Early Childhood Research Quarterly 65 (2023) 68–80

Contents lists available at ScienceDirect

Early Childhood Research Quarterly

journal homepage: www.elsevier.com/locate/ecresq

Designing formative assessments of early childhood computational thinking

Jody Clarke-Midura[a,*], Victor R. Lee[b], Jessica F. Shumway[c], Deborah Silvis[a], Joseph S. Kozlowski[c], Rebecca Peterson[c]

[a] Instructional Technology and Learning Sciences, Utah State University, 2830 Old Main Hill, Logan, UT 84342, United States
[b] Graduate School of Teacher Education, Stanford University, Stanford, United States
[c] Department of Teacher Education and Leadership, Utah State University, Logan, United States

ARTICLE INFO

Keywords:
Formative assessment
Computational thinking
Kindergarten
Tangible coding toys
Assessment design

ABSTRACT

With growing interest in supporting the development of computational thinking (CT) in early childhood, there is also need for new assessments that serve multiple purposes and uses. In particular, there is a need to understand the design of formative assessments that can be used *during* classroom instruction to provide feedback to teachers and children in real-time. In this paper, we report on an empirical study and advance a new unit of observational analysis for formative assessment that we call an *indicator of a knowledge refinement* opportunity or as a shorthand, *KRO indicators*. We put forth a new framework for conceptualizing the design of formative assessments that builds on the Evidence Centered Design framework but centers identification and analysis of indicators of knowledge refinement opportunities. We illustrate a number of key indicators through empirical examples drawn from video recordings of Kindergarten classroom lessons.

1. Introduction

With recent and growing interest in supporting the development of computational thinking (CT) in early childhood through design of new curricula, tools, and standards, there is also need for a variety of assessments that serve multiple purposes and uses. In the present study, we define an assessment as a procedure for making inferences about student learning (e.g., Black & Wiliam, 2018). Central to the design of assessments are questions around what knowledge and skills are being assessed, what inference should be made (e.g., about an individual's performance), what students should be capable of doing and in what kinds of situations, and how we recognize and evaluate salient aspects of their performance (Oliveri, Lawless & Mislevy, 2019). While some work is underway to develop assessments for early childhood CT (e.g., Clarke-Midura, Silvis, Shumway, Lee, & Kozlowski, 2021; Govind & Bers, 2021; Relkin, de Ruiter & Bers, 2020), the emphasis thus far has been on assessments that serve a summative purpose, specifically a summary of what children know, understand, and can do either *before* or *after* the learning has taken place. There also needs to be development of formative assessments intended for use *during* instruction that provide feedback to teachers and children for the purpose of adapting instruction and improving learning. Ideally, formative and summative assessments would both be essential elements of a larger "system" of assessments that altogether provides a more thorough picture of children's learning on multiple time scales and for multiple stakeholders (Herman, 2010).

In this paper, we advance a new unit of observational analysis for the design of formative assessments that we call an 'indicator of a knowledge refinement opportunity,' or as a shorthand, KRO indicators. KRO Indicators are based on a constructivist view of learning, which requires seeing errors in children's reasoning (relative to normative expectations) as involving elements of knowledge that are ripe for refinement rather than replacement (Smith, Disessa & Roschelle, 1994). When a child demonstrates an error, the asset-based perspective is that some aspects of the child's thinking are already correct but there are still other aspects that need more tuning. This stance is highly compatible with the goals of formative assessment design, which is intended to generate feedback and actionable information about how a child is progressing in their learning so that modifications to instruction can be made in a timely manner.

In the sections that follow, we first share our operational definition of CT for early childhood. We then describe the work currently being done on assessing CT in early childhood and why we are focusing on the design of formative assessments. We then introduce the design framework we are using to develop our assessments, Evidence-Centered Design (ECD). After describing ECD, we introduce and define 'indicators of knowledge refinement opportunities' and how we are integrating these indicators with the ECD framework to develop formative assessment tasks. We then discuss our methods and materials, including participants, data, and data analysis. Finally, we present our results followed by our discussion and conclusion.

* Corresponding author.
E-mail address: jody.clarke@usu.edu (J. Clarke-Midura).

https://doi.org/10.1016/j.ecresq.2023.05.013
Received 7 January 2022; Received in revised form 11 May 2023; Accepted 30 May 2023
Available online 16 June 2023

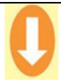
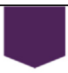
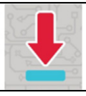
Forward				
Rotate Right (90°)				
Rotate Left (90°)				
Back				

Fig 1. Symbols used in pre-literate coding environments.

1.1. *Computational thinking in early childhood*

The ideas behind computational thinking (CT) in childhood date back to Papert (1972, 1980), who wrote about the "computer-controlled cybernetic" Turtle within the LOGO environment as a model for "objects-to-think-with." It was a model for the design of "objects in which there is an intersection of cultural presence, embedded knowledge, and the possibility for personal identification" (Papert, 1980, p.11). Introducing these ideas through LOGO involved children engaging with elements of CT including sequences, debugging, procedures, and decomposition. However, "objects to think with" were meant to teach children mathematical thinking—not programming (Papert, 1972). Papert's work around children's ideas and mathematical thinking lay the foundation for the research currently being done in early childhood around computational thinking. One of the most developed frameworks for early childhood CT comes from Bers (2018). Bers talks about CT in terms of *powerful ideas*, which according to Papert, "afford new ways of thinking, new ways of putting knowledge to use, and new ways of making personal and epistemological connections with other domains of knowledge" (Papert, 2000 cited in Bers, 2018, p. 70). This view of CT has roots in Papert's foundational work with LOGO on solving problems algorithmically (Bers, 2018). Bers proposed seven developmentally appropriate powerful ideas for early childhood computer science education: algorithms, modularity, control structures, representation, hardware/software, design process, and debugging (Bers, 2018, p.71).

Coding, even for young children, is thought to be an important *context* for developing CT (e.g., Bers, 2018; Relkin et al., 2020; Wang, Choi, Benson, Eggleston & Weber, 2021). Sharing those views, the operationalization of CT we present here focuses on the learning we *observed* kindergarten children demonstrate (e.g., gestures, movements, actions, words) as they used screen-free robot coding toys that we refer to broadly as following a "grid-agent" model. These toys are developmentally appropriate for children who are emerging readers because they rely on a symbol system for codes (see Fig. 1 for an example of some of the symbols used in coding environments designed for pre-literate children).

Starting from Ber's powerful ideas, we focused on: *algorithmic thinking, decomposition (modularity), debugging, abstraction,* and *spatial thinking* (Clarke-Midura, Silvis, Shumway, Lee, & Kozlowski, 2021). We define *algorithmic thinking* (AT) as developing and using ordered sequences of instructions. Important subcomponents of AT include sequencing by ordering and arranging codes based on knowledge of syntax and semantics; planning programs by ordering and arranging codes and reading and enacting programs. We define *decomposition* as the ability to recognize parts in part-whole relationships (coordinating units or segments of code with one another as well as with the whole program), building a whole from parts (combining chunks or sequencing codes one-by-one), and breaking a whole into parts (breaking program into units or segments of code to simplify the task). We define *debugging* as the ability to recognize that bugs/errors exist, locate the specific error/bug, propose a solution for how to fix the error/bug, and fixing the bug. We define *abstraction* as a process of ignoring irrelevant details for the purpose of focusing on the essential aspects of a problem or situation (Rich & Yadav, 2020). In addition, we added an additional skill, *spatial thinking,* as important to CT in early childhood (Clarke-Midura, Silvis, Shumway, Lee, & Kozlowski, 2021; Shumway, Welch, Kozlowski, Clarke-Midura, & Lee, 2021). The National Research Council (NRC) defined spatial thinking as comprised of three elements: concepts of space, tools of representation, and processes of reasoning (NRC, 2006, p.9). Spatial thinking is multifaceted, it involves reasoning with representations that provide access to spatial information such as spatial locations and transformations (Whiteley, Sinclair & Davis, 2015). It is increasingly being recognized elsewhere as connected to CT for elementary school students (Tsarava et al., 2022). In the context of our work, *spatial thinking* involves reasoning with an agent's orientations, locations, and navigation in space and operations on those spatial relationships (NRC, 2006; Sarama & Clements, 2009; Whiteley et al., 2015).

1.2. *Assessing CT in early childhood*

Interest in CT assessment has grown in recent years, as evidenced by two independently published systematic literature reviews (Cutumisu, Adams & Lu, 2019; Tang, Yin, Lin, Hadad & Zhai, 2020) and special issues dedicated to the topic (e.g., Weintrop, Wise Rutstein, Bienkowski & McGee, 2021). A common observation from these reviews is that there are few valid and reliable assessments of early childhood CT.

Most existing assessments of early childhood CT that provide evidence of validity or reliability were designed to be used either after instruction has taken place or in the context of a pre-post sequence. Such assessments are *summative*; they characterize a child's proficiency and support inferences about what a child knows, understands, or can do before or after a unit of instruction. For example, TechCheck is a multiple-choice assessment with 15 items (Relkin et al., 2020), designed to assess Bers' powerful ideas of CT in early childhood including algorithms, modularity, control structures, representation, debugging, and hardware/software (Bers, 2018). It does not require programming knowledge and is not bound to a particular programming context. The instrument was validated for 5- to 9-year-old children and provides a total score of CT ability (Relkin et al., 2020). TechCheck-K is a modified version of TechCheck for kindergarten children, in which the 15 multiple-choice items have three answer responses, as opposed to four. It also provides a total score of CT ability (Relkin & Bers, 2021). TACTIC-KIBO is an assessment for children aged 4–7 that is specific to the KIBO robot (Relkin & Bers, 2019). It is interview-based and scored to classify children using four levels of proficiency: proto-programmer, early programmer, programmer, and fluent programmer. The Coding Stages Assessment is an interview-based assessment for children aged 5–7 that measures mastery of the ScratchJr syntax and grammar and children's ability to use it purposefully (de Ruiter & Bers, 2021). It is based on Bers's Coding Stages Framework (Bers, 2019), which describes a learning path in coding from simple skills to more complex ones they define as: emergent, coding and decoding, fluency, new knowledge, and purposefulness. It is designed to be interactive; children respond to questions or complete tasks in ScratchJr. Children are assigned to one of Ber's five coding stages and are provided a numeric total score. There is a version of this assessment called "Solve-its" that can be used periodically throughout a curriculum (de Ruiter & Bers, 2021). KIBO Project Rubric is a rubric-based assessment to measure KIBO projects in terms of programming concepts and project design elements. A score is awarded that provides an estimated level of mastery as exhibited in the KIBO project (Govind & Bers, 2021).

1.3. *The present study*

Currently, there is a dearth of research on formative assessments of CT in early childhood, in particular, how to design formative assessment tasks to aid in the immediate modification of curriculum and instructional activities. We define formative assessments as processes and procedures for making inferences about children's learning, linked to immediate classroom instruction and pedagogy. The formative assessments we describe in this paper take place in classrooms when teachers and small groups of children engage in hands-on coding tasks with grid-agent coding toys on the floor during instruction.

1.4. *Evidence-Centered design of formative assessments*

We use the Evidence Centered Design framework (ECD; Mislevy & Haertel, 2006) in our assessment design process. ECD views assessment as an evidentiary argument from what we observe children say, do, or make in a few particular circumstances, to formulate inferences about what they know, can do, or have accomplished more generally (Mislevy & Haertel, 2006 p.7). As such, ECD provides a unified framework that considers key questions about whom we assess, for what purposes, and under which constraints (Oliveri et al., 2019). In doing so, such assessment designers explicitly link features of assessment tasks, the evidence of child performances on the tasks, and the knowledge or understanding exhibited by the evidence. This view of assessment as argument is central to discussions around validity (e.g., Kane, 2006) while offering what Mislevy (2007) calls "validity by design" where, as designers, we structure our approach in such a way that validity evidence emerges (Mislevy, 2007, p. 467).

ECD consists of 5 layers: (1) domain analysis, (2) domain modeling, (3), the conceptual assessment framework, (4) assessment implementation, and (5) assessment delivery. In the first two layers, the focus is on the purposes of the assessment, the nature of knowing, and structures for observing and organizing knowledge. This information is put into "design patterns" that articulate the kinds of features that assessment tasks will need and the kinds of performances those features will elicit. In the third layer, the conceptual assessment framework (CAF), the focus is on the student model (what skills are being assessed), the evidence model (how do we measure it), and the task model (situations that elicit the behaviors/evidence). These three models are developed with the information from the first two layers in ECD – the design pattern in particular – to provide technical details of the tasks (such as potential student performances/products during assessment implementation and delivery) and a specification of the kinds of features of the tasks that will provide evidence about the student model.

1.5. *KRO indicators*

Our formative assessment tasks are designed around the errors that children make when solving CT tasks. Historically, work that has focused on children's errors in reasoning and problem-solving tasks has asserted the cause of those errors are underlying misconceptions. The typical recommended instructional approach for responding to those misconceptions involves challenging, countering, or refuting the inferred misconceptions and then immediately doing instructional work to replace those misconceptions with correct or desired conceptions (Posner, Strike, Hewson & Gertzog, 1982; Tippett, 2010). However, an alternative approach is to take a less adversarial view of children's 'incorrect' ideas and instead see children's erroneous reasoning as the result of an active knowledge system that is in need of refinement. This 'knowledge refinement' approach (Smith et al., 1994) asserts that children's thinking should not be seen as deficient, but rather as containing many elements, some of which may be resources for constructing normative understanding (Hammer, 2000).

The hypothesis underlying our formative assessment development work is that for early childhood CT teaching and learning – in this case, coding environments developed for emerging readers that rely on a symbol system for codes – the mistakes that children make exhibit some patterns and consistencies. These patterns in how they solve CT tasks may vary child by child and task by task. However, we predict that when we aggregate these patterns, they will reveal a general trend that is consistent across different groups of children. These regularly occurring mistakes can thus serve as indicators of knowledge refinement opportunities and are where we can expect to see when CT knowledge is still developing.

Our approach has some similarities to research on learning trajectories in early childhood mathematics, which postulates most content knowledge is learned along developmental progressions or levels of thinking that are consistent with children's informal knowledge and patterns of thinking and learning (Sarama & Clements, 2009). While our work is not as formalized as the learning trajectories research, our identification of *indicators of knowledge refinement opportunities* provide insight into very early understandings of what potential patterns and levels of thinking may look like in early childhood CT.

With respect to defining KRO indicators as occasions of children's semi-regular errors when engaged in computational reasoning tasks, it is common to see those as opportunities for children to develop their CT debugging competencies. When a child produces errors with programming codes, one of the long-term educational goals of CT development is to help children become proficient in 'debugging' those. However, for our purposes, indicators should not be confused as instantiations of debugging. Debugging is a practice linked to solving a problem or fixing a program and has both a technical dimension (e.g., identifying bug, fixing bug) and an affective dimension (e.g., persistence) (DeLiema et al., 2020). KRO indicators, on the other hand, are linked to intuitive knowledge and developmental understandings that lead to the need to debug or that occur, albeit fragilely, during debugging attempts and processes.

1.6. *Integrating KRO indicators with ECD*

The aim of this paper is to put forth a new framework for conceptualizing the design of formative assessments that builds on ECD but centers identification and analysis of indicators of knowledge refinement opportunities. Two research questions guided this analysis: 1. *What indicators of knowledge refinement opportunities emerge as kindergarten students participate in a coding curriculum?* 2. *How do we design formative assessment tasks that activate these indicators of knowledge refinement opportunities?* This means that rather than start only with targeted CT knowledge, skills, and abilities in the design of assessment tasks, we propose a concurrent look at indicators coupled with analyses of how those indicators connect to targeted CT knowledge, skills, and abilities. This then yields design patterns that can then support specification of formative assessment task models.

The remainder of the paper will show what this process looks like and what it yields with respect to design patterns and task models. The process involves three steps: identify the KRO indicators as they arise, create a general task model based on the indicator, use the task model to design new formative assessment opportunities. To help orient the reader, consider a simple hypothetical example. Imagine that when using a grid-agent coding toy and given the task of programming it to move forward three spaces, many young children are observed to only provide a single 'forward' (F) code. When asked to predict where the robot will move given that single code, the children point to the third space in front of the robot. In terms of knowledge refinement, the children are correctly understanding that the forward code is used for forward movement, although they are not yet recognizing that one code produces a single discrete movement. This tendency to use a single forward code for a multi-unit forward movement is an indicator of a knowledge refinement opportunity. From this, we can begin to articulate some task models that could be used during a grid-agent coding activity that can help a teacher test for and elicit this indicator. This process formalizes what many teachers may already choose to do in such situations. However, articulating these indicators and identifying what types of formative assessment tasks elicit them can prime teachers for what to look for as children solve tasks and how to recognize children's level of CT knowledge (through the indicators).

2. Methods and materials

The present study is part of a larger design-based research project (NSF # 1842116) that is operationalizing CT in early childhood and developing corresponding curricular materials and resources, including assessments. The present study focuses on a small sub-group of children who participated in coding lessons and then completed an early version of a summative CT assessment. In particular, the present study grew out of seeing the errors children made while participating in the coding instruction as well as when they completed the tasks in the summative assessment.

2.1. Setting, participants, ethics statement, and data collection

Participants for this study came from a rural kindergarten classroom in the western United States. Participation was voluntary. We followed our approved Institutional Review Board (IRB) protocols for conducting research with minors. The school was classified as Title 1 and received funding to offer full-day kindergarten to a subset of students. Seventeen children (female = 5), ages 5 and 6, from the full-day kindergarten participated in a total of six 30-minute coding lessons in small groups. The lessons introduced coding using Botley (Learning Resources, 2023) for three lessons and Cubetto (Primo, 2023) for three lessons. Two to three days after the participants completed all the coding lessons – which took about 2 weeks – they took the standardized interview-format CT summative assessment (~15 min) we developed for the larger project (see Clarke-Midura, Silvis, Shumway, Lee, & Kozlowski, 2021; Na & Clarke-Midura, 2023). The assessment was interview-format to account for kindergarteners' emerging literacy skills. All curriculum lessons and assessments were video recorded (16 h total), and the videos were the basis for the current analysis. Four members of the research team who had formal classroom teaching experience, advanced degrees in education, and formal training on assessment administration taught the lessons and administered the assessments. Each researcher taught one group for the six lessons and also administered the summative assessment.

2.2. Coding toy lessons

The Botley and Cubetto lessons were designed through an iterative design-based-research process. We developed a menu of tasks for small groups of children for each toy starting with an introductory lesson focused on the toy's syntax and designed to help children understand the relationship between each code and a specific movement. The other tasks focused on sequencing codes, decomposing paths, and debugging programs. Some of the tasks required children to program the robot to get from one location to another. Other tasks required them to "crack the code" and figure out the program after watching the robot move. There were also open-ended tasks where children were able to build their own programs.

2.3. CT summative assessment

We developed a summative, task-based assessment to be administered after the coding lessons. While the coding lessons were taught using Cubetto and Botley, the assessment tasks were not bound to a specific coding toy; they were unplugged and "toy-free." The assessment tasks involved a paper grid and agents (i.e., small tokens that were made to look like an insect). There were 27 items total in the assessment. The tasks were performance based in that children were asked to sequence or debug programs that involved moving an agent around a 10″x10″ grid (Fig. 2). The assessment codes consisted of four directional arrows (Forward, Backward, Rotate Left, Rotate Right). The tasks involved storylines around moving an agent from one location to another (e.g., moving the beetle to the grass patch on the grid). Some tasks had only one possible correct answer whereas other tasks had multiple correct answers. The administration of the assessment was standardized. For more information on the assessment please see ; (Clarke-Midura, Silvis, Shumway, Lee, & Kozlowski, 2021; Na & Clarke-Midura, 2023).

2.4. Data analysis

Analysis took place in multiple phases. In the first phase, we engaged in open and axial coding of the assessment video data (Patton, 2014; Saldaña, 2021; Strauss & Corbin, 1998). This involved reviewing the assessment videos for all 17 students, identifying children's coding errors (i.e. indicators) and then grouping them thematically. We created definitions of the indicators with examples and then recoded the assessment video to conform to this resulting coding scheme (assessment indicators) (Fig. 3).

Next, we conducted a priori coding of the six coding lessons for the same 17 children. This involved review of video for the identified indicators and then documenting the characteristic and variable features of the task and situation where they occurred (curriculum indicators). We created summaries of the indicators by task within each lesson and by child. We then compared the indicators in the assessment to the curriculum by child, ensuring the assessment reflected the learning that occurred during the curriculum.

We then brought in an additional coder in order to establish inter-rater agreement and determine if the coding scheme was reliable (Vacha-Haase, 1998). Two researchers independently coded 10% of the assessment and curriculum data (42 excerpts from the curriculum and 16 excerpts from the assessment). Overall, the two researchers had strong agreement when double coding 10% of all the assessment indicators and curriculum indicators data combined (κ =0.887). Specifically, the two researchers had strong agreement on the assessment indicators (κ =0.828) as well as strong agreement on the curriculum indicators (κ =0.905).

After establishing strong inter-rater agreement, an individual researcher went back and re-coded all the curriculum and assessment data using the final codebook to produce a final count and classification of indicators.

We then specified the design patterns that document the characteristic features and variable features for potential assessment items. We coded the indicator data from the classroom lessons indicating the child performance context (i.e., writing program, debugging program, enacting program, identifying a code), and the variable features (i.e., length of code, number of turns, position of turns, and the orientation of the robot from the student's perspective). These allowed us to create design patterns and the task models for the formative assessment tasks.

3. Results

Recall that our research questions asked, What indicators of knowledge refinement opportunities emerge as kindergarten students participate in a coding curriculum? And How do we design formative assessment tasks that activate these indicators of knowledge refinement opportunities? In the following sections, we first present the indicators of knowledge refinement opportunities that emerged, specifically exemplifying the three most common indicators. We then describe how we designed formative assessment tasks that target the indicators.

3.1. Identifying KRO indicators

We identified 12 indicators of knowledge refinement opportunities that occurred with different levels of frequency in the data (Table 1; see Appendix A for definitions of each). The three most frequent indicators

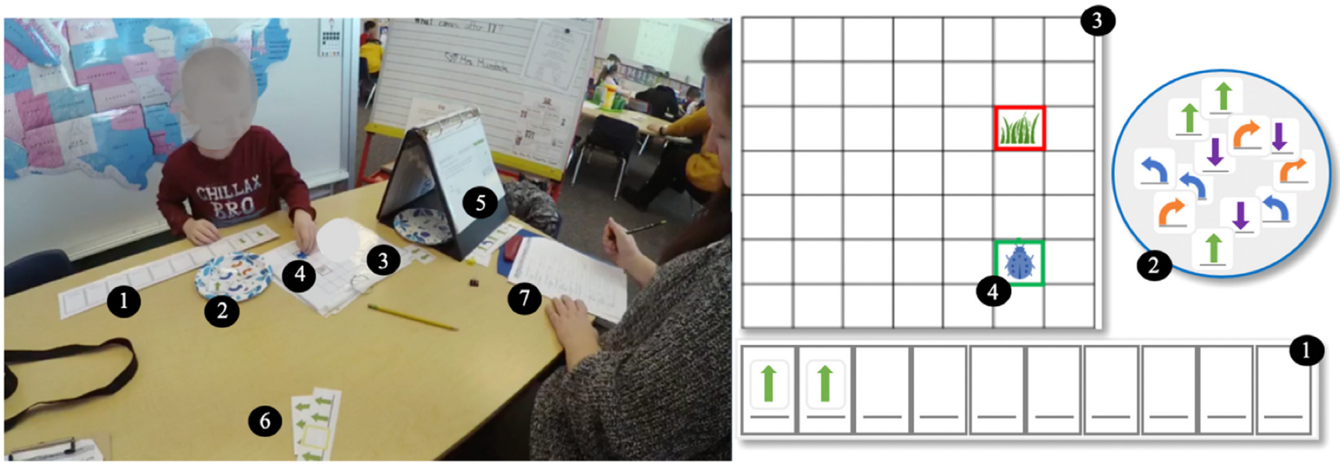


Fig 2. Unplugged CT Assessment. Left shows materials used in assessment tasks: (1) program organizer (2) arrow codes (3) grid pages, administration flip book (4) moveable agent (5) administration pages, with script (6) preset code strips (7) scoring sheets. Right shows child's-eye view of assessment materials.

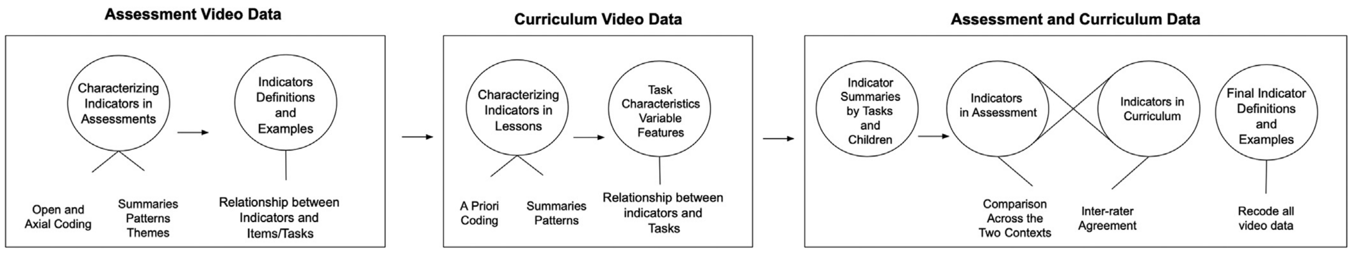

Assessment Video Data
Characterizing Indicators in Assessments
Indicators Definitions and Examples
Open and Axial Coding
Summaries Patterns Themes
Relationship between Indicators and Items/Tasks
Curriculum Video Data
Characterizing Indicators in Lessons
Task Characteristics Variable Features
A Priori Coding
Summaries Patterns
Relationship between indicators and Tasks
Assessment and Curriculum Data
Indicator Summaries by Tasks and Children
Indicators in Assessment
Indicators in Curriculum
Final Indicator Definitions and Examples
Comparison Across the Two Contexts
Inter-rater Agreement
Recode all video data

Fig. 3. Representation of the Analytic Process.

Table 1
Frequency Count of Indicators of Knowledge Refinement Opportunities.

Knowledge Refinement Indicators	Curriculum Frequencies	Assessment Frequencies
Unit of Turn	101	71
Wrong Rotation	61	12
Includes Starting Space	78	7
Distance of Arrow	34	19
Spaces Between	12	14
No Turn Arrows	14	6
Corresponds to Direction not Movement	9	9
Use of Backward	15	3
Incorrect Sequencing	12	1
Orientation of Arrows = Movement	3	4
Diagonal Line	3	3
Disregards Starting Orientation	1	2

were *Unit of a Turn, Wrong Rotation*, and *Includes Starting Space*. These indicators involved children's spatial knowledge, counting knowledge, and unit knowledge. We then reviewed the variable features of the tasks where the KRO indicators occurred. Illustrations of the top three indicators and some of the contextual features are provided below.

3.2. Indicator: unit of the turn

The most frequent KRO indicator is what we called *Unit of the Turn*. In coding successfully with these robot toys, children needed to develop an understanding of a precise rotation in space. The coding toys we used treat a rotation code as a 90-degree rotation. The symbol is usually in the form of an arrow with some curved and some straight. To code with these robot toys, children need to learn both how the robot rotates in space and how this movement is symbolized in the specific codes for the robot's system. When coding, children in our study often indicated their knowledge that the robot needs a rotation movement. However, rather than interpret the rotation code as a 90-degree rotation, children often interpreted it as a rotation 90-degrees *and* one forward movement (Rotate + Move). We call this indicator *Unit of the Turn*.

3.2.1. Unit of the turn examples

In the following examples, children used the rotation arrow to do things other than the rotation arrow's true meaning and thereby indicated an opportunity for knowledge refinement. In the first example (A), children were trying to move Cubetto to a specific square on the grid (see Fig. 4). This task was designed to be a FFFRF program, which we considered moderately challenging due to variable features of more than three codes and one rotation. The complexity was heightened by Cubetto's starting orientation in relation to the child in that Cubetto and the child did not share the same perspective (they were oriented facing different directions).

The children built the program FFFR. The teacher asked the child operating Cubetto's programming board what the program FFFR will instruct Cubetto to do, and they moved their finger along the grid to show it moving forward three, rotating, and then moving forward one more space, thereby interpreting the red code (Rotate Right) to mean Rotate + Forward.

In this example, *Unit of the Turn* is an indicator that the child does not fully understand 1-to-1 *code-to-movement correspondence* and possibly the *spatial code meanings*, the semantic knowledge of what the codes mean (that rotate right is only rotate right, not rotate and move). While they were able to gesture with their finger the path Cubetto would take, they were not able to coordinate that with the program. Children's CT knowledge is fluid and constantly in refinement; while in this example the child may have interpreted the right turn code as rotate and move, in a later task with different variable features, they may treat it only as a rotation.

In the second example, Fig. 4(B), the children need to get Botley from its starting location on the grid space to the grid space with the bug

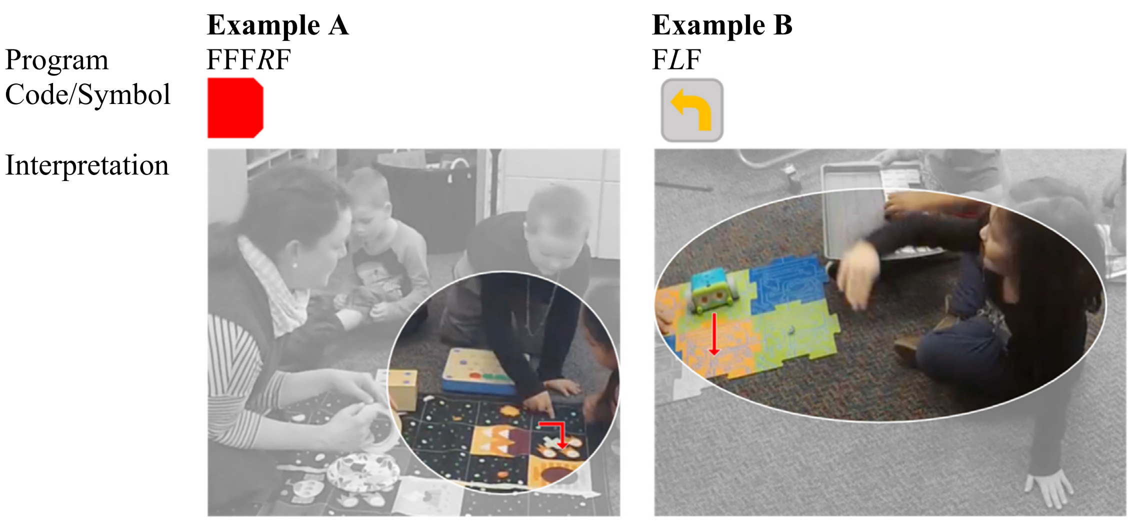


Fig. 4. Unit of the Turn KRO Indicator. The child's finger movement from one square to the next indicates he is treating the unit of the turn as Rotation + Forward (Example A). The child's arm movement sweeping to the left indicates she is treating the unit of the turn as moving to the left and possibly rotating (Example B).

(FLF). Regarding variable features, the program has three codes and one rotation. What makes this task challenging is the starting orientation of the robot in relation to the children (not sharing a perspective).

The teacher asked the child in the picture how to program Botley to reach the bug. The student said, "I know" and moved their arm in a sweeping gesture to the left (Fig. 4). They then put a left turn arrow on the program organizer (L) to represent this movement. Botley's orientation in relation to the child's orientation was not yet considered as she was operating from her own perspective. From her perspective, she was using the rotate left arrow for Botley to move forward (and potentially rotate) in the cardinal direction (see the left turn arrow superimposed over on the grid in Fig. 4). Again, we see that *Unit of the Turn* is an indicator that the child understands the direction of the movement needed but is still developing the 1-to-1 *code-to-movement correspondence* and possibly the semantic knowledge of what the codes mean (*spatial code meanings)*. As we see in both of these examples of the *Unit of the Turn* indicator, the children are operating under partial knowledge of what the rotation arrows do, but not complete knowledge of the entire unit of the turn.

3.3. Indicator: wrong rotation

The second most frequent indicator was *Wrong Rotation* and occurred when children chose the wrong rotation arrow for the program or when they rotated the robot the wrong direction compared to the rotation arrow selected. In the latter, they may rotate the robot to the left but place a rotate right code in their program. Developmentally, kindergarten children are expected to understand relational positions between objects, such as next to, behind, and below (CCSSI, 2010). They are able to identify the differences among various kinds of movements and directions but are still learning precision and language that represents those spatial relationships (Clements & Sarama, 2009). For example, they are still learning to remember which direction is labeled "left" versus "right." When using the robots, some robots' arrow systems are color coded allowing the children to associate the rotation to a color rather than using language for left and right. Sometimes, indicators illustrated students' need to coordinate these symbols with precise actions.

3.3.1. Examples of wrong rotation

Three common examples of *Wrong Rotation* are provided in Fig. 5 (A, B, and C). In example A, the teacher rotated Botley to the left and asked the child in the picture what code would make Botley do that. She spun her hands in the air and said "turn around" and then put a Rotate Right code on the board. During that lesson, the child was still refining her spatial reasoning capabilities. She understands a rotation is necessary but could benefit with more practice using rotations to understand how the robot moves in 3 dimensions, as well as more practice thinking about the robot in different positions and orientations (Shumway, Welch, Kozlowski, Clarke-Midura, & Lee, 2021). Including formative tasks that vary the rotations and starting orientation of the agent will help children develop spatial thinking and reasoning.

Example B illustrates two children enacting the robot's rotation correctly but putting the wrong rotation code in their program. The children were sharing Cubetto's orientation and perspective as they worked on the task. In terms of variable features that increase task complexity, this program has more than 3 codes, two rotations, and starts with a rotation.

The child on the right first traced the path with his finger and put a red tile in the board, which represents Rotate Right. He picked up Cubetto and enacted the movement and rotated Cubetto to the left, despite having placed the right rotation tile in his program. He then enacted Cubetto's movement again, Rotate Left and Forward one square, putting a forward tile in the program board to correspond to the movement. The child on the left took over and rotated Cubetto to the right but handed his partner a yellow tile, which represents Rotate Left.

In this example, the children knew which way Cubetto was supposed to rotate but they selected the wrong rotation tile. The rotation directions on the Cubetto tiles are very subtle and the children using the *Wrong Rotation* is an indicator that they need to coordinate symbol-action correspondence, which is semantic knowledge of what each code means. We do see them demonstrate knowledge of 1-to-1 *code-to-movement correspondence* because they place the rotation code with the correct association that the rotation is going to merely rotate the robot without any forward or backwards movement. However, their understanding of the technical direction of the rotation in still in refinement.

In example C, both children are facing the robot, meaning that they are not in the perspective of the robot. This starting orientation requires the spatial knowledge that the orientation of the robot is different from their own perspectives. The children are working to get Botley to the square behind it. The teacher's intended solution was the use of one backward arrow. The children instead were creating the program LFLFLF. The variable features of the program they were building consisted of the following characteristics, making it a complicated program: (> 3 codes), 3 rotations, starting with a rotation, and the starting orientation of the robot is at 180° from their perspective.

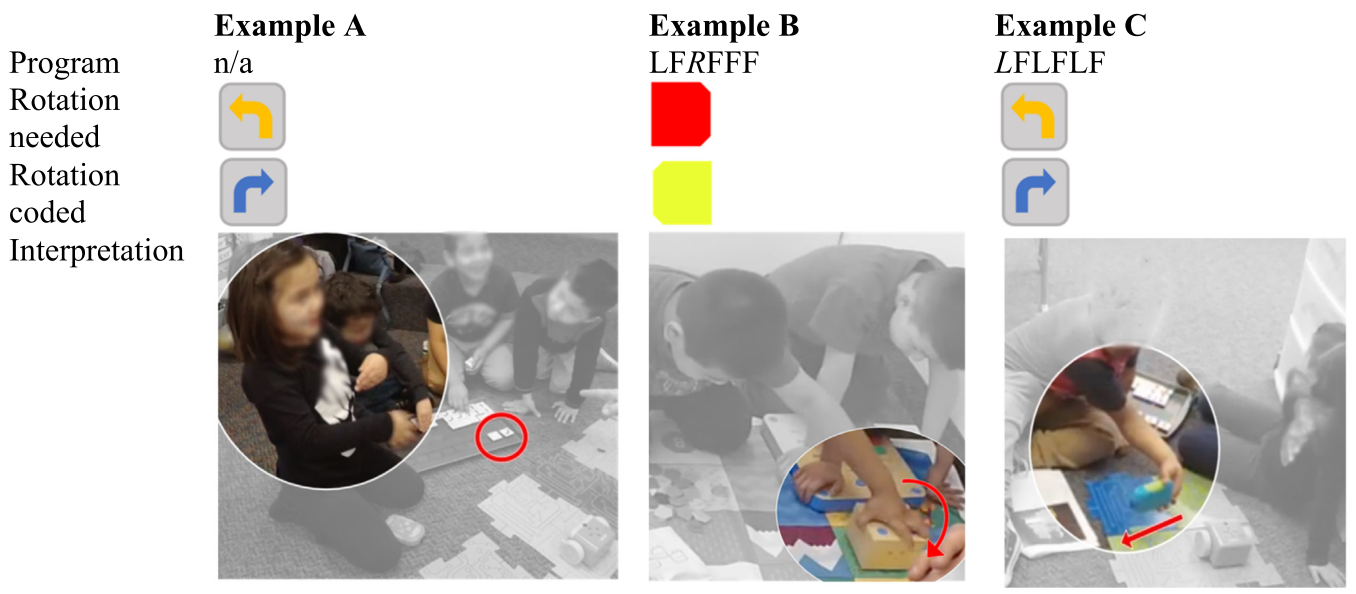


Fig. 5. Wrong Rotation KRO Indicator. The child spins her hands in the air to demonstrate left rotation and then codes a *right* rotation (Example A). The child correctly rotates the robot to the right but then codes a *left* rotation (Example B). The child motions with the remote to his own right when the robot needs to rotate *left* (Example C).

In the figure, we see the child on the left is holding onto the remote and using it to gesture to his right, to the square next to Botley's starting position. From this child's own perspective, the direction he wants Botley to rotate is to the right of the square Botley is on. However, from Botley's perspective, a left rotation is required. He pointed to the rotate right arrow and his partner put the rotate right arrow into their program.

Example C demonstrates that *Wrong Rotation* is an indicator of learning to coordinate his own spatial orientation with the *spatial orientation of the robot*. This involves understanding that the codes always produce the same movement, but the orientation of the robot determines which code to choose. When children select the wrong rotation, this error is sometimes explained by not yet understanding how to write the code from the robot's perspective regardless of their own orientation and positioning. Thus, formative tasks that involve children writing programs that not only have rotations, but where the robot is at a different orientation from their own can provide information about children's understanding of spatial orientation related to CT.

3.3.2. Indicator: includes starting space

The third most common indicator in the curriculum was *Includes Starting Space*, in which children counted the robot's starting grid square when determining how many forward codes they needed for the program. Children making this error counted the grid pieces or squares rather than the robot's movements. Counting the robot's movement entails *counting on* from the starting square (i.e., the robot's starting point), rather than including the square the robot is on in their forward count.

3.3.3. Examples of includes starting space

In this example, the teacher asked the child how many forward arrows they needed to get Botley to the square that is two squares forward. The child counted the squares, starting with the one Botley was on, and indicated they needed three forwards, rather than two. *Includes Starting Space* is an indicator that children may be refining their understanding of counting the movements of the robot rather than the grid squares. Based on the variable features of this task (i.e. 3 or fewer codes, no rotations, robot's orientation) we see this as a useful introductory formative task for assessing children's understanding of mathematical and spatial competencies underlying CT. Designing tasks for children to engage in more instances of counting movements and make predictions could be one way to help them focus on the movements and not the grid (Fig. 6).

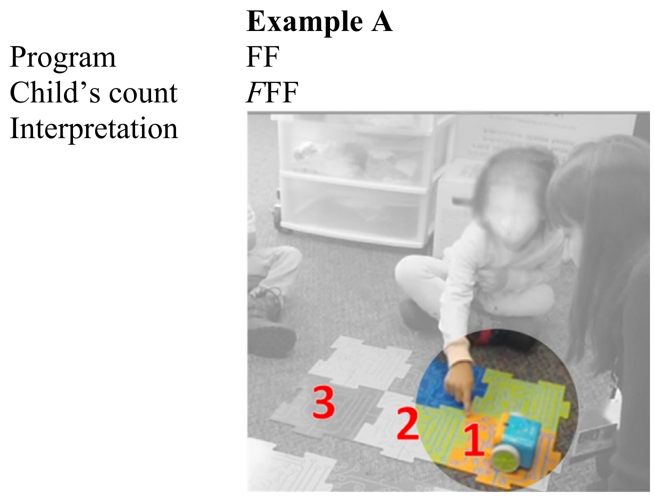
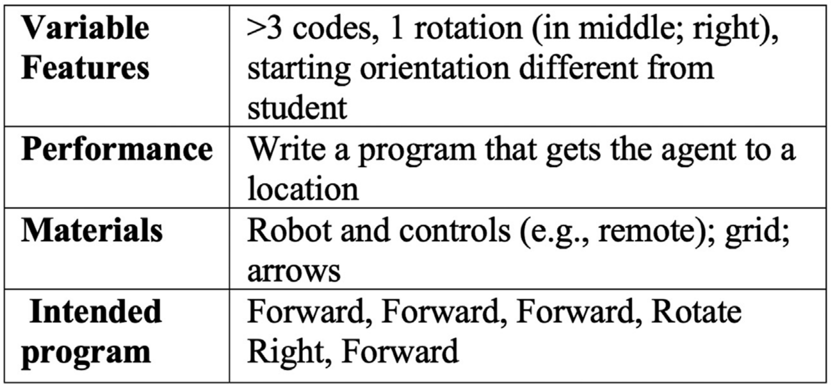


Fig. 6. Includes Starting Space Indicator. Children include the robot's starting space in their counts, resulting in an extra forward code.

3.4. Designing formative assessment tasks that respond to indicators

In the previous section, we shared examples of the indicators of knowledge refinement opportunities that emerged in the data. In this section, we now share examples of how ECD can be integrated with indicators to design formative assessments that provide evidentiary arguments about the state of children's CT proficiencies. We focus on how to explicitly link features of the assessment tasks, the evidence of student performances on the tasks, and the knowledge exhibited by the evidence.

Identification of the indicators informed the specification of design patterns to help us define and model the indicators (the student model) across a variety of screen-free coding toys. After specifying the design patterns, we identified potential task performances (task model), charac-



Variable Features	>3 codes, 1 rotation (in middle; right), starting orientation different from student
Performance	Write a program that gets the agent to a location
Materials	Robot and controls (e.g., remote); grid; arrows
Intended program	Forward, Forward, Forward, Rotate Right, Forward

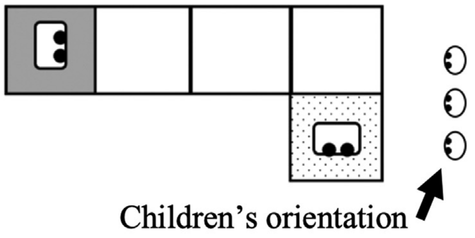


Fig. 7. Task Model for Unit of the Turn. The task model is on the left. On the right is the program goal as well as where to position children during the task.

Table 2
Design Template for Unit of the Turn Formative Assessments Assessment argument components 1=student model, 2=evidence model, 3=task model.

Focal Indicator[1]	Unit of the Turn
Additional Knowledge and Skills needed[1]	1-to-1 code-to-movement correspondence; semantic knowledge of what the codes mean; orientation of the robot determines which code to choose
Potential observations[2]	Rotations enacted/coded as rotate + move; forward enacted/coded to include rotation
Potential task performances[3]	Write a sequence of codes; identify code(s) based on movement; enact code or a sequence of codes; gesture path on grid
Task Characteristic features[3]	Robot coding toy and its materials (e.g., remote, board), grid, codes, rotations
Task Variable features[3]	Program: with more than 3 codes; more than 1 rotation; starts with rotation; starting orientation of agent different from children

teristic and variable features (what the task should contain and ways to make new related tasks by varying features) as well as potential observations (evidence model). Given that CT is a multi-dimensional construct, there are overlaps in the design patterns across the indicators such that many of the additional knowledge and skills required, potential task performances, and characteristic and variable features are similar. Where they differ is in the observations and what those observations tell us about children's understanding.

In Table 2, we present a design pattern that provides argument components for formative tasks around *Unit of the Turn* indicators. These design patterns can be used to create task models, which are essentially templates for the general specifications of tasks. The task models can be used to generate multiple tasks that target the same concepts. Figs. 7 and 8 present two different task models for *Unit of the Turn* that can be used across screen-free coding toys and integrated in lessons.

The design of our existing curriculum focused on teaching *algorithmic thinking, decomposition (modularity), debugging,* and *spatial thinking.* However, focusing on indicators helped us identify other knowledge that is necessary for CT in the context of grid-agent coding toys. For example, in order to engage in algorithmic thinking and build a sequence of codes, children must refine their knowledge of 1-to-1 *code-to-movement correspondence* and *spatial code meaning,* semantic knowledge of what individual codes mean. Our design patterns and task models allow us to integrate these formative assessment tasks into our existing curriculum alongside possible sequences of tasks that start easy and become more challenging based on the variable features.

Figs. 7 and 8 show that in order to get feedback on children's understanding of *Unit of the Turn* (rotations), teachers need to implement tasks that involve rotations. There are a few potential performances the tasks can target, such as write a sequence of code (program), enact a sequence, or identify a code based on the robot's movement. These task models can easily be integrated into any story line around movement on a grid. If a child writes a program where a rotation code functions as a 'rotate + move' then the teacher can observe that they are still needing to refine how they understand 1-to-1 *code-to-movement correspondence* and possibly the *spatial coding meaning,* that rotate right is only rotate right, not rotate and move, and that rotations occur on a fixed point and do not involve forward movement. It is also possible that the child is in the midst of refining their ability to coordinate the path on the grid with the program (*space-symbol coordination*). Viewed as formative information, teachers can use this evidence to alter instruction, asking the child to explain what each code does and having them enact the movements with the robot before running the program. They can also ask the child to program iteratively where they write a piece of code, test it, and keep adding one code at a time to see how each code represents one movement. Alternatively, teachers could design multiple mini-tasks that isolate indicators to determine systematically whether a child's error is related to their incomplete understanding of spatial code meaning, code-to-movement correspondence, orientation of the robot, counting on, etc.

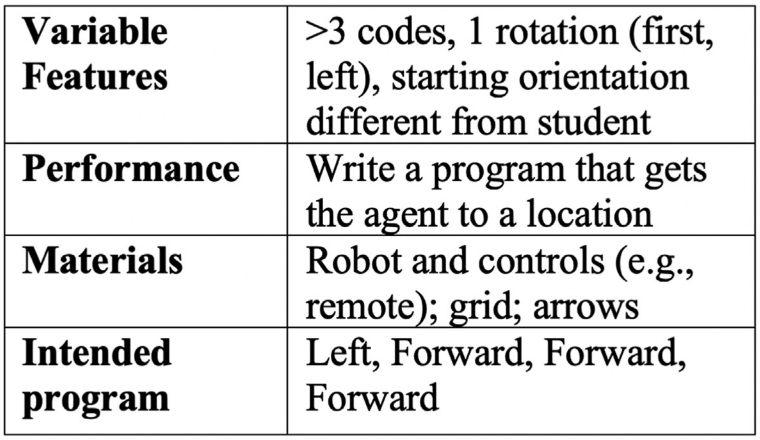
The templates for developing formative assessment tasks can be easily integrated into our existing curriculum.

4. Discussion

In this paper, we proposed a principled approach to formative assessment design that starts by analyzing patterns in errors that children make when completing CT tasks, which we call *indicators of knowledge refinement opportunities*. This view of children's errors as containing productive elements of the targeted understanding has similarities to the knowledge refinement approach introduced by Smith et al. (1994). Rather than refer to these errors as misconceptions and try to "correct" them, we see children's errors as productive indicators of their understanding. Our approach of identifying the indicators and then illustrating how to use the indicators to develop formative assessment tasks, integrates research on knowledge refinement (Smith et al., 1994), Evidence Centered Design (e.g., Mislevy & Haertel, 2006) and classroom-based formative assessments (e.g., Black & Wiliam, 2018). This integrated approach allows for a better understanding of the nature and growth of CT in early childhood, which we discuss below.

4.1. Implications for a model of CT that takes into account indicators of knowledge in refinement

The *indicators of knowledge in refinement opportunities* provide insight about children's understanding of CT and how it develops over time.



Variable Features	>3 codes, 1 rotation (first, left), starting orientation different from student
Performance	Write a program that gets the agent to a location
Materials	Robot and controls (e.g., remote); grid; arrows
Intended program	Left, Forward, Forward, Forward

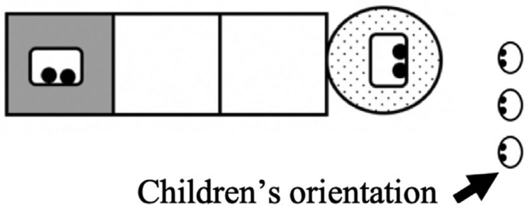


Fig. 8. Task Model for Unit of the Turn. The task model is on the left. On the right is the program goal as well as where to position children during the task.

As we saw in the *Unit of the Turn* and *Wrong Rotation* examples, some of the indicators suggest the children do not yet understand the *spatial code meanings,* or what the code instructs the agent to do. Some children may interpret a rotation code as a rotation in one task but then use it as a forward code in another task. Some indicators suggest the children do not understand that one code or symbol produces a single discrete movement. Overall, some of it may be an indicator of their developing precision in spatial knowledge. As we saw in the *Unit of the Turn* example B, the child interpreted the Right rotation code as a Rotate Right + Move Forward. Some of the indicators highlight that children have an emerging understanding of orientation. Recall that in the *Wrong Rotation* example C above, the children built the program from their perspective and not the robot's orientation. The codes always produce the same movements but depend on the agent's orientation-this concept may be difficult to grasp at first for young children because they often have difficulty determining spatial relations from perspectives that differ from their own (Roberts, Jr. & Aman, 1993). Our findings regarding the indicators are not isolated to the present study or research on screen-free coding toys. Previous studies of pre-literate children engaging in CT tasks (both plugged and unplugged) identified similar "difficulties" to what we refer to as indicators or errors in our study. For example, in a study with kindergarten children (ages 5–6), using a LOGO-based environment, (Fessakis, Gouli and Mavroudi 2013) found that the children had difficulties with the *spatial code meanings* such as decoding the various symbols for commands, they had trouble with angle and turn symbols, and the orientation direction of the agent on the screen (a lady bug). Children also had difficulty expressing oral directions and relied on the arrows to communicate. In their verbal communication children often used imprecise language to describe movement (e.g., turn down, turn up, go straight) rather than turn left or turn right and forward (Fessakis et al., 2013). Similarly, in a study with 2–4-year-old children, Critten, Hagon and Messer (2021) found that younger children did not understand left and right rotations. They used the word 'turn' and gestured with hand directions. The children in Critten et al.'s study experienced difficulty with what we refer to as *space-symbol coordination*, which involves knowing how the program or movement corresponds to the domain in which the agent moves around, whether it is a map, screen, grid, or the real-world. Findings from previous studies align with findings in the present study in that the concepts behind the indicators reported here as 'errors' or in other studies as 'difficulties,' are presented as concepts that children came to understand over time by engaging in CT tasks. For example, in a study with children aged 3–4.5, Palmér (2017) found that after 4 months, the children understood what we refer to as the *spatial code meanings, one-to-one code to movement correspondence*, and *space-symbol coordination.*

As we saw in the examples above, children needed knowledge of: *code-to-movement correspondence*, knowledge that one code produces a single discrete movement; *spatial code meanings*, knowing what each code means and does; *spatial orientation of the robot*, knowing that the codes always produce the same movement but depend on the agent's orientation; and *space-symbol-coordination*, knowing how the program domain corresponds to the domain in which the agent moves around. We also identified mathematical knowledge, skills, and abilities that influence their performance; *counting-on, rotation on a point, linear units of measurement*, and *spatial reasoning* (Shumway, Welch, Kozlowski, Clarke-Midura, & Lee, 2021). Appendix A shows how the indicators map on to the prior knowledge, skills, and abilities.

These CT skills and related prior knowledge are similar across the toys we used in our study and also apply to other coding environments designed for pre-literate children, including board games designed to teach young children CT (Poole, Clarke-Midura, Rasmussen, Shehzad, & Lee, 2021). It is important to note that other coding contexts and toys have the potential to foster a wider range of CT skills such as repeat loops and conditionals (Bers, 2019) and math skills such as patterning, unitizing, and more advanced iterating.

4.2. Design implications for formative assessments

Designing formative assessment tasks of variable complexity to inform instruction requires first understanding the knowledge and skills being assessed. We have described what some of these key knowledge indicators are and what common observables may be, such as gesture, manipulating materials, and using imprecise language. These modes of responding during tasks provide evidence for interpreting students CT knowledge in refinement. However, rather than discreet pieces of evidence that map neatly onto algorithmic thinking, debugging, or abstraction per se, children's demonstrations of knowledge often co-implicate multiple CT knowledge domains. For designers of formative assessment tasks, the challenge becomes deciphering from a single indicator, exactly which knowledge is co-implicated. In other words, when children like those in *Wrong Turn* Example C (Fig. 5) indicate a wrong turn, there are potentially multiple CT components implicated. Figuring out whether this error is due to (a) a developing understanding of *spatial reasoning* (i.e., understanding the robot's orientation determines the rotation) (b) a developing understanding of *one-to-one correspondence* (i.e., understanding that the unit of a turn requires a rotation +forward), or (c) a developing understanding of *code meanings* (i.e., understanding that the turn arrow simply rotates it) or (d) some combination of these skills involves interpreting the child's demonstrated knowledge in context of the formative task. In these design spaces, tasks have many moving parts, thus we encourage the combined use of our indicator set with design patterns for variable features of formative tasks as loose task models, rather than straightforward means of inferring a child's knowledge.

Our approach for developing formative assessment tasks based on KRO indicators and observing how children respond to variable features of tasks underlies the importance of scaffolding and the need for teachers to support learning in each child's zone of proximal development (ZPD). Vygotsky wrote that "a child's ZPD is the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined under adult guidance or in collaboration with more capable peers" (Vygotsky, 1978, p. 86). This view of formative assessments taking place in the child's ZPD requires teachers to observe children during activities and notice the errors they make in order to understand the kinds of assistance that will help their learning develop. For example, a teacher may observe that a child is struggling to select a correct rotation code when the robot does not share their orientation (e.g., because the robot starts facing them, they want to program it to turn right, rather than left) (see Wang et al., 2021 and Flood, Wang & Sheridan, 2022 for rich pedagogical examples of this particular piece of knowledge in refinement). Scaffolded assistance could consist of introducing several short tasks or paths that start from the child's perspective, progressively scaffolding them to assume the robot's starting perspective. However, any path design that involves a rotation will eventually position the robot at some non-zero angle relative to the child; therefore, anticipating where children will need to "see from" during different stages of tasks is part of modifying consequential variable feature such as starting position or numbers of turns. Alternatively, teachers may choose to eliminate rotations altogether at first, and focus on changing orientation along linear paths and vary the starting position or path length. Our framework and task models provide an accessible way for teachers to know which forms of assistance to use when working with children within their ZPD.

As we have highlighted, one great affordance of the 3D grid-agent toy paradigm for early CT development is the multimodal means children have of demonstrating their developing understanding. Multimodal responses multiply the ways children have of conveying their thinking, which is particularly advantageous given the co-implication of domains and lamination of variable features in a task. A final implication for teachers and tasks designers involves the need to filter out or alternatively, to target, some response modalities in some tasks. In other words, when asking questions is simply not enough- because children are still developing the precise language of coding or because they are still emerging English language learners- we encourage the use of gesture, object manipulation, or embodying the robot as a form of evidence that supports making valid inferences about a child's knowledge during formative assessments. Future research is needed that explores these different modes of response and the strategies children use to better understand the relationship between what children say, what they do, and what they are learning during assessment tasks, both formative and summative.

4.3. Limitations and future directions

The major limitations of the present study are the small sample size and the focus on coding environments designed for pre-literate children. While our findings align with those by Fessakis et al. (2013) around a LOGO-based environment, more research should be conducted with larger samples of children on how these indicators, contextual proficiencies, and math knowledge relate to other screen-based coding environments like Scratch Jr. Although, it is important to note that *the process of identifying indicators through the errors children make and then using the indicators to develop formative assessment tasks could apply to any coding environment.*

Our principled approach and the identification of indicators has the potential to provide insight into very early understandings of what levels of CT may look like in early childhood in the context of coding environments designed for pre-literate children. Kindergarten is often a period of rapid developmental and cognitive growth, where numeracy and literacy skills are emergent and involve making meaning of symbols (i.e., numerals and letters). Future research should explore the relationship of symbolic-type contextual proficiencies and mathematical knowledge with CT facets across children at different developmental levels. Further research should also examine whether the indicators, contextual proficiencies, and mathematical knowledge we have identified as important to kindergarten children's CT understanding are bound to environments designed for pre-literate children or if they are important in other coding environments.

Conclusion

While work is underway to develop assessments that can be used for summative purposes (e.g., Clarke-Midura, Silvis, Shumway, Lee, & Kozlowski, 2021; de Ruiter & Bers, 2021; Relkin & Bers, 2021; Na & Clarke-Midura, 2023), the long-term goal of building a coherent system of assessments for early childhood CT requires a range of assessments be created. In particular, there is a need to understand how to design formative assessments that can be used *during* classroom instruction to provide feedback to teachers and children in real-time. We offer a framework for conceptualizing the design of formative assessments that starts with observing the errors children make, that we refer to as indicators of knowledge in refinement. We combine this with the ECD framework, a principled approach for designing assessments as evidentiary arguments. The process in which we identify indicators and link them to our tasks and cognitive model aids us in refining our operationalization of early childhood CT in the context of coding environments designed for pre-literate children. It is also allowing us to create a coherent assessment system in which our curricular tasks, formative assessments, and summative assessment tasks are all aligned.

Theoretically, we advance two arguments with this paper: (1) an argument for looking at indicators of knowledge refinement opportunities that are manifest in children's coding errors and analyzing them for continuities in the broader spectrum of developing CT competencies and (2) a formative assessment development process to help the field of early childhood education advance ways of thinking about CT. By identifying these indicators and how they can be integrated into formative assessment practices and lessons, we hope to make the learning of computational thinking in early childhood more accessible to children and to teachers alike.

Data availability

Data will be made available on request.

CRediT authorship contribution statement

Jody Clarke-Midura: Conceptualization, Methodology, Investigation, Writing – original draft. **Victor R. Lee:** Investigation, Writing – review & editing. **Jessica F. Shumway:** Investigation, Writing – review & editing. **Deborah Silvis:** Investigation, Writing – review & editing, Visualization. **Joseph S. Kozlowski:** Investigation, Formal analysis. **Rebecca Peterson:** Investigation, Data curation.

Acknowledgments

This research was supported by the National Science Foundation (grant #1842116) and a Utah State University Research Catalyst grant.

Appendix A

Table of Indicators of Knowledge Refinement Opportunities with Definitions and Links to Emerging Understanding

Indicators of Knowledge Refinement Opportunities	Definition	Emerging understanding of….
Unit of Turn	The child believes that a turn arrow moves the robot forward. This is seen when a child thinks a turn arrow rotates the robot and then moves or when they believe that the arrow represents a cardinal direction and they move it accordingly.	*Spatial Code Meanings*- Knowing what each of the codes instructs the agent do (i.e. semantics) *Code-to-Movement Correspondence*- Knowing that one code (i.e. arrow) produces a single discreet movement either forward, backward, rotating right, rotating left *Spatial Reasoning*- Knowing how the agent moves in 3 dimensions and thinking about them in different positions and orientations *Rotation on a Point*- Knowing that an agent's rotation occurs by rotating on a fixed point at a set angle, not translating to an adjacent point
Wrong Rotation	The child chooses the wrong turn arrow for the program or rotates the robot the wrong direction (compared to the turn arrow selected).	*Spatial Code Meanings*- Knowing what each of the codes instructs the agent do (i.e. semantics) *Spatial Reasoning*- Knowing how the agent moves in 3 dimensions and thinking about them in different positions and orientations
Includes Starting Space	The child includes the starting spaces when counting the number of forwards needed in a program. They count squares and not movements. Usually this is what is occurring every time children have one too many forwards in a section of their program.	*Counting on*- Knowing that you do not include the starting location when counting forward movements, or that the start space is "zero" *Linear Units*- Knowing how to use a standard unit of measure to make the agent travel along a linear path
Distance of Arrow	The child doesn't understand how far one arrow moves the robot. This is often seen when children are being imprecise in how they move the robot or when they make a counting error that is not one less (spaces between) or one more (includes starting space).	*Spatial Code Meanings*- Knowing what each of the codes instructs the agent do (i.e. semantics). *Code-to-Movement Correspondence*- Knowing that one code (i.e. arrow) produces a single discreet movement either forward, backward, rotating right, rotating left *Sequencing*- Knowing how to sort and arrange codes and also how to use number sequencing to determine how many movements or codes are needed *Linear Units*- Knowing how to use a standard unit of measure to make the agent travel along a linear path
Spaces Between	The child doesn't count the beginning or ending space, only the squares between the two. This is a subset of Distance of Arrow because the child thinks that one arrow makes the robot go farther than it really does. Usually this is what is occurring every time children have one too few forwards in a section of their program.	*Spatial Code Meanings*- Knowing what each of the codes instructs the agent do (i.e. semantics). *Code-to-Movement Correspondence*- Knowing that one code (i.e. arrow) produces a single discreet movement either forward, backward, rotating right, rotating left *Sequencing*- Knowing how to sort and arrange codes and also how to use number sequencing to determine how many movements or codes are needed *Linear Units*- Knowing how to use a standard unit of measure to make the agent travel along a linear path
No Turn Arrows	The child builds a program of all forward arrows when there is at least one turn marked or needed to get to the goal.	*Spatial Code Meanings*- Knowing what each of the codes instructs the agent do (i.e. semantics) *Spatial Orientation of robot*- Knowing that the codes always produce the same movements but depend on the agent's orientation *Spatial Reasoning*- Knowing how the agent moves in 3 dimensions and thinking about them in different positions and orientations *Rotation on a Point*- Knowing that an agent's rotation occurs by rotating on a fixed point at a set angle, not translating to an adjacent point
Corresponds to Direction not Movement (CTDNM)	The child rotates the robot to face the cardinal direction of the arrow.	*Spatial Code Meanings*- Knowing what each of the codes instructs the agent do (i.e. semantics) *Spatial Orientation of robot*- Knowing that the codes always produce the same movements but depend on the agent's orientation *Space-symbol Coordination*- Knowing how the program domain corresponds to the domain in which the agent moves around (i.e. how codes or specific parts of programs correspond to specific parts of paths traveled by the agent) *Spatial Reasoning*- Knowing how the agent moves in 3 dimensions and thinking about them in different positions and orientations *Rotation on a Point*- Knowing that an agent's rotation occurs by rotating on a fixed point at a set angle, not translating to an adjacent point
Use of Backward	The child uses a backward arrow incorrectly in a program when one shouldn't be needed at all. This is often seen when the child believes a backwards arrow will do something different than it does or when the child is placing seemingly random arrows on the program.	*Spatial Code Meanings*- Knowing what each of the codes instructs the agent do (i.e. semantics) *Spatial Orientation of robot*- Knowing that the codes always produce the same movements but depend on the agent's orientation

(*continued on next page*)

(*continued*)

Indicators of Knowledge Refinement Opportunities	Definition	Emerging understanding of….
Incorrect Sequencing	The child places arrows in the wrong sequence on the program organizer. Either they have all the right arrows and place them in a different order, or they use the program boards to incorrectly build their program, for example, starting from the right to the left.	*Sequencing*- Knowing how to sort and arrange codes and also how to use number sequencing to determine how many movements or codes are needed
Orientation of Arrows = Movement	The child orients the arrows differently in the program, which affects what they believe the arrow will make the robot do.	*Spatial Code Meanings*- Knowing what each of the codes instructs the agent do (i.e. semantics) *Spatial Orientation of robot*- Knowing that the codes always produce the same movements but depend on the agent's orientation *Spatial Reasoning*- Knowing how the agent moves in 3 dimensions and thinking about them in different positions and orientations *Rotation on a Point*- Knowing that an agent's rotation occurs by rotating on a fixed point at a set angle, not translating to an adjacent point
Diagonal Line	The child believes the robot can move diagonally across squares to get to a goal.	*Spatial Code Meanings*- Knowing what each of the codes instructs the agent do (i.e. semantics) *Spatial Reasoning*- Knowing how the agent moves in 3 dimensions and thinking about them in different positions and orientations *Rotation on a Point*- Knowing that an agent's rotation occurs by rotating on a fixed point at a set angle, not translating to an adjacent point
Disregards Starting Orientation	The child writes a program as if the starting orientation of the robot is facing the goal (regardless of which way the robot is actually facing).	*Spatial Code Meanings*- Knowing what each of the codes instructs the agent do (i.e. semantics) *Spatial Orientation of robot*- Knowing that the codes always produce the same movements but depend on the agent's orientation *Spatial Reasoning*- Knowing how the agent moves in 3 dimensions and thinking about them in different positions and orientations *Rotation on a Point*- Knowing that an agent's rotation occurs by rotating on a fixed point at a set angle, not translating to an adjacent point

References

Bers, M. U. (2019). Coding as another language: A pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education, 6*(4), 499–528.

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

Black, P., & Wiliam, D. (2018). Classroom assessment and pedagogy. *Assessment in Education: Principles, Policy & Practice, 25*(6), 551–575.

Clarke-Midura, J., Silvis, D., Shumway, J. F., Lee, V. R., & Kozlowski, J. S. (2021). Developing a kindergarten computational thinking assessment using evidence-centered design: the case of algorithmic thinking. *Computer Science Education, 31*(2), 117–140.

Common Core State Standards Initiative [CCSSI]. (2010). *Common Core State Standards for Mathematics*. http://www.corestandards.org/Math/.

Critten, V., Hagon, H., & Messer, D. (2021). Can pre-school children learn programming and coding through guided play activities? A case study in computational thinking. *Early Childhood Education Journal*, 1–13.

Cutumisu, M., Adams, C., & Lu, C. (2019). A scoping review of empirical research on recent computational thinking assessments. *Journal of Science Education and Technology, 28*(6), 651–676.

de Ruiter, L. E., & Bers, M. U. (2021). The Coding Stages Assessment: Development and validation of an instrument for assessing young children's proficiency in the ScratchJr programming language. *Computer Science Education*. 10.1080/08993408.2021.1956216.

DeLiema, D., Dahn, M., Flood, V. J., Abrahamson, D., Enyedy, N., & Steen, F. (2020). Debugging as a context for collaborative reflection on problem-solving processes. In E. Manolo (Ed.), *Deeper learning, dialogic learning, and critical thinking: Research-based strategies for the classroom* (pp. 209–228). New York, NY: Routledge.

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

Flood, V. J., Wang, X. C., & Sheridan, M. (2022). Embodied responsive teaching for supporting computational thinking in early childhood. In C. Chinn, E. Tan, C. Chan, & Y. Kali (Eds.), *Proceedings of the 16th international conference of the learning sciences - ICLS 2022* (pp. 855–862). Hiroshima, Japan: International Society of the Learning Sciences.

Govind, M., & Bers, M. (2021). Assessing robotics skills in early childhood: Development and testing of a tool for evaluating children's projects. *Journal of Research in STEM Education, 7*(1).

Hammer, D (2000). Student resources for learning introductory physics. *American journal of physics, 68*(S1), S52–S59.

Herman, J. L. (2010). *Coherence: Key to next generation assessment success (AACC report)*. Los Angeles, CA: University of California.

Kane, M. (2006). Content-related validity evidence in test development. *Handbook of test development, 1*, 131–153.

Mislevy, R. J. (2007). Validity by design. *Educational researcher, 36*(8), 463–469.

Mislevy, R. J., & Haertel, G. D. (2006). Implications of evidence-centered design for educational testing. *Educational Measurement: Issues and practice, 25*(4), 6–20.

Na, C, & Clarke-Midura, J (2023). Assessing young children's computational thinking using cognitive diagnostic modeling. In *Proceedings of the 17th International Conference of the Learning Sciences - ICLS 2023*. International Society of the Learning Sciences.

National Research Council (US). (2006). Committee on support for thinking spatially. *Learning to think spatially: Gis as a support system in the K-12 curriculum*. National Academy Press.

Oliveri, M. E., Lawless, R., & Mislevy, R. J. (2019). Using evidence-centered design to support the development of culturally and linguistically sensitive collaborative problem-solving assessments. *International Journal of Testing, 19*(3), 270–300.

Palmér, H. (2017). Programming in preschool–with a focus on learning mathematics. *International Research in Early Childhood Education, 8*(1), 75–87.

Papert, S. (1972). Teaching children to be mathematicians versus teaching about mathematics. *International journal of mathematical education in science and technology, 3*(3), 249–262.

Papert, S. (1980). *Children, computers, and powerful ideas*. Harvester Press (Unitend Kingdom).

Papert, S (2000). What's the big idea? Toward a pedagogy of idea power. *IBM systems journal, 39*(3.4), 720–729.

Patton, M. Q. (2014). *Qualitative research & evaluation methods: Integrating theory and practice*. Sage publications.

Posner, G. J., Strike, K. A., Hewson, P. W., & Gertzog, W. A. (1982). Accommodation of a scientific conception: Toward a theory of conceptual change. *Science Education, 66*(2), 211–227.

Poole, F, Clarke-Midura, J, Rasmussen, M, Shehzad, U, & Lee, V. R (2021). Tabletop games designed to promote computational thinking. *Computer Science Education, 32*(4), 449–475.

Primo, Toys. https://www.primotoys.com/.

Relkin, E., & Bers, M. U. (2019). Designing an assessment of computational thinking abilities for young children. In L. E. Cohen, & & S. Waite-Stupiansky (Eds.), *STEM for early childhood learners: How science, technology, engineering and mathematics strengthen learning* (pp. 85–98). New York, NY: Routledge.

Relkin, E., & Bers, M. (2021). TechCheck-K: A measure of computational thinking for kindergarten children. In *2021 IEEE global engineering education conference (EDUCON)* (pp. 1696–1702). IEEE. 10.1109/EDUCON46332.2021.9453926.

Relkin, E., de Ruiter, L., & Bers, M. U. (2020). TechCheck: Development and validation of an unplugged assessment of computational thinking in early childhood education. *Journal of Science Education and Technology*. 10.1007/s10956-020-09831-x.

Rich, K. M., & Yadav, A. (2020). Applying levels of abstraction to mathematics word problems. *TechTrends, 64*(3), 395–403.

Roberts, R. J., Jr, & Aman, C. J. (1993). Developmental differences in giving directions: Spatial frames of reference and mental rotation. *Child development, 64*(4), 1258–1270.

Saldaña, J. (2021). *The coding manual for qualitative researchers*. Thousand oaks, CA: Sage publications.

Sarama, J., & Clements, D. H. (2009). *Early childhood mathematics education research: Learning trajectories for young children*. Routledge.

Shumway, J. F., Welch, L. E., Kozlowski, J. S., Clarke-Midura, J., & Lee, V. R. (2021). Kindergarten students' mathematics knowledge at work: the mathematics for programming robot toys. *Mathematical Thinking and Learning*, 1–29. 10.1080/10986065.2021.1982666.

Smith, J. P., III, Disessa, A. A., & Roschelle, J. (1994). Misconceptions reconceived: A constructivist analysis of knowledge in transition. *The Journal of the Learning Sciences, 3*(2), 115–163.

Strauss, A., & Corbin, J. (1998). *Basics of qualitative research techniques* (pp. 1–312). Sage publications.

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education, 148*, Article 103798.

Tippett, C. D. (2010). Refutation text in science education: A review of two decades of research. *International Journal of Science and Mathematics Education, 8*(6), 951–970. 10.1007/s10763-010-9203-x.

Tsarava, K., Moeller, K., Román-González, M., Golle, J., Leifheit, L., Butz, M. V. ., et al., (2022). A cognitive definition of computational thinking in primary education. *Computers & Education*. 10.1016/j.compedu.2021.104425.

Vacha-Haase, T. (1998). Reliability generalization: Exploring variance in measurement error affecting score reliability across studies. *Educational and Psychological Measurement, 58*(1), 6–20.

Vygotsky, L. S. (1978). Mind and society: The development of higher psychological processes. Cambridge, MA: Harvard University Press.

Wang, X. C., Choi, Y., Benson, K., Eggleston, C., & Weber, D. (2021). Teacher's role in fostering preschoolers' computational thinking: An exploratory case study. *Early Education and Development, 32*(1), 26–48.

Weintrop, D., Wise Rutstein, D., Bienkowski, M., & McGee, S. (2021). Assessing computational thinking: An overview of the field. *Computer Science Education, 31*(2), 113–116.

Whiteley, W., Sinclair, N., & Davis, B. (2015). What is spatial reasoning?. In B. Davis (Ed.), *Spatial reasoning in the early years: Principles, assertions, and speculations* (pp. 3–14). Routledge.