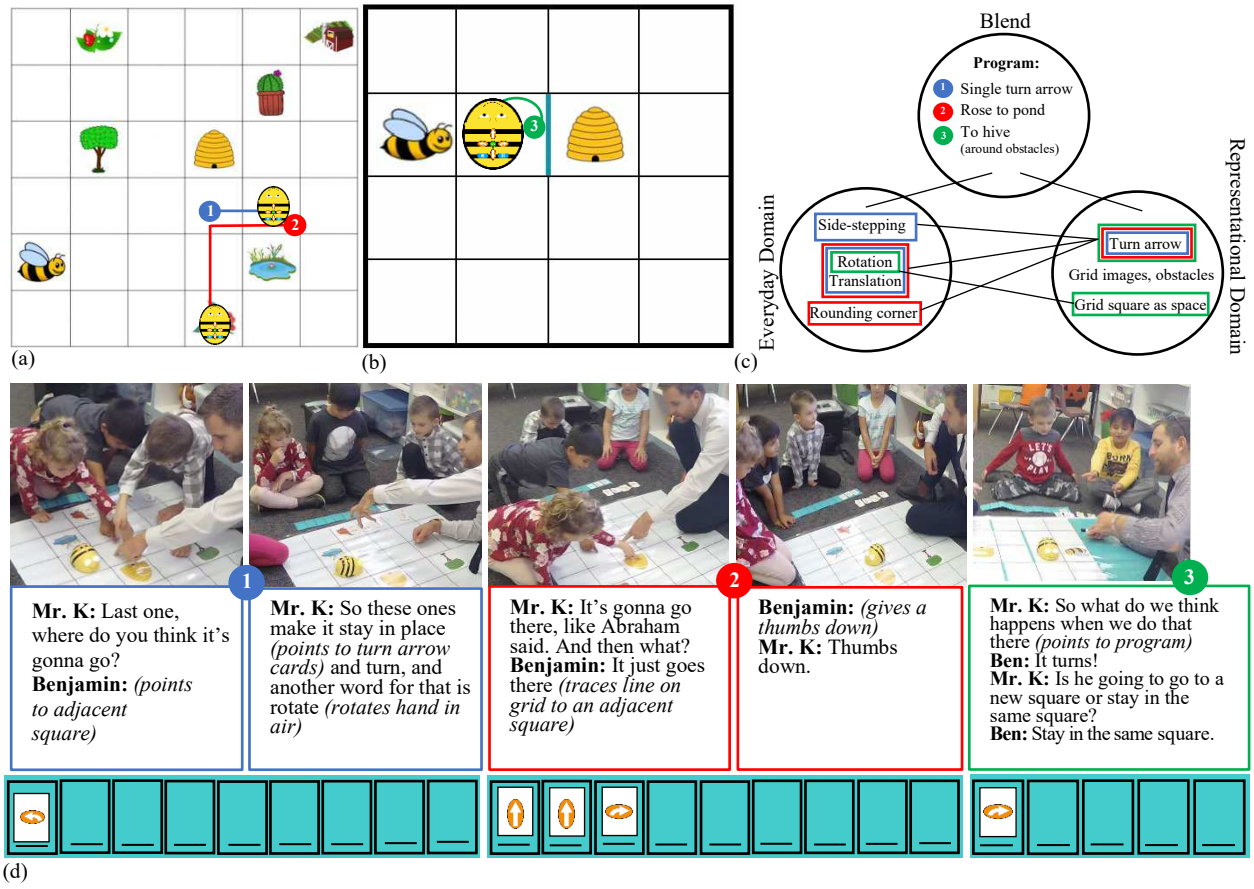

Figure 4. (a,b) map of Benjamin's predictions for where a turn arrow will make BeeBot move on two days and three programs (c) selective projection of elements for "turning" to the blend. (d) Benjamin's gestures and talk during programs 1, 2, and 3.

When Mr. Koz first introduced Benjamin (wearing a black t-shirt in the photos in Figure 4) to a single left turn arrow and asked him to predict what it made BeeBot do, Benjamin and his group members predicted it would end in the square to BeeBot's left (Figure 4d1). In conceptualizing a robot turn, Benjamin mapped from the everyday input domain possibilities like "side-stepping" and selectively projected these movement experiences to the blend. Later in the same lesson, Benjamin began to recruit new inputs from the symbolic domain of the grid as they ran repeated programs. While the group planned a program to get BeeBot to the pond (which they had already overshot by one grid space), Mr. Koz asked them where they thought the right turn arrow would

make BeeBot move next. Benjamin again pointed to the square just to the right of BeeBot, despite having just witnessed it simply rotate in place when given a turn command. After watching the program run a second time, Benjamin looked up at Mr. Koz and gave him a thumbs down, meaning his prediction had proven incorrect. A new convention for robot movement—that robots "stay" and rotate in place before moving to another square—was projected back to the source domain, which was beginning to incorporate meanings for the grid space into the program blend. At the same time, the representational domain that included turn arrows and grid squares was being granted new meaning, stabilizing these symbols as material anchors for future blends.

We saw evidence of this the next day, when, using a slightly modified grid, with a smaller area and a blue "fence" that blocked direct access to the hive (Figure 4b), Benjamin revised his predictions regarding turn arrows. When Mr. Koz asked them to predict what would happen when they made a turn, pointing to a right turn arrow they had placed at the start of the program organizer, Benjamin exclaimed that "it turns." When Mr. Koz then qualified his question, asking if BeeBot would go to a new square or stay in the same square, Benjamin confidently told him it would "stay in the same square" (Figure 4d3). Running repeated turn-programs had projected back to the input domain for representations a new meaning for the turn arrow. It was no longer a sign for robots to rotate *and* translate simultaneously, as in his predictions for turning in the previous day's programming. Benjamin's concept for turning now isolated these movements, and a grid square delineated a space to stay and rotate. The representational domain space now selectively projected this new structure to the blend, a working program for turns that could be run again and again (Figure 4c).

Discussion

As young children project onto the representational domain space their conventional concepts of how things move, the stability of the material anchors is tested because robots do not move as we do. When they try to map their everyday concept of motion onto the learning materials, breakdowns in the conceptual model occur that must be restructured to incorporate conventions of robot movements. It is only by crystalizing the symbolic relationship between direction arrows, grid space, and robot trajectories, that workable conventions for robot movements get established, and a strong mapping occurs. But rather than a straightforward representational problem—whereby students simply map a symbolic set of meanings onto available materials—before starting to code, they are already accessing known conventions for movement as resources to program robot routes.

While their everyday experiences of linear motion and turning (as well as heading, topography, speed, and journeying) were experienced in one set of ways, the symbolic system of the programming language imported a different set of relations into the blend. Symbols like arrows and squares on a grid order a series of relations not typically associated with everyday movement. In order to program robot routes, children selectively projected some of these relations and not others, creating a conceptual space (a blend) for robot routes that conformed to the robot's constraints while structuring a new set of meanings related to the arrows and the grid. It was not only the case that this new blend constituted a new mental space for future action; the blended space of robot routes also strengthened the input space for symbolic meanings, at least locally. Whereas before running the blend, the meanings of arrows and grids were weak compared to the everyday conceptual counterparts in the domain of everyday movements, the blend organized a set of meanings for the representations, which could then serve as material anchors for elaborating new programs. In this case, running the program meant running the blend.

While there is ample evidence that even children older than age five may struggle to make accurate directional decisions when solving spatial problems (Clements et al., 1996; Gauvain & Rogoff, 1989), we believe that this is not due to spatial reasoning alone. In the types of tasks described here—and with many other robots like BeeBot—the programming language itself is entangled with the solving a spatial problem. In other words, because the syntax is an arrow-based system, children are not only learning how robots move but also what the commands themselves mean. In order to produce and manipulate a stable representation of the conceptual elements ordering robot routes, the elements must be somehow held (or anchored) in place. According to Hutchins (2005), this holding-in-place is accomplished by mapping the conceptual elements onto a relatively stable material structure, in this case, the grid space and arrow commands. However, the conventional meanings of grids and arrows import a series of correspondences into the task that are not yet stably structured for many Kindergarteners. That is, the representational system of grids and arrows is still new to children.

Resolving the meanings of arrows, grid space, and sequences of codes was necessary to program BeeBot, yet watching programs run was necessary to learn what the symbols meant and how the task space worked to structure the routes. This bi-directional projection of meaning—from the input to the blend and then back to the input—is a regular aspect of blending that has received little attention. The very notion of an "anchor" for a material resource connotes stability. However, for young children, it is the anchoring of material resources that is partly at stake in conceptual change. Composing and running the blend was one moment in this process, and

stabilizing the input spaces as systems of meaning was another. Both were necessary and mutually contingent for making sense of programming robots.

Conclusions and implications

Computer programming appears to happen across these blended spaces, where ambiguous reference and incongruous motion conventions must be resolved in order to code a robot's route. In the case of Beebot, coding did not simply require representing movement using a program organizer, the arrow codes and buttons, or the grid space. Coding involves a blend of the mental space of conventional movement with the representational system of robot movement. We think there are several implications of construing coding as conceptual blending.

For one, it acknowledges the messiness of a skill like coding, not only because some coders are young children just starting to systematically organize their computational knowledge and actions, but also because of the complexity of all programming languages. Programmers must continuously resolve one set of inputs with another, must wrangle meaning out of novel algorithms, or reconcile competing parameters (diSessa, 2000). In this sense, Kindergarteners' programming is not far from what computer scientists do all the time. Second, a blend for robot routes points to how conceptual spaces both feed into blended spaces and can benefit from running blends. Third, conceptual blends in this context require material anchors, however the meanings being constructed are precisely what is at stake in early childhood, emphasizing how human development is implicated in conceptual development.

In our designs for collaborative CS tasks, we should attend to how individual and groups of learners are making sense of the complex blend of computational domains, in particular how children's roles and tools like robots are distributed across the blended spaces (Enyedy et al., 2015). Finally, as robotics picks up steam in early childhood education, we will need to pay even closer attention to how the human and nonhuman domains interact in practice to enact computational thinking. As robots become more integrated into daily life, their actions will become conventionalized, transforming everyday notions of how things move and what mechanical objects can do. In this emerging context, programming and directing robot actions constitutes a new, evolving learning space that Kindergarteners are already navigating.

References

Bers, M.U. (2018). *Coding as a playground: Programming and computational thinking in the early childhood classroom*. New York, NY: Routledge.

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences, 2*(2), 141-178.

Clements, D.H., Battista, M.T., Sarama, J. & Swaminathan, S. (1996). Development of turn and turn measurement concepts in a computer-based instructional unit. *Educational Studies in Mathematics, 3*0(4), 313-337.

Coulson, S. & Faucconier, G. (1999). Fake guns and stone lions: Conceptual blending and privative adjectives. In B. Fox, D. Jurafsky, & L. Michaelis (Eds.), *Cognition and Function in Language*. Palo Alto: CSLI.

diSessa, A. (2000). *Changing minds: Computers, learning, and literacy*. Cambridge, MA: MIT Press.

Enyedy, N., Danish, J.A. & DeLiema, D. (2015). Constructing liminal blends in a collaborative augmented-reality learning environment. *International Journal of Computer-supported Collaborative Learning, 10*:7-34.

Fauconnier, G. & Turner, M. (1998). Conceptual integration networks. *Cognitive Science, 22*(2), 133-187.

Fauconnier, G. & Turner M. (2002). *How we think*. New York, NY: Perseus Books.

Gauvain, M. & Rogoff, B. (1989). Collaborative problem-solving and children's planning skills. *Developmental Psychology, 25*:1, 139-151.

Goodwin, C. (2018). *Co-operative action*. New York, NY: Cambridge University Press.

Grady, J.E., Oakley, T., Coulson, S. (1997). Blending as metaphor. In G. Steen & R. Gibbs (Eds.), *Metaphor in Cognitive Linguistics*. Philadelphia, PA: John Benjamins.

Hall, R., Stevens, R. & Torralba, T. (2002). Disrupting representational infrastructure in conversations across disciplines. *Mind, Culture and Activity, 9*:3, 179-210.

Hamilton, M., Clarke-Midura, J., Shumway, J. & Lee, V.R. (2019). An emerging technology report on computational toys in early childhood. *Technology, Knowledge, and Learning, 25*, 213-224.

Hutchins, E. (1995). *Cognition in the wild*. Cambridge, MA: MIT Press.

Hutchins, E. (2005). Material anchors for conceptual blends. *Journal of Pragmatics, 37*, 1555-1577.

Jordan, B. & Henderson, A. (1995). Interaction analysis: Foundations and practice. *The Journal of the Learning Sciences, 4*(1), 39-103.