Educ Stud Math (2009) 72:17–37
DOI 10.1007/s10649-008-9180-y

Encrypted objects and decryption processes: problem-solving with functions in a learning environment based on cryptography

Tobin White

Published online: 22 January 2009

Abstract This paper introduces an applied problem-solving task, set in the context of cryptography and embedded in a network of computer-based tools. This designed learning environment engaged students in a series of collaborative problem-solving activities intended to introduce the topic of functions through a set of linked representations. In a classroom-based study, students were asked to imagine themselves as cryptanalysts, and to collaborate with the other members of their small group on a series of increasingly difficult problem-solving tasks over several sessions. These tasks involved decrypting text messages that had been encrypted using polynomial functions as substitution ciphers. Drawing on the distinction between viewing functions as processes and as objects, the paper presents a detailed analysis of two groups' developing fluency with regard to these tasks, and of the aspects of the function concept underlying their problem-solving approaches. Results of this study indicated that different levels of expertise with regard to the task environment reflected and required different aspects of functions, and thus represented distinct opportunities to engage those different aspects of the function concept.

Keywords Functions · Multiple representations · Problem solving · Technology · Context

1 Introduction

Applied and 'real-world' problem-solving contexts can provide opportunities for meaningful engagement of rich mathematical ideas (Boaler, 1993). Similarly, technology-rich learning environments can present students with dynamic representations of real-world phenomena and authentic settings for applied tasks, as well as powerful problem-solving tools (Kieran & Yerushalmy, 2004; Roschelle, Pea, Hoadley, Gordin, & Means, 2000). While the promise of these tasks and technologies for supporting student learning is considerable, the very open-endedness of the problem-solving process and the richness of

T. White (✉)
School of Education, University of California, Davis, 2043 Academic Surge Building,
One Shields Avenue, Davis, CA 95616, USA
e-mail: twhite@ucdavis.edu

the relevant mathematics can make the particular concepts students might engage, and the nature of that engagement, difficult to specify.

This paper seeks to detail the specific links between a curricular concept and a set of problem-solving processes as those links emerge from learners' repeated engagement with a particular task. I introduce a set of applied problem-solving tasks, situated in the context of cryptography and embedded in a network of computer-based tools, and explore the strategies students brought to bear in solving those tasks in relation to the relevant mathematics concept, namely function. Working in collaborative small groups, students undertook a series of similar but increasingly difficult tasks in this setting over several sessions. The paper presents a detailed analysis of two groups' developing fluency with regard to these tasks, and of the aspects of the function concept underlying their problem-solving approaches. The resulting account suggests important links between the mathematical features of the applied context, the technical and representational features of the problem-solving environment, and the specific conceptual demands of the assigned task.

2 Reifying functions in multi-representational environments

Understanding of functions is often characterized in terms of the distinction between *processes* and *objects*. Sfard (1991), for example, distinguishes between a structural perspective on functions as objects comprising sets of ordered pairs, and an operational perspective emphasizing the role of functions as processes for mapping an input value to an output. While contemporary mathematics texts emphasize the structural account, Sfard argues that the operational aspects should be considered complementary, and may be more salient or relevant in certain contexts. Drawing on a number of historical examples to show that operational conceptions tend to precede structural ones in the development of mathematical concepts, she argues that new mathematical objects can be seen as resulting from the *reification* of processes involving previously known objects.

Extending this epistemology of mathematical objects to the realm of concept formation in individual learners, Sfard (1992) proposes that students might be expected to encounter and grasp operational aspects of functions first, and only gradually see those operations reify into abstract structures, or objects, which might themselves undergo operations. Indeed, researchers have found structural conceptions to be particularly elusive, among even advanced high school and college algebra students (Kaput, 1992; Kieran, 1992; Sfard, 1992; O'Callaghan, 1998). While that shortcoming may follow from difficulties inherent in grasping the structural view, or the necessary ordering of understanding functions as processes prior to objects, it may also reflect an emphasis on operational aspects of function in many students' curricular experience. Moreover, as Confrey and Smith (1994) argue, these deficit models of learners' understanding of functions may be "indicative of a narrow view of the function concept, bounded by its placement in a decontextualized and heavily abstract view of mathematical systems" (p. 137). These authors show how concepts such as rate of change, especially when applied across a range of real-world problem contexts and through multiple representations, can support the emergence of a rich and diverse array of learner conceptions about function.

Similarly, other investigations of student function understanding in a variety of computer-based environments featuring multiple linked representations indicate that appropriately structured learning experiences might support the development of structural conceptions independently of operational fluency (Schwarz & Dreyfus, 1995; Slavit, 1997).

Indeed, different representations of function afford different perspectives; while symbolic representations of functions tend to emphasize process aspects, other representations such as graphs may make the object-like properties of such functions particularly salient (Schwartz & Yerushalmy, 1992). In this vein, Moschkovich, Schoenfeld and Arcavi (1993) make the case that rich problem contexts encourage or require students to alternately engage both object and process perspectives on functions. Moreover, such tasks should encourage use of and connections among several function representations, including symbols, tables and graphs as well as real-world contexts and verbal descriptions. The authors argue that understanding of functions might be envisioned in terms of a learner's ability to flexibly move in a grid spanned along one dimension by the process and object perspectives, and along another by an array of representational modes.

Thompson (1994), however, cautions against drawing hasty conclusions about the nature of connections drawn between different forms of representational activity. He argues that while a knowledgeable adult may perceive a mathematical object such as a function being represented by, for example, a table or graph, a student considering the same representation will not necessarily recognize the same object, let alone the continuity of that object across other representations (Thompson & Sfard, 1994). Instead of cultivating an object conception of functions by highlighting the invariance of those objects across different domains, attempts to engage learners with multiple representations of functions may simply leave them with disconnected conceptions of those various representations. Rather than focusing on abstract objects, instructional activities involving multiple representations should emphasize aspects of specific situations that students might perceive across different representational forms (Thompson, 1994).

In a similar spirit, this paper presents a learning environment designed to ground the concept of function in the context of an applied problem-solving situation supported by a set of multiple linked representational tools. In the remainder of this paper, I explore the ways participating in open-ended problem-solving activities in this setting afforded opportunity for students to engage functions as both processes and objects. The next sections introduce the learning environment and describe results from its implementation in a classroom-based design experiment (Brown, 1992; Collins, 1992). I present an analysis of the ways different strategies students brought to bear on the tasks variously emphasized operational and structural features of functions. This study seeks to address two central research questions:

1. In what ways do students' strategies for solving problems in an applied context make use of properties of functions as processes and objects?
2. To what extent are learners' developing strategies for solving those problems compatible with a theory of reification?

3 The *Code Breaker* learning environment

This paper draws on data collected during the first classroom implementation of a curricular unit situated in a classroom network of wireless handheld computers, and set in the applied context of cryptography. This designed learning environment engaged students in a series of collaborative problem-solving activities intended to introduce the topic of functions across a variety of representations. In particular, this environment was organized around functions as substitution ciphers—codes based on polynomial functions that map the letters in the alphabet to numerical ciphertext characters in an encrypted message, and problem-solving tasks centered on using multiple representations of functions to analyze and break those codes. Students were asked to imagine themselves as cryptanalysts, and to collaborate with the other members of their small group to decrypt these coded messages. Substitution ciphers such as the ones used in this study played a central role in early code-breaking history—one, for instance, was so extensively and so famously used by Julius Caesar that it is now commonly referred to as the “Caesar cipher” (Singh, 1999). Though far too easily broken to be used in contemporary cryptography, substitution ciphers provided a rich context for developing an introductory unit on functions and motivating students' engagement with problem-solving tasks.

To generate these codes, each letter in a plaintext message was assigned to its ordinal alphabetic value between 1 and 26, and then mapped through a polynomial ‘encoding function’ to produce its corresponding output value in a numerical ciphertext alphabet. Groups were presented with ciphertext messages that had been encrypted in this way, and charged with determining the encoding function in order to decipher the message. In addition to the polynomial mapping, some codes also featured an “offset.” Offsets shifted the relationship between the letters of the alphabet and their ordinal values, so that while an offset of zero meant that A was associated with an input value of one, an offset of one associated A with an input of 2, B with 3, and so on, including associating Z with 1. Codes featuring offsets were introduced approximately halfway through the 3-week series of decryption activities, as part of an effort to steadily increase the difficulty of the tasks.

Each student was equipped with a handheld computer running a software application called *Code Breaker*, which provided a suite of tools—graphs, function tables, letter and word frequency charts—for use in the decryption tasks. Groups used these tools to generate and examine ‘candidate’ functions, and to compare the image of these functions with the characters in the ciphertext message, in order to match one such candidate to the encoding function. The classroom network linked the devices of each student in a group so that changes to the candidate function on one student's computer automatically propagated to the devices of the other group members. Consequently, though a single device could display only one or two of those tools at a time, a group could collectively examine the full array of representations simultaneously. Four of these representational tools were linked to the candidate function—an algebraic expression, a graph, and tabular displays of both that candidate function and its inverse. Three other displays pertained to the cryptographic context in which these functions were situated, providing information about the unknown encoding function as well as aspects of the encrypted text message. These seven tools are described below in accordance with four representational modes: algebraic, graphical, tabular, and situational. I then outline the ways student groups might employ these tools to break codes based on polynomial functions.

3.1 Algebraic

One student in a group was assigned responsibility, rotated daily, for viewing and editing a symbolic expression for the candidate function. Encoding and candidate functions were of the form $y=ax^b+c$, where c could be any integer, a any nonzero integer, and b could equal to 1, 2, or 3. Candidate parameters were adjusted from their default settings ($a=1$, $b=1$, $c=0$) by tapping with a stylus on either the top or the bottom half of the number on the handheld screen, causing the value to increment or decrement by one unit. As the designated student made these changes to the algebraic candidate, the corresponding graphical and tabular displays updated accordingly on each group member's device.

3.2 Graphical

The *Code Breaker* graph (Fig. 1) displayed the current state of the candidate function in a window scaled in accordance with the encoding function. The *x*-axis of the graphing window was always fixed to the alphabetic domain, spanning from 0 to 26. The *y*-axis, on the other hand, adjusted automatically around the range of values included in the coded text message. Each of those cipher text values was also represented in the graph by a horizontal 'trace line' stretching from the *y*-axis until it intersected the candidate curve. A corresponding vertical line, drawn from this intersection to the *x*-axis, reflected the mapping of the cipher text value through the inverse of the candidate function. When the output of this inverse mapping was an integer between one and twenty-six, students could click on the associated letter and highlight the trace lines and the ordered pair they connected. By highlighting the ordered pairings associated with a particular candidate function, these lines were intended to serve as both a code-breaking resource, and as a scaffold to students' understanding of the functional relationship between the input and output values of the code.

3.3 Tabular

The *Code Breaker* software featured two different tabular function representations. The function table (Fig. 2a) paired a static *X*-column, displaying the numbers 1 through 26, with a dynamic *Y*-column that updated with each adjustment of the candidate function to show the new output value to which that polynomial would map each of the *X*-values. The inverse function table (Fig. 2b) displayed the inverse mapping of cipher text values through the candidate function. The *Y*-column of the inverse function table displayed the range of values in an encrypted message, as shown in Fig. 2b for an encoding function of $y=x^2+7$. Each of these cipher text values was then mapped through the inverse of the current candidate function ($y=2x^2+6$, in the example of Fig. 2b), with the result displayed in the

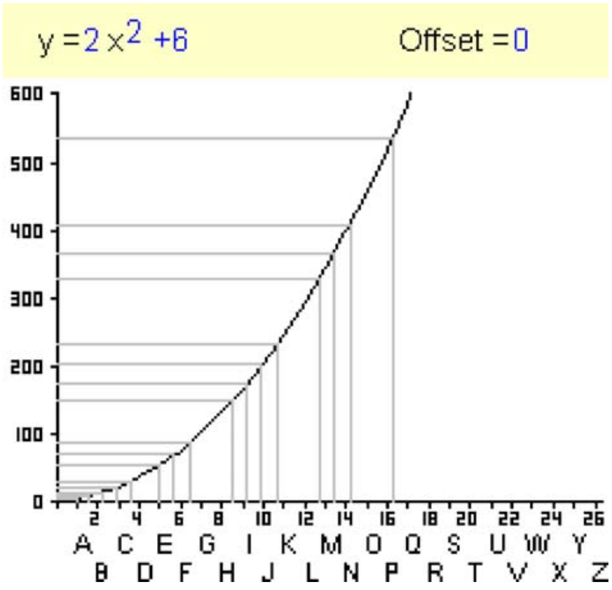
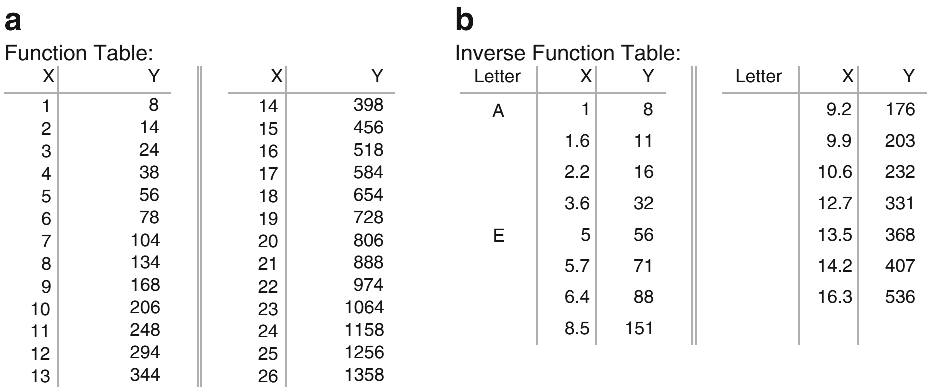


Fig. 1 The *Code Breaker* graph



a

Function Table:

X	Y
1	8
2	14
3	24
4	38
5	56
6	78
7	104
8	134
9	168
10	206
11	248
12	294
13	344
14	398
15	456
16	518
17	584
18	654
19	728
20	806
21	888
22	974
23	1064
24	1158
25	1256
26	1358

b

Inverse Function Table:

Letter	X	Y
A	1	8
	1.6	11
	2.2	16
	3.6	32
E	5	56
	5.7	71
	6.4	88
	8.5	151
	9.2	176
	9.9	203
	10.6	232
	12.7	331
	13.5	368
	14.2	407
	16.3	536

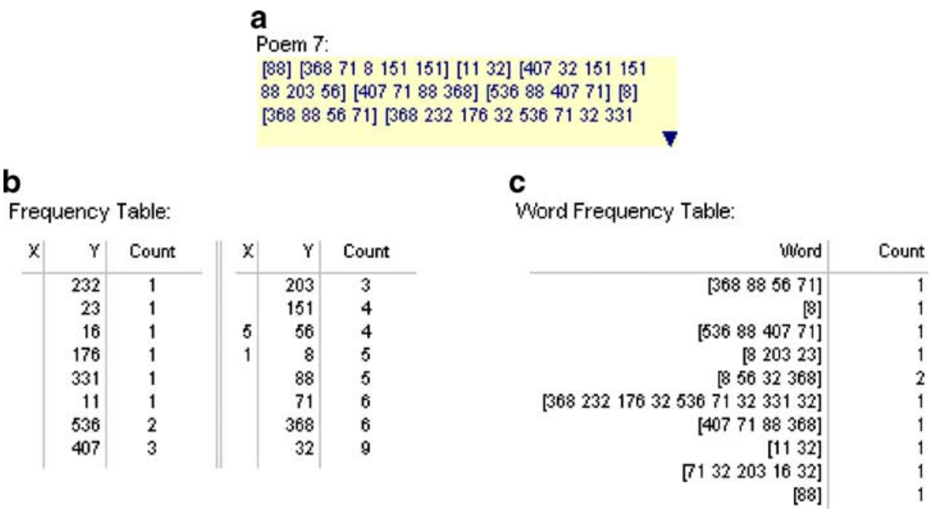
Fig. 2 **a** and **b**. The function and inverse function tables

corresponding cell of the *X*-column. When this process yielded an integer from 1 to 26, the corresponding plaintext letter appeared in the "Letter" column.

Partway through the implementation, students and researchers together discovered a significant bug associated with the inverse function table; concern about both this bug and the transparency of the representation prompted researchers to replace it with the function table for the remainder of the study (for a detailed account of this discovery and subsequent software revision, see White, 2008).

3.4 Situational

One *Code Breaker* representation displayed the cipher text itself, shown as a series of numbers comprising the encoded letters of the encrypted message, with spaces between numbers bounding each letter and brackets around strings of numbers to demarcate words (Fig. 3a). Two additional tables provided other information about the cipher text message under scrutiny. The frequency table (Fig. 3b) displayed each distinct character in the cipher



a

Poem 7:

[88] [368 71 8 151 151] [11 32] [407 32 151 151 88 203 56] [407 71 88 368] [536 88 407 71] [8] [368 88 56 71] [368 232 176 32 536 71 32 331

b

Frequency Table:

X	Y	Count
	232	1
	23	1
	16	1
	176	1
	331	1
	11	1
	536	2
	407	3
	203	3
	151	4
5	56	4
1	8	5
	88	5
	71	6
	368	6
	32	9

c

Word Frequency Table:

Word	Count
[368 88 56 71]	1
[8]	1
[536 88 407 71]	1
[8 203 23]	1
[8 56 32 368]	2
[368 232 176 32 536 71 32 331 32]	1
[407 71 88 368]	1
[11 32]	1
[71 32 203 16 32]	1
[88]	1

Fig. 3 **a**–**c**. Situational Representations

text paired with a count of the number of times that value appeared in the encrypted message. Similarly, the word frequency table (Fig. 3c) displayed the number of occurrences of each distinct word in the coded text.

3.5 Cryptanalysis and functions in the *Code Breaker* learning environment

Generally, decryption strategies in this environment could move in either of two directions—from candidate function to possible cipher text range, or from cipher text values to possible candidate function. The first path involves using the other representational tools to examine how a given candidate maps the plaintext alphabet in relation to the set of cipher text values—how well the graph 'fits' the adjusted window by spanning the domain and range specified by the plain and cipher text sets respectively, or how the function table values compare to those in the situational displays. The other line of approach begins with the values in the cipher text, and involves matching one or more of these with a corresponding plaintext letter to form an ordered pair. For example, a student examining the frequency table shown in Fig. 3b might note that 32 is the most commonly occurring value in the cipher text, and conclude that it was mapped from a plaintext "E". They would then seek out a candidate function that included the ordered pair (5, 32).

The intent behind the *Code Breaker* design was for student groups to use a combination of these approaches—doing so follows from the collaborative context and the networking of multiple devices, as different students in the group can be simultaneously engaged with different linked representational tools. Moreover, while these codes could in principle be broken using only analyses of the situational displays, the other representational tools were quite useful not only for determining the degree and often also the lead coefficient of the polynomial encoding function, but also as scaffolds for students who did not yet have the familiarity with functions or the fluency with algebra to easily or successfully enact such analyses. Below, I examine the ways students' decryption strategies in a classroom study balanced these different approaches, and the extent to which process and object characteristics of functions were salient in each.

4 Method

4.1 Context

The *Code Breaker* handheld environment was implemented with two middle school mathematics classes during a 5-week summer school session. During the first 2 weeks of the summer session, students had lessons on the history and key terminology of cryptography, the principles of substitution ciphers, the evaluation of simple polynomial expressions, coordinate graphing, and the mechanics of the handheld computers and the *Code Breaker* client software. Students spent the remaining 3 weeks collaborating in small groups to make and break codes. As they grew familiar with the different representational artifacts in the *Code Breaker* software, students in each group were encouraged to work together to develop their own strategies for using those resources to solve increasingly challenging decryption problems.

This paper analyzes the problem-solving work of two focus groups, A and B, comprised of four students each. A total of four groups from the two classes were selected as focus groups to be videotaped during all small-group activities for the duration of the study. These four groups were purposively selected according to several criteria, including the consent of all members to be videotaped and interviewed, and informal observations of their work during preliminary activities. Groups A and B were selected from among the four focus groups for more detailed analysis because they again exhibited particularly high levels of on-task discourse during decoding activities.

4.2 Analytic approach

The primary problem-solving activity in these decryption tasks involved using the *Code Breaker* software to identify encoding functions and decrypt messages. This paper analyzes the strategies enacted by groups in these problem-solving tasks in order to characterize the ways they engaged the concept of function. The following analysis examines thirty-two problem-solving episodes undertaken by the two focus groups over the final 3 weeks of the study. These decryption events, ranging from 2 to 30 min in duration, began when a group downloaded a new code to break, and concluded when they either solved or stopped working on the code. Video records of each event were transcribed and reviewed in detail. When applicable, server logs of student decoding activity, researchers' notes, and students' written records were also consulted to supplement the video data.

Each decryption event was examined in detail in order to characterize the strategies groups employed. All instances in which a group correctly determined a parameter of an encoding function, including any of the coefficient a, exponent b, and constant c displayed in the candidate function interface, were categorized in terms of the extent to which they incorporated specific ordered pairs in an encoding function. For the purposes of this analysis, a parameter was deemed "correct" if it (a) matched the corresponding value in the encoding function, (b) produced an equivalent decryption, (c) failed to meet the previous criteria only because of an error such as the misassignment of a plaintext letter's ordinal value, or (d) was determined following the incorrect specification of the other candidate parameters, and combined with those to generate a function that included the ordered pair (s) students specified. So, for example, during event A9, students in Group A9 were credited with determining a constant of 3 after they had incorrectly determined a linear coefficient of 6 and specified an ordered pair (2, 15). The majority of parameter determinations made use only of general features of an encoded message, such as the size and sign of values in the coded text, without considering how the encoding function would map specific input values to specific outputs. Some strategies, however, involved the specification of an ordered pair associated with an encoding function. And a small number of strategies featured the use of multiple ordered pairs in combination. Hereafter, parameter determinations that did not involve the specification of ordered pairs are referred to as "non-ordered pair" strategies. This name is not intended to assess whether students using these approaches recognized an encoding function as a set of ordered pairs, but rather to distinguish these strategies from those in which students specifically identified and utilized one or more ordered pairs to identify a candidate parameter.

The first part of the analysis presented below seeks to characterize these various decryption strategies in detail, and to examine the aspects of the function concept involved in each approach. Fine-grained analyses of illustrative episodes involving inferences from no, one, and multiple ordered pairs are presented in three successive sections. Using the framework outlined by Sfard (1991), I explore the extent to which these decryption strategies may have reflected or relied on structural and operational aspects of the function concept. In particular, I interpret those instances when groups adjusted parameters on the basis of the mapping between sets suggested by a candidate function or drew on characterizations of functions as sets of ordered pairs in order to assess those candidate parameters as utilizing object-like properties of functions. By contrast, I interpret instances when their approach to determining parameters hinged on the numerical evaluation of a candidate expression for a given output as emphasizing process characteristics of functions. These characterizations of groups' strategies are drawn both from the ways students used various software tools and other resources, and from students' descriptions of candidate and encoding functions as they interpreted each across different representational modes and in relation to the decryption task. They are not meant to stand as inferences regarding students' concept images (Tall & Vinner, 1981) of function, but rather as illustrations of the ways different aspects of the function concept may be salient in or relevant to learners' emerging strategies in the context of these novel cryptographic tools and tasks.

The second part of the analysis examines all strategies used by the two groups across all decryption tasks in order to identify patterns in the development of those strategies over time and in relation to increasing task difficulty. Each parameter determination was categorized according to the number of ordered pairs on which it relied, distinguishing among decryption strategies that incorporated none, one, or more than one of the ordered pairs associated with the encoding function. Those different strategies are compared with regard to their relative effectiveness in completely identifying encoding functions, and in determining individual parameters of those functions.

5 Non-ordered pair strategies

Strategies that did not rely on ordered pairs were more frequent and more varied than those that did. The three brief episodes presented in this section serve to illustrate that variety, and to highlight the common ways these different approaches used features of the *Code Breaker* representational tools to engage object-like properties of functions.

5.1 Episode 1

Student groups made extensive use of the software's graphing tool to evaluate candidate function parameters by assessing the fit of various candidate graphs to window dimensions determined by the range of values in the cipher text. As the following segment opened the group's candidate function was set to $y=-x$ (Fig. 4a), and Tina sought to ascertain whether a quadratic function might be a better fit:

1. Tina: Now change the x to squared.
2. Monique: The who?
3. Tina: x to squared.
4. Jessica: Exponent to three. [Monique taps the PDA screen with her stylus twice, increasing the exponent to two and then three.]
5. Tina: Mind you I assume the exponent...which sucks. Ok that's way too big.
6. Shirley: [leaning in close to examine her PDA screen] Whoa.
7. Tina: No, no, no, no. Back, back, back, back back...
8. Monique: Back down?
9. Shirley: What the heck is that?
10. Tina: [As Monique decrements the exponent] Back, back, back, back, back to one, back to one, back to one.
11. Tina: See this obviously, it's obviously not times negative one. So try negative two. [Monique adjusts the lead coefficient to −2.] [Event B11]

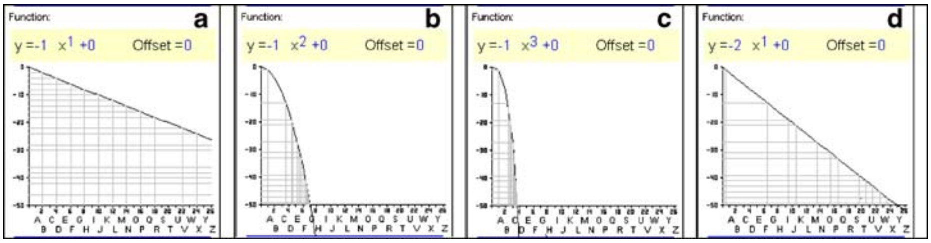


Fig. 4 a–d. Sequence of graphical displays from Event B11

As Monique adjusted the candidate function (Fig. 4b), Tina announced that the exponent was "way too big" (line 5). Following up on Jessica's instruction (line 4), Monique continued increasing the exponent to three (Fig. 4c). Acting on Tina's urgings to change the exponent "back to one," (line 10) Monique edited the candidate to again show the graph of Fig. 4a. Tina then proposed that they "try negative two" for the coefficient (line 11), yielding the graph shown in Fig. 4d.

These examinations of the graph hinged on comparing the range of the numerical values in the message with those produced by the candidate function. The graphical display allowed students to evaluate candidate parameters by making these comparisons visually, so that Shirley reacted with considerable surprise (lines 6 and 9) to the curves displayed in Figs. 4b and c, respectively, and Tina appealed to the obviousness (line 11) of the misalignment shown in Fig. 4a. They drew these inferences without recourse to specific ordered pairings, instead relying on the general trends in the graph that corresponded to each change to the candidate function.

5.2 Episode 2

In fact, the two focus groups almost never discussed specific ordered pairs in the context of the graph despite the fact that the representation included a feature, the 'trace lines', intended to highlight the ordered pairings associated with a particular candidate function. Instead, groups tended to use these lines to draw inferences of the kind illustrated in the following excerpt. As Vince adjusted the candidate parameters, Jason observed the graph, and CJ made a comment to clarify that observation:

Vince: How's that?

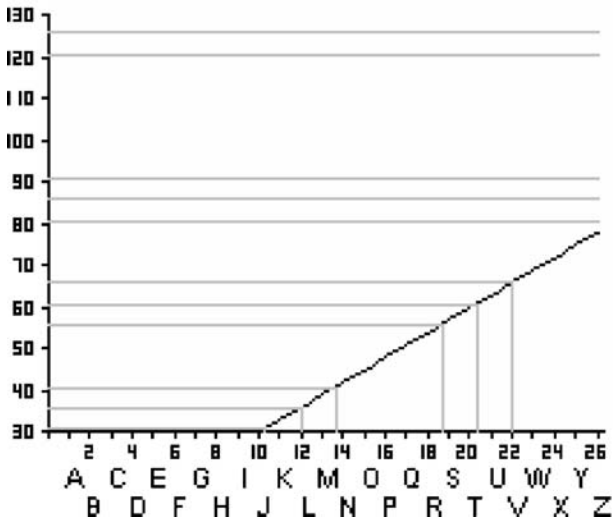
Jason: Terrible.

Jason: There has to be one for each num... for each letter.

CJ: For each letter in the code. [Event A11]

Figure 5 shows the graphing window Group A was looking at in this segment. Jason asserted that there should be "one" input, represented by a vertical trace line, to complement "each letter" displayed as a horizontal line. This candidate graph was "terrible" because there were clearly not enough vertical lines to complement all the horizontal lines shown. CJ followed up by reiterating Jason's reference to cipher rather than plaintext characters—the letters "in the code"—and clarifying that there needn't necessarily be twenty-six different vertical trace lines—only enough to match the eleven distinct letters in the encrypted message. Together, the two students recognized the need for one vertical trace line, and so one input value, for each letter-output in the coded message. In effect,

Fig. 5 ‘Trace lines’



they relied on the one-to-one correspondence between the input and output values of an encoding function as a resource for evaluating the candidate Vince had entered.

These groups’ uses of the candidate graph, both as smooth curve in Episode 1 and as set of intersection points between horizontal and vertical trace lines in Episode 2, reflect similar inferences regarding the fit of that object to the encoding mapping represented by the viewing window dimensions. As one student in each group adjusted a candidate parameter, others compared among members of the parameterized class of candidate functions displayed in the graph. In the context of these linked symbolic and graphical representations, seeking a match between a candidate and an encoding function became a process of exploring and evaluating graphical behavior in relation to parametric variation. In other words, the salient relationship, and the one around which the groups oriented their decryption efforts, was that between the candidate parameters and the curve or points as an object to be manipulated by adjustments to those parameters in order to match known features of the encoding function.

5.3 Episode 3

Group A developed another strategy suggestive of an object perspective on functions by capitalizing on the programming error in the inverse function table. For certain linear candidate functions, this bug caused the same letter to appear next to each of two or more coded text values in the inverse function table (Fig. 6). This error went undiscovered by the researchers or the teacher until Group A began employing it as a decoding resource. The following excerpt finds CJ providing a steady series of reports from the inverse function table as Jason adjusted the parameters of the candidate function:

CJ: We have two D’s, two G’s, and two I’s, at the moment. Two B’s, two E’s, two F’s, and two J’s. Ok, two H’s. Three F’s. You’re getting closer.

Teacher: (passing by the group’s table) How do you know you’re getting closer?

CJ: I can tell that when two line up and I’ll have A, A, B, B, C, C, D, D, E, E, F, F, G, G, H, H, that’s bad. We have A, well, A is alone, and H and I and K and certain letters are alone, which is good. We don’t want to have repeats. [Event A4]

Fig. 6 The inverse function table with bug

Letter	X	Y
B	2.1	12
B	2.1	17
C	3	22
C	3.1	32
D	4.1	42
D	4.1	47
D	4.1	52
E	5.1	67
E	5.1	72
F	6.1	82
F	6.1	87
G	7.1	102
H	8	107
H	8.1	112

Identifying as problematic those candidate functions that produced "repeats" in the inverse function table, the boys instead wanted the table to show each letter only once, and attempted to edit the candidate function accordingly. Just as the previous episode found the group seeking a distinct letter input value for each code output, here they similarly recognized and made use of the fact that each of these letters, as input values of the encoding function, should have a unique output. Vince's subsequent explanation of the phenomenon to another classmate elaborates this insight:

> So, we just, if there's like one A, one B, one C, that's good. If there's like two A's, that's bad. Cause there's only one code for an A.

Recognizing that each plaintext letter should map to a single encoded output because "there's only one code for an A," the group attempted to vary candidate parameters in ways that would eliminate these repeated letters. They thus sought functions that established a unique correspondence between sets of plaintext inputs and ciphertext outputs, and rejected candidates that failed to maintain this uniqueness. In doing so, they creatively utilized object-like properties of these correspondences as problem-solving resources.

5.4 Summary: functions as objects

In each of these episodes, groups made changes to candidate function parameters in order to bring the set of candidate output values into alignment with the encoding function outputs. In doing so, they relied on features of the candidate and encoding functions as objects rather than processes. Each of their inferences about those functions emphasized sets of ordered pairs or transformations on a set of values, rather than operations for mapping one element of an ordered pair to the other or for generating the values in the set. The salience of functions-as-objects in these decryption approaches likely reflects particular features of the dynamic links between the symbolic candidate expression and other *Code Breaker* representations. The central action in this environment was that of incremental parameter variation in the algebraic representation, so the corresponding object manipulations, and the objects under those manipulations, in the other representational modes were particularly salient. In effect, these graphical and tabular objects emerged from the dynamic behaviors in these representational modes in concert with actions in the symbolic domain—a complete set of output values in a table or ordered pairs in a graph were simultaneously transformed with each parameter adjustment.

This notion of functions-as-objects constituted by parametric variation echoes the account by Kaput (1992) of students in another computer-based activity similarly organized around determining the rule for an unknown function. Students who were successful in that environment were generally those who had come to see functions in terms of varying parameters, rather than as defined by fixed values of those parameters. In the present case, however, my aim is not to illustrate whether students came to understand the algebraic expression as spanning a family of functions, but rather how they came to *make use* of parametric variation in order to compare among candidate members of that family. Here, such variation was not only a conceptual possibility potentially achieved by some students but also a software feature utilized by all students. In other words, these students used the manipulation of parameters in the symbolic mode as a resource for dynamically comparing and assessing candidate ordered-pair mappings illustrated in the graphical and tabular displays.

6 Inferences from an ordered pair

Prior to the following excerpt, the group had determined by analyzing the ciphertext that the character 40 in that text represented the letter I. In the segment below, they seek a function that would map 9, the ordinal value of I, to 40:

1. Tina: I is nine. And obviously... 9 times 5 would be 45. 45 minus 5...
2. Shirley: We're close.
3. Monique: We've got a long time.
4. Tina: Minus 5 would be it.
5. Tina: But...negative 8 might be A. So 5 minus 5 is 0, that will not work. [Adjusts coefficient to 6].
6. Tina: 9 times 6 is 54. 54 minus 9... [Begins reducing constant]
7. Jessica: I saw a negative 2 in there. [Tina adjusts the constant to 9 and waits for the result]
8. Tina: Hmm...didn't get it right...be 14. [Event B7]

Substituting the input value of 9 into the current candidate function, $y=5x$, to produce an output of 45 (line 1), Tina determined they would need a constant of "minus 5" (line 4) to produce the desired value of 40. When the candidate $y=5x-5$ did not decrypt the message, she considered the alternative ordered pair, (1, −8). Computing that the candidate would map 1 to 0 rather than −8, she rejected the candidate and tried a new coefficient (line 5). Pairing this coefficient of 6 with an incorrect constant value of negative 9, she spoke aloud the series of operations on the input value 9 performed by the new candidate function $y=6x-9$ as she entered the new parameters (line 6). After pausing while the device updated, Tina acknowledged that she "didn't get it right", and revised the constant to its correct value of −14 (line 8). In the moments that followed, she edited the candidate function accordingly, and saw the message decrypt.

This attempt to determine parameters based on the specification of an ordered pair hinged on inputting a value into various candidate functions to see which one generated the expected output—in other words, it treated the candidate function as a process for mapping one element of that pair to the other. Tina led Group B to the determination of a constant value by computing the product of a candidate coefficient and an ordered pair input of 9, and then calculating the difference between that product and the output value 45. Whereas the non-ordered-pair inferences illustrated in Episodes 1 through 3 emphasized and utilized object-like properties of the encoding and candidate functions to evaluate candidate exponents and coefficients, these efforts to determine constant values and to revise candidate coefficients based on ordered pairs relied much more heavily on operational aspects of those functions. Though the approach in this episode also relies on variation of parameters, as all uses of the *Code Breaker* interface must, here that variation serves as a resource for evaluating whether the candidate achieves the desired match between a single input and output, rather than a set of such pairs. In other words, the salient characteristic of the desired candidate function in this approach is as a process for computing 40 from operations on 9.

7 Inferences from multiple ordered pairs

The excerpt below joins Group B after they had already matched the ciphertext values 8 and 88 with the plaintext letters A and I, respectively. In the time since, Tina had worked alone, with paper and pencil, while the other students in the group continued examining the code with their handhelds. The segment opens with Tina explaining her efforts to the rest of the group:

1. Tina: I'm trying to find all possible...I'm writing down all possible functions for A to equal 8. But...
2. Monique: Wow.
3. Tina: I has to equal 88 at the same time. So I'm crossing out all of these functions that I've thought of so far. [Monique, Jessica, and Shirley discuss this as Tina continues to work for about 20 more seconds]
4. Tina: This would work. This would work. Try *y* equals 10*x* minus 2. [Event B8]

As her account suggests, Tina had been writing down a list of linear candidate functions and checking to see which would include both the ordered pairs (1, 8) and (9, 88) in a methodical attempt to solve the system of equations $8=a(1)+b$ and $88=a(9)+b$. First fixing a value for a, she would generate a list of equations with that coefficient and varying values for b ($y=ax+1$, $y=ax+2$, and so on) and then systematically eliminate all those functions that failed to generate both ordered pairs. Eventually, she found a function, $y=10x-2$, that solved the system of linear equations she had established, successfully mapping 1 to 8 and 9 to 88.

Tina's strategy relied on the fact that the correct encoding function would necessarily be simultaneously satisfied by these two distinct ordered pairs, which together uniquely specified the linear function she sought. The approach relied on both operational and structural aspects of the encoding function. Much as in the previous episode, she treated the encoding function as a process for mapping a known input to a known output, in this case both A to 8 and I to 88, and methodically organized and enacted a series of candidate processes for generating each of these mappings. But she also relied on her understanding of the encoding function as an object characterized by the completion of these two mappings "at the same time" (line 3)—as a set of such ordered pairs rather than merely a procedure for generating one such pairing at a time. Correctly identifying this object involved examining an array of candidate functions and rejecting all that lacked this property. Consequently, her approach integrated both the attention to processes for input to output values typical of ordered pair strategies, and the emphasis on systematic parametric variation to achieve a desired set-mapping characteristic of non-ordered pair strategies.

8 Decoding strategies

The functions used to encode these messages generally grew more complex as each group progressed through the tasks, featuring gradual increases from first to third degree polynomial functions, and the eventual introduction of codes with offsets. The general increases in code complexity were accompanied by changes in the ways the groups went about trying to solve them, as each group began breaking codes using only inferences from general characteristics of the encoding function depicted across the *Code Breaker* representations, and gradually developed decryption approaches that relied on the specification of one or more ordered pairs. In particular, of the six codes they successfully decrypted, Group A specified no ordered pairs in the first three, one ordered pair in each of the next two, and multiple ordered pairs in the last. Likewise, Group B first solved two codes without use of an ordered pair, then three codes through the specification of an ordered pair, then two codes using multiple ordered pairs. A complete list of the decoding events undertaken by the two groups is presented in Table 1.

One notable trend spanning the decryption activities of both groups is illustrated in Table 2. Decoding approaches involving ordered pairs were utilized more effectively than those based on other information derived from the *Code Breaker* representations. Six of the 11 codes based on linear functions and decrypted by the two groups were solved without the use of a specific ordered pair. By contrast, the focus groups solved no codes based on quadratic or cubic functions, and only one featuring an offset, without the specification of at least one ordered pair. In other words, groups broke both more, and more challenging, codes when they utilized ordered pairs than when they did not. In total, the two groups solved 11 of the 13 codes in which they successfully specified one or more ordered pairs, and only six of the 15 in which they relied only on general characteristics. Indeed, Group B's more extensive use of ordered pair strategies may explain their higher overall rate of success (see Table 1) in solving decryption problems.

While groups often needed to implement ordered-pair strategies in order to fully identify an encoding function, particularly of degree two or three, they were usually able to correctly determine at least some of the parameters whether or not they fully solved the code. Table 3 illustrates the relative importance of these strategy types in the determination of various encoding function parameters. Exponents were nearly always, and coefficients usually, identified through general characteristics only, though several coefficients required the application of one or more ordered pairs. By contrast, a majority of constants were determined through specification of an ordered pair. Together, Tables 2 and 3 help to characterize a general pattern in the groups' code-breaking processes, in which they were usually able to determine the exponent and coefficient of an encoding function by analyzing its general characteristics. To identify a constant and thus fully solve the code, however, often required additional inferences involving specific ordered pairs.

In general, groups solved decryption problems with greater success, and decrypted codes based on more complex functions, as they learned to specify and draw inferences from ordered pairs. Strategies based on inferences from general characteristics solved only codes based on linear encoding functions, and determined most exponents and coefficients but few constants. By contrast, strategies featuring ordered pairs solved more—and more complex—codes, and specified most constants and some coefficients.

9 Discussion

The preceding analyses reveal two main results, aligned with the two research questions. First, different decryption strategies employed by the two groups emphasized different aspects of function. While structural or object-like properties were particularly salient in non-ordered pair strategies, inferences from a single ordered pair relied on operational

Table 1 Decryption events

Decrypt. event	Solved code	Encoding function	Determined parameter	Ord. pairs
Group A				
A2	No	$-2x^2-2$	Exponent	
A4	No	$5x+7$	Coefficient	
A5	Yes	$2x$	Coefficient	
A6	Yes	$-2x$	Exponent	
			Coefficient	
			Constant	
A7	Yes	$4x+7$	Exponent	
			Coefficient	
			Constant	
A8	Yes	$-4x+7$	Coefficient	1
			Constant	1
A9	No	$5x$, offset	Coefficient	1
			Constant	1
A10	No	$-2x+1$, offset	Coefficient	
			Constant	
A11	No	$5x+21$	Exponent	
			Coefficient	
A12	No	$5x^3+6$, offset	Exponent	
			Coefficient	
A14	No	$-ax^2+b$	Exponent	
			Coefficient	
			Constant	
A15	No	$-3x^2-1$, offset	Exponent	
			Coefficient	
A16	Yes	$12x^3-15$, offset	Exponent	
			Coefficient	
			Constant	1
A17	Yes	$19x^3+1$, offset	Exponent	
			Coefficient	2
			Constant	1
Group B				
B1	No	$4x-6$	Coefficient	
B2	Yes	$-2x$	Coefficient	
B3	Yes	$4x+7$	Coefficient	
			Constant	
B4	Yes	$-4x+7$	Coefficient	1
			Constant	1
B5	Yes	$6x-14$	Coefficient	1
			Constant	1
B6	No	$-6x+14$	Coefficient	
B7	Yes	$6x-14$	Coefficient	1
			Constant	1
B8	Yes	x^2+7	Exponent	
			Coefficient	2
			Constant	2
B9	Yes	$2x^2+7$	Exponent	2
			Coefficient	2
			Constant	2
B10	Yes	$2x$, offset	Constant	

Table 1 (continued)

Decrypt. event	Solved code	Encoding function	Determined parameter	Ord. pairs
			Offset	1
B11	Yes	$-2x+1$, offset	Exponent	
			Coefficient	
			Constant	
B12	Yes	$-x^2-1$, offset	Exponent	
			Coefficient	
			Constant	
			Offset	1
B13	No	$37x^3-51$, offset	Exponent	
B15	Yes	$-20x^3+16$, offset	Exponent	
			Coefficient	
			Constant	1

features of functions, and approaches that simultaneously incorporated multiple ordered pairs integrated elements of functions as both processes and objects. Second, groups were able to determine encoding function parameters more precisely and break more challenging codes as they developed decryption strategies that relied on the specification of ordered pairs. In other words, the groups' progress toward greater proficiency in the code-breaking tasks unfolded in concert with their moves from strategies emphasizing functions-as-objects to those that also drew on process characteristics.

At first pass, this account of increasing problem-solving sophistication in the *Code Breaker* environment may appear incompatible with the theory of reification, which holds that structural features of a mathematical concept follow the operational. The students in each of these groups developed fluency with non-ordered pair strategies, and made use of object-like properties of these functions, well before they began to successfully utilize ordered pairs. My claim, however, is not that groups' increasing efficacy with regard to the decryption tasks constitutes a developmental trajectory with regard to the function concept. Rather, the point is that different levels of expertise with regard to the task environment reflected and required different aspects of functions, and thus represented distinct opportunities to engage those different aspects of the function concept.

In simultaneously capitalizing on operational and structural characteristics of functions, the multiple-ordered-pairs approach is suggestive of a procept account (Gray & Tall, 1994), in which a symbolic expression integrates both the process it specifies and the object it generates. Thus, learning to enact successful strategies in this problem-solving environment might be understood in terms of developing a proceptual view of the candidate function—as achieving the flexibility to anticipate changes to candidate parameters in terms of either operations on a specific value or relations between plain and cipher text sets. Indeed, while the determination of exponents and coefficients tended to both follow from non-ordered

Table 2 Number and type of code solved by number of ordered pairs specified

Ordered pairs	Solved codes (linear)	Solved codes (quadratic)	Solved codes (cubic)	Solved codes (offset)
None	6	0	0	1
One	5	1	2	4
Multiple	0	2	1	1

Table 3 Number of parameters determined by number of ordered pairs specified

Ordered pairs	Exponents	Coefficients	Constants
None	14	17	8
One	0	5	8
Multiple	1	3	2

pair inferences and suggest an object perspective, the specification of a constant was usually not accomplished without an inference from an ordered pair and a corresponding shift to a process perspective. Though students were able to break some less-challenging codes in early episodes without identifying an ordered pair or drawing on a process perspective, they generally needed to utilize both perspectives and draw on strategies with and without ordered pairs in order to solve most decryption problems.

Importantly, the extent to which students engaged functions as objects in non-ordered pair strategies may not plumb the full depth of the structural conception posed in the theory of reification. The distinct manifestations of objects underlying non-ordered pair and multiple-ordered pair strategies, respectively, may reflect the differing notions of object that Tall, Thomas, Davis, Gray & Simpson (2000) argue are embedded within Sfard's characterization of structural perspectives on function. While "her description of a function as a 'set of ordered pairs' as structural...agrees with the formal Bourbaki approach," she also "considers the visual imagery of a graph of a function to be structural" (p. 234). Tall et al. assert that the first of Sfard's uses of the term "structural," in reference to a set-theoretic notion of functions, "involves a more sophisticated form of mental construction than the visual and enactive construction of meaning from mental images" (p. 235).

In the same way, many of the manifestations of function as object in non-ordered pair decryption strategies probably reflected only this weaker sense of "visual and enactive" meaning as they emerged from dynamic graphical images and tabular objects in the *Code Breaker* representations. In that sense, while these students may have engaged objects before processes in developing decryption strategies, they did not necessarily do so in ways that reflected the more sophisticated structural perspective consistent with a formal definition of function. In particular, many of the inferences regarding candidate coefficients and exponents that these students routinely drew from observations of the graphical view, such as the example presented in Episode 1, may have relied on object-like properties of candidate functions only in this visual sense. On the other hand, Tina's insistence that "A has to equal 8" and "I has to equal 88 at the same time" comes much closer to the formal sense of a structural perspective on function. Her approach relied on the recognition that multiple ordered pairs simultaneously define this polynomial function—that such a function was the union of such ordered pairs. A multiple-ordered-pairs strategy, then, may entail the more robust of the two notions of function-as-object highlighted by Tall et al.

However, some non-ordered-pair strategies, such as Jason and CJ's joint observation, while viewing the graph, that there must be one plaintext input for each ciphertext output, or Vince's explanation regarding the inverse function table that "there's only one code for an A"—only one output for each input—also correspond to a structural definition of functions, as pairing a set of input values with a unique set of output values. In other words, though the enactment of a strategy involving the simultaneous use of multiple ordered pairs may have necessarily invoked structural features of function, such features were also at least sometimes utilized even when students did not specify ordered pairs at all. More pointedly, the common emphasis on object properties of functions across the quite different decryption strategies of these last two non-ordered-pair examples may highlight the importance of the role played in each by the cryptographic context. Indeed, the relation between codes and polynomial functions in this learning environment may, for example, be compatible with that between embodied objects and symbolic procepts described by Gray and Tall (2001). In other words, the notion of function-as-code may have supported students' successful efforts to make mathematical meaning of the problem situation even in the absence of more sophisticated strategies for specifying candidate function parameters using one or more ordered pairs.

More generally, the flexibility groups demonstrated in shifting between strategies oriented toward processes and objects may reflect the degree to which each relied on features of the cryptographic situation. Because the activities asked students to conceptualize the familiar plaintext alphabet in relation to a given set of ciphertext values, function as mapping between sets was arguably more salient than in applied settings used to introduce the function concept through an emphasis on the co-variation of real-world quantities (e.g., Confrey & Smith, 1994). At the same time, features of the cryptographic context such as the frequency of a character or the length of a word in the ciphertext also lent particular meaning to ordered pair relationships, and supported decryption approaches which emphasized the candidate function as a process for mapping between two values. And finally, the simultaneous specification of multiple ordered pairs, much like the insights afforded by the global examination of sets of ordered pairs through graphical and tabular representations, emphasized the changing relationship between domain and range sets associated with varying candidate function parameters.

The central role of a single applied context in the *Code Breaker* environment may also mark a departure from other software oriented toward functions. In a recent review of technology-based learning environments for algebra, Kieran & Yerushalmy (2004) emphasize the ways that computer-based tools centered on functions and multiple representations can provide resources for modeling a wide range of real-world phenomena, and for expanding the array of problem types and problem-solving approaches typically available in algebra instruction. The environment presented in this paper follows this approach in the reverse direction: rather than emphasizing functions as tools for modeling phenomena, *Code Breaker* uses a particular phenomenon as a resource for conceptualizing functions, and as a context within which to equip students with multiple representational tools. To be sure, the analogy to codes does not begin to span the full scope or complexity of the function concept. But the apparent importance of that analogy in some students' problem-solving strategies may speak to both the particular usefulness of cryptography as a setting for introducing functions, and to the general potential of deeply embedding both multi-representational tools and complex concepts like function in applied contexts that teachers can mine across many problem-solving tasks and strategies and over the course of a full instructional unit.

Together, the mathematical features and the situational context of the task appear to have alternately called for perspectives on functions as both objects and as processes. While representations in the *Code Breaker* software afforded resources for viewing candidate and encoding functions as objects, and making considerable progress in solving decryption problems, groups became more effective and efficient in solving those problems only as they began to develop strategies for building on those object insights by identifying and drawing inferences from specific ordered pairs, and thus treating functions as processes. And, in a small number of instances such as Tina's simultaneous use of multiple ordered pairs, a few students developed highly effective strategies by integrating aspects of functions as both object and process. In that way, these groups' developing expertise in this cryptographic task environment can be characterized in terms of the broadening and deepening extent to which problem-solving strategies engaged the concept of function.

10 Conclusion

This paper elaborates the opportunities for engaging particular conceptions of mathematics topics associated with successfully solving applied problems in a novel computer-based learning environment. This approach illustrates the importance of carefully examining problem-solving processes, rather than simply problem content or task structure, in such environments in order to specify conceptual demands. Moreover, the account presented here suggests that learning to solve problems in an applied and multi-representational context such as the *Code Breaker* environment can invite students to alternately, and even simultaneously, view functions as processes and objects.

Importantly, and in keeping with the design-based research approach, the analysis presented in this paper treats structural and operational perspectives not as static conceptions held or not held by students, but as emergent resources in an active problem-solving process. Doing so provides an opportunity to consider the reciprocal relations between the design, as enactment of a problem-solving environment based on an analogy between the mathematical concept of function and the context of cryptanalysis, and theoretical perspectives that seek to resolve the duality of function conceptions as process and object (for a related example, see Abrahamson, 2009). In particular, the instructional value of a design for cryptography as analogy to function hinges on the extent to which that analogy sustains learner engagement with the full scope of the function concept—as process, as object, as integrated symbolic procept. At the same time, these groups' developing engagement with different aspects of functions in relation to an applied problem-solving context points to an alternative way to understand student learning about the function concept: not in terms of the reification of process into object, but rather in terms of increasing fluency in interpreting, selecting among and applying procedural and structural features of function toward a task objective.

Acknowledgements Support for this project was provided by Stanford University, The Wallenberg Global Learning Network, and Hewlett-Packard Corporation. Thanks to Roy Pea, Shelley Goldman and Jo Boaler for their guidance and collaboration in the design and investigation of this learning environment, and to Rebecca Ambrose, Garrett Kenehan, and several anonymous reviewers for their feedback on earlier drafts.

References

Abrahamson, D. (2009) Embodied design: constructing means for constructing meaning. *Educational Studies in Mathematics*, in press.

Boaler, J. (1993). Encouraging the transfer of 'school' mathematics to the 'real world' through the integration of process and content, context and culture. *Educational Studies in Mathematics*, *25*, 341–373. doi:10.1007/BF01273906.

Brown, A. (1992). Design experiments: theoretical and methodological challenges in creating complex interventions in classroom settings. *Journal of the Learning Sciences*, *2*(2), 141–178. doi:10.1207/s15327809jls0202_2.

Collins, A. (1992). Towards a design science of education. In E. Scanlon, & T. O'shea (Eds.), *New directions in educational technology* (pp. 15–22). Berlin: Springer.

Confrey, J., & Smith, E. (1994). Exponential functions, rates of change, and the multiplicative unit. *Educational Studies in Mathematics*, *26*, 135–164. doi:10.1007/BF01273661.

Gray, E., & Tall, D. (1994). Duality, ambiguity, and flexibility: a "proceptual" view of simple arithmetic. *Journal for Research in Mathematics Education*, *25*(2), 116–140. doi:10.2307/749505.

Gray, E., & Tall, D. (2001). Relationships between embodied objects and symbolic procepts: An explanatory theory of success and failure in mathematics. In M. van den Heuvel-Panhuizen (Ed.), *Proceedings of the 25th Conference of the International Group for the Psychology of Mathematics Education, 3* (pp. 65–72). Utrecht, the Netherlands: PME.

Kaput, J. (1992). Patterns in students' formalization of quantitative patterns. In G. Harel, & E. Dubinsky (Eds.), *The concept of function: Aspects of epistemology and pedagogy* (pp. 290–317). Washington, DC: Mathematical Association of America.

Kieran, C. (1992). The learning and teaching of school algebra. In D. Grouws (Ed.), *Handbook of Research on mathematics teaching and learning* (pp. 390–419). New York, NY: Macmillan.

Kieran, C., & Yerushalmy, M. (2004). Research on the role of technological environments in algebra teaching and learning. In K. Stacey, H. Chick, & M. Kendal (Eds.), *The future of the teaching and learning of algebra* (pp. 99–152). Boston, MA: Kluwer.

Moschkovich, J., Schoenfeld, A., & Arcavi, A. (1993). Aspects of understanding: On multiple perspectives and representations of linear relations and connections among them. In T. Romberg, E. Fennema, & T. Carpenter (Eds.), *Integrating Research on the Graphical Representation of Functions* (pp. 69–100). Hillsdale, NJ: Lawrence Erlbaum Associates.

O'Callaghan, B. (1998). Computer-intensive algebra and students' conceptual knowledge of functions. *Journal for Research in Mathematics Education*, *29*(1), 21–40. doi:10.2307/749716.

Roschelle, J., Pea, R., Hoadley, C., Gordin, D., & Means, B. (2000). Changing how and what children learn in school with computer-based technologies. *The Future of Children: Children and Computer Technology*, *10*(2), 76–101. doi:10.2307/1602690.

Schwarz, B., & Dreyfus, T. (1995). New actions upon old objects: A new ontological perspective on functions. *Educational Studies in Mathematics*, *29*, 259–291. doi:10.1007/BF01274094.

Schwartz, J., & Yerushalmy, M. (1992). Getting students to function in and with algebra. In G. Harel, & E. Dubinsky (Eds.), *The concept of function: Aspects of epistemology and pedagogy* (pp. 261–289). Washington, DC: Mathematical Association of America.

Sfard, A. (1991). On the dual nature of mathematical conceptions: Reflections on processes and objects as different sides of the same coin. *Educational Studies in Mathematics*, *22*, 1–36. doi:10.1007/BF00302715.

Sfard, A. (1992). Operational origins of mathematical objects and the quandary of reification—The case of function. In G. Harel, & E. Dubinsky (Eds.), *The concept of function: Aspects of epistemology and pedagogy* (pp. 59–84). Washington, DC: Mathematical Association of America.

Singh, S. (1999). *The code book: The science of secrecy from ancient Egypt to quantum cryptography*. New York, NY: Anchor.

Slavit, D. (1997). An alternate route to the reification of function. *Educational Studies in Mathematics*, *33*, 259–281. doi:10.1023/A:1002937032215.

Tall, D., Thomas, M., Davis, G., Gray, E., & Simpson, A. (2000). What is the object of the encapsulation of a process? *The Journal of Mathematical Behavior*, *18*(2), 223–241. doi:10.1016/S0732-3123(99)00029-2.

Tall, D., & Vinner, S. (1981). Concept image and concept definition in mathematics with particular reference to limits and continuity. *Educational Studies in Mathematics*, *12*(2), 151–169. doi:10.1007/BF00305619.

Thompson, P. (1994). Students, functions, and the undergraduate curriculum. In E. Dubinsky, A. Schoenfeld, & J. Kaput (Eds.), *Research in collegiate mathematics education, 1 (Issues in Mathematics Education, Vol. 4)* (pp. 21–44). Providence, RI: American Mathematical Society.

Thompson, P., & Sfard, A. (1994). Problems of reification: Representations and mathematical objects. In D. Kirshner (Ed.), *Proceedings of the Annual Meeting of the International Group for the Psychology of Mathematics Education—North America, Plenary Sessions, 1* (pp. 1–32). Baton Rouge, LA: Louisiana State University.

White, T. (2008). Debugging an artifact, instrumenting a bug: Dialectics of instrumentation and design in technology-rich learning environments. *International Journal of Computers for Mathematical Learning*, *13*(1), 1–26. doi:10.1007/s10758-007-9119-x.