JOURNAL OF MATHEMATICAL BEHAVIOR **10**, 55–77 (1991)

Learning About the Genetic Code Via Programming: Representing the Process of Translation

DON PLOGER
University of California, Berkeley

Representation, long recognized as a central theme in mathematics education, is also important in biology education. This study is a step toward greater collaboration between these two enterprises. The topic chosen is the genetic code, generally acknowledged as the discovery of the century in biology. This study presents an example of a student who initially had considerable difficulty in recalling the essential information related to the genetic code, particularly the process of translation. With Boxer, he created symbolic representations for information flow, then created a new representation for the process of translation. Three months later, he recalled his program in great detail, and he was able to use it as a basis for learning subtle features of the genetic code. Finally, he wrote a program to draw the standard representation for the genetic code. This study demonstrates that students can potentially learn about biology by creating their own representations for a topic, or by writing programs to draw standard representation. Possibilities for collaboration between mathematics educators and biology educators are suggested.

In order to learn modern biology, students need effective ways to represent biological knowledge. Knowledge representation has been addressed by mathematics educators. Recognizing the differences between the two areas, it is not surprising that, to date, biology educators have rarely cited the research on representation by mathematics educators. This study is intended to promote greater collaboration between these two groups. The genetic code is a topic that is centrally important to biology education. This topic is relevant for mathematics educators because it is an example of using an abstract representation to solve an otherwise complex problem.

This article describes how the genetic code can be represented using a flexible

---

I am grateful to Nina Gabelko and Lloyd Nebres of the University of California, Berkeley Academic Talent Development Program. I thank Steve Adams, Margaret Carlock, Andy diSessa, David Hammer, Michael Leonard, Luciano Meira, Henri Picciotto, Alan Schoenfeld, Bruce Sherin, Jack Smith, and Dan Stoll for suggestions and contributions to this research.

This research is supported primarily by a grant from the National Science Foundation, NSF–MDR–88–50363, to Andrea A. diSessa. I also gratefully acknowledge a contribution to support students from Apple Corporation, and a contribution of equipment from Sun Microsystems.

Correspondence and requests for reprints should be sent to Don Ploger, The Boxer Research Group, Education in Mathematics, Science, and Technology, 4533 Tolman Hall, University of California, Berkeley, CA 94720.

computer system; it traces the work of one student writing Boxer programs to simulate some of the most important features of the genetic code. In Boxer, it is possible to write programs to construct complex data objects and to control the movement of graphical objects. Students, therefore, have a wide range of potential representations: graphics, data objects, and the programs that construct those representations.

Representation in Mathematics

Representations, often in compact or abstract form, convey important information about the structure of the objects being represented. Understanding this relation is essential. Students often fail to recognize the correspondence between numbers and the numerals that stand for numbers. For instance, many third graders have written that 7,002 − 25 = 5,087 (Davis, 1984). If the children had understood what the numerals meant, they would have realized that the calculation was off by about 2,000.

There is also a fundamental difference between a representation and the thing itself. This distinction is not necessarily difficult. In real life, people have no difficulty distinguishing between a *menu* and the *food* it represents. But in school, children who would never eat a *menu* often fail to distinguish between numbers and the base 10 symbols that stand for numbers. Errors in understanding representations can occur even when the answer *seems* to be in the ballpark. One example, the "bus problem" (Carpenter, Lindquist, Matthews, & Silver, 1983), has achieved folklore status. Students were told that 1,128 soldiers were to be transported by buses that hold 36 soldiers, and asked to determine how many buses were needed. Most students were able to perform the relevant calculation (31 with a remainder of 12). However, fewer than one third of those students went on to deduce that 32 buses would be required. The plurality wrote "31 R 12 buses" were needed. In short, using a representation is not enough: Students must also operate on it in a meaningful way.

Fortunately, there are examples of students using representations meaningfully. In a classic example, the young Karl F. Gauss was given an assignment in his arithmetic class to add the first 10 integers (Wertheimer, 1945–1959). While his classmates were struggling with the computation, Gauss made a discovery. He represented the computation: 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 in a new way. He grouped them as follows: (1 + 10) + (2 + 9) + (3 + 8) + (4 + 7) + (5 + 6). Each of these five sums is 11, and 5 * 11 = 55. Another representation is:

$$\begin{array}{cccccccccc} 1 & + & 2 & + & 3 & + & 4 & + & 5 & + & 6 & + & 7 & + & 8 & + & 9 & + & 10 \\ 10 & + & 9 & + & 8 & + & 7 & + & 6 & + & 5 & + & 4 & + & 3 & + & 2 & + & 1 \\ \hline 11 & + & 11 & + & 11 & + & 11 & + & 11 & + & 11 & + & 11 & + & 11 & + & 11 & + & 11 \end{array}$$

In this representation, $2 * Sn = n * (n + 1)$. This led Gauss to discover the general principle that the sum of the first n integers, $Sn = n * (n + 1)/2$.

Another example of the effective use of representations by math students appeared in the *Journal of Mathematical Behavior*. It illustrates the importance that transforming a representation can make a problem that appears to be difficult into a problem that is actually easy to solve. The problem was to prove that 16 and 18 are the only positive integers that can represent, respectively, the perimeter of a rectangle and the area of the same rectangle (Davis, 1985). Although it is straightforward to demonstrate that 16 and 18 satisfy this constraint, most students (and teachers) found it difficult to prove that these are the only integers that do so.

One student, however, offered a strikingly simple solution (Imrey, 1988). Lee began with a simple algebraic statement which most students were able to derive: $ab = 2a + 2b$. Then, by dividing each side of the equation by (ab), he obtained a new representation:

$$1 = (2/b) + (2/a)$$

Lee recognized that if a is greater than 4, then the value of $(2/a)$ will be less than ½. If a and b are both greater than 4, the right-hand side of the equation is less than 1. Hence, at least one term must be less than or equal to 4. At this point, the problem becomes straightforward: It can be completely solved, simply by substituting four values for a. The only pairs of positive integers satisfying the conditions are (3, 6) and (4, 4). This study demonstrates that a simple transformation from the obvious can lead almost directly to the solution of the problem.

Representation in Modern Biology

The focus of this study is on reasoning in *modern* biology. Research on classical genetics, however, is relevant to the general theme of representation. Thomson and Stewart (1985) found that students can become adept at manipulating symbols without understanding the relevant genetics. In that case, there are two distinct representations: one relating to the algorithms of classical genetics, the other related to the biology of inheritance. Students do not coordinate these representations.

Much of the cognitive research relevant to modern biology has addressed issues of clinical medicine. That work has demonstrated that, unlike physicians, biochemists extensively consider normal causal mechanisms (Patel, Evans, & Groen, 1989; Ploger, 1988). When reasoning about a disease, biochemists follow a set of principles of cell function. But their reasoning differs from reasoning in mathematics because there is no convenient representation that is independent of the chemical details.

For other topics in biology, however, reasoning is quite similar to mathe-

matical reasoning. For example, proteins (an important class of molecules) can be represented as a symbolic sequence of subunits (amino acids). Frederick Sanger used this representation in his research (for which he received the Nobel Prize). Sanger recognized that, due to the technical limitations of the time, he could not *directly* determine the sequence of amino acids in a protein. He could however, make fragments of the protein, and determine the sequence of amino acids in the fragments. His next insight was that the information from *two* overlapping sets of fragments could be used to reconstruct the original sequence. Although Sanger had to solve enormously difficult technical problems, the solution also depended upon reasoning with a symbolic representation that was largely independent of the chemical details.

Previously, I presented advanced-placement high school students with a version of Sanger's problem (Ploger, 1990). When given a verbal description of the problem, these students did not initially solve it, and some even resisted any attempt at a solution. Then they were given a puzzle in Boxer that had two sets of fragments (with letters of the alphabet representing amino acids). With this representation of the problem, students were able to reconstruct the original sequence. One student extended the project by writing a program to solve the problem. His program served as the basis for learning activities of two other students.

These results indicated that even good students have difficulty solving problems in biology when the problem is not structured for them. However, these students, with relatively little instruction, were able to create and modify their own Boxer data structures for the problem. This suggests that Boxer is capable of providing many kinds of representations to help students conceptualize and solve problems.

Representing the Genetic Code

By 1952, biologists realized that protein is essential for life, and that the instructions to build protein are somehow contained in DNA. The problem—*how* does DNA specify the structure of a protein—was completely solved by 1966. The solution is the genetic code, generally considered to be the discovery of the century in biology. Although the basic data are obtained by extremely complex chemical methods, much of the reasoning is independent of that chemistry. That reasoning requires an effective set of representations.

The discovery of the genetic code depended upon knowledge of the structure of deoxyribonucleic acid (DNA). During the research that led to the elucidation of that structure, James Watson and Francis Crick found it important to maintain their focus by means of a simple, highly abstracted representation of the essential substances and processes. As early as 1952, they used this representation as a reminder that, at one level, they were dealing with information flow, not with chemical reactions. A version of their representation (reported in Watson, 1968) is shown in Figure 1: DNA → RNA → Protein (the arrows represent the flow of

DNA ---> RNA ---> protein

Figure 1. A representation for information flow used by Watson and Crick.

information). This abstract representation has become central in modern biology (Lehninger, 1982).

The flow of information from DNA to RNA (called *transcription*) is relatively simple. Because there are four letters in the alphabets of each component, the mapping from DNA to RNA is *one-to-one*. Transcription can be adequately represented by four rules (e.g., "If there is a G in the DNA sequence, put a C in the corresponding place in the RNA sequence.") All of the information from transcription can be fully recovered: If the RNA sequence is known, the DNA sequence can be determined without ambiguity.

The flow of information from RNA to protein (called *translation*) is much more complex. The four-letter RNA alphabet must be translated into the 20-letter protein alphabet. A group of three adjacent letters (in RNA), called a codon, codes for an amino acid (there are $4 \times 4 \times 4 = 64$ codons in all). The 64 possible codons must be assigned to 20 amino acids. Because each codon is meaningful, the mapping is *many-to-one*. Consequently, the information cannot be fully recovered: If the protein sequence is known, the corresponding DNA sequence *cannot* be determined without ambiguity.

The genetic code is "the little dictionary that relates the four-letter language of the nucleic acids [of DNA and RNA] to the twenty-letter language of proteins" (Crick, 1988, p. 169). Crick designed a representation for the assignment for each of these 64 codons (shown in Figure 2) that has become standard in the field (Lehninger, 1982; Stryer, 1981). It is a matrix with 16 cells: the rows represent the four possibilities for the first symbol and the columns represent the four possibilities for the second symbol. There are four entries for each of the 16 cells, corresponding to the four possibilities for the third symbol. For example, the cell for the codon UCA is determined by the first row (U) and the second column (C). The third element in that cell is UCA with the corresponding amino acid, serine (abbreviated "ser").

Figure 3 shows an example of transcription and translation represented in Boxer. In this example, a data box called **DNA** contains the letters CAGAGT. The second data box in the figure is **RNA;** the symbols in this box are GUCUCA. For each base in **DNA** there is a complementary base in **RNA.** The information in DNA is used to determine the sequence of bases in RNA by simple base pairing. In the process of transcription, a strand of RNA is built that is complementary to the DNA. The third data box in Figure 3 is **protein.** For instance, the symbols in the data box **RNA,** GUCUCA, are read as two codons. The first codon, GUC, represents the amino acid valine (abbreviated "val"); the second codon, UCA, as before, represents the amino acid serine.

To emphasize the role of reasoning in the discovery of the code, Francis Crick

Figure 2. The standard representation for the genetic code.

Figure 3. A Boxer representation for transcription and translation.

(1988, p. 93) stated: "We eventually realized that solving the code could be viewed as an abstract problem divorced from the actual biochemical details." Although this may not be a precise description of Crick's realization (the observation is 30 years after the fact), it is an accurate description of how the problem can be conceptualized *now*.

Learning the Genetic Code Via Programming

Because it is essentially symbolic, the genetic code readily lends itself to computer representation. For instance, David Touretzky (1984) used DNA replication as an example in a book on LISP instruction. Replication is straightforward. Like transcription, it depends upon four simple rules for base pairing. Touretzky did not address the more difficult process of translation.

This study uses Boxer instead of LISP. In Boxer, the distinction between data and procedure is made much more concrete than it is in languages such as LISP. In Boxer, any important biological structure (such as DNA or protein) can be represented simply as a data box. The contents of that data box are visible to the student. Furthermore, biological processes (such as transcription or translation) can be written as Boxer procedures acting on those data boxes. The construction of these data boxes can be as simple or complex as desired. When a student is first learning the topic, these procedures can be executed step-by-step, to reveal the details. Once those details are understood, the procedures can be executed in order to show the result of the process. Therefore, it should be easier in Boxer than in LISP for students to observe the construct and reconstruct representations of biological components.

This study examines the representations that a student made in learning about the genetic code. In particular, it addresses the following questions: What did the student already know about the topic? Did he find it valuable to abstract, elaborate, and extend this knowledge using Boxer as a representational medium? Did he create new representations? Did he remember what he learned from programming? Did the representation serve as the basis for new learning?

METHOD

A 16-year-old high school student participated in this study at the University of California at Berkeley. The study consisted of five sessions over a 6-month period. The first two sessions took place on the same day. During the first session, approximately 1 hour, the student was asked to provide a clear and concise summary of the genetic code, focusing on the flow of information. (All of this subject matter had been covered in his biology courses.) The interview was recorded on audiotape. The second session took place immediately after the interview and lasted approximately 2 hours. During this session, the student began a project in Boxer. The third session, approximately 2 hours, was held the following week; the student revised his project. In the fourth session, approximately 30 minutes, 3 months later, the student was interviewed using the representations. He was shown a version of the actual genetic code and asked to compare it to his simulation. There had been no review of this subject matter during the 3-month interval. The fifth session took place 6 months later and lasted 1 hour 20 minutes. The student wrote a program to create the standard representation for the genetic code. He had available to him a textbook containing the relevant figure.

Prior to this study, the student had been programming in Boxer for a total of 18 months. He had learned Boxer in a 6-week statistics course, offered by the Academic Talent Development Program at the University of California, Berkeley (a program which offers summer courses to intelligent, highly motivated students from ages 6 to 16). After that summer course, the student worked on Boxer projects once a week after school. In addition to this Boxer programming experience, the student had taken 2 years of biology.

RESULTS

The results are presented in five sections. The first section describes an interview with the student about the genetic code. The second section describes a Boxer program the student wrote on the same day. He made data boxes for the three most important components and the two most important processes of the genetic code. The third section reports the changes the student made 1 week later, including a graphic representation of the process of translation. The fourth section summarizes an interview 3 months later, assessing his understanding of the genetic code and his simulation. The final section describes a program the student wrote 6 months after the study began, to create the standard representation for translation, *from scratch,* in less than 1 hour 30 minutes.

Initial Interview Results (Day 1)

The data in this section come from a discussion with the student that covered broad issues in modern biology. The exerpts presented here will focus on one

important part of that interview: the student's attempt to describe the flow of information in transcription and translation. Although these quotes are only a small portion of that interview, they accurately reflect the student's difficulty in recalling the essential information. (A far more detailed examination will be prepared as a separate research report.)

The student was not readily able to provide a correct answer, even though he knew, in some sense, more than enough information. He had difficulty because he was not able to describe the flow of information without introducing details from other areas of biology. For instance, he remembered that there are proteins in the nucleus that stabilize DNA. At one level of detail this is very important, but in describing information flow, that fact is completely irrelevant. He noted (8 min 55 s into the interview) that: "There are proteins in the nucleus." When asked by the investigator, "Where are we now?" (9:15), the student replied:

Student: In the nucleus. [9:20]
Investigator: We're in the nucleus. Okay. [9:20]
Student: So it sees the protein and it builds a . . .
What am I doing here? [9:29]
Investigator: I don't know. [9:30]
Student: I don't know either [9:33]

Within 20 seconds, the student recovered from this error. He stated, correctly, that protein synthesis occurs at the ribosomes. Here, he used his knowledge of cell physiology to assist his recall of the genetic code. The investigator asked the student the direct question: "So what is the genetic code?" (10:28). The student began to reply, then admitted:

Student: Oh, I'm getting confused
I can't seem to work this out in my mind. [10:52]
Investigator: Well, that's interesting. [10:54]
Student: It is
I don't know.
There's some link I'm missing here. [11:04]

Over 2 minutes later, the investigator gave a specific question:

Investigator: And then how does the information in RNA get to the protein? [13:35]
Student: In the ribosomes -
A group of three bases stands for an amino acid. [13:42]

Finally, after 13 minutes, the student was able to provide a reasonable summary of the process of translation: "A group of three bases stands for an amino acid."

As soon as he said this, he remarked, "That's what I was thinking of, but I couldn't get to it." Because he had previously recalled the simpler process of transcription, the student had given a summary of the two most important processes associated with the genetic code.

The student's knowledge was insufficiently abstracted and coordinated to allow him to answer simple but high-level questions about information flow. When the interview reached the point just described, the student had little desire to continue the discussion. He was, however, quite interested in creating a Boxer representation. This programming activity began immediately after the conclusion of the interview.

Constructing an Initial Representation in Boxer (Day 1)

The student began to write a program to simulate transcription and translation by explicitly representing the most essential knowledge. First, he represented the three most important substances: DNA, RNA, and protein by three data boxes (the names of all boxes are indicated in boldface). He used ***DNA** to represent the information in DNA; ***RNA** to represent the information in RNA; and ***protein** to represent the information in protein. The student's representations were simpler than the actual genetic code. (The student used different names. The names used here are intended to be meaningful to the reader.)

Transcription. Figure 4 shows the result of the student's simulation of transcription. The process transfers the information in ***DNA** to ***RNA.** In the figure, there are a total of six symbols in ***DNA,** XXRROX. The student's version (like the real DNA) has four letters in its alphabet. He used the symbols: B, O, X, and R.

The ***transcription*** procedure has placed six symbols into the data box ***RNA.** For each symbol in the data box ***DNA,** the corresponding symbol to complete a pair is placed in ***RNA.** The assignment of these six symbols is done by a simple rule: B pairs with O; O pairs with B; X pairs with R; and R pairs with X. For instance, the first symbol in ***DNA** is X. Therefore, the symbol R is placed in ***RNA.**

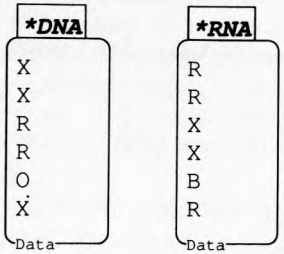

Figure 4. The student's representation for ***transcription.***

The student used the same components for ***DNA** (his DNA analog) and ***RNA** (his RNA analog). His simulation is simpler than the relationship between DNA and RNA. RNA has four components, three of which are the same as for DNA (A, G, and C). RNA does not have T, but rather U (uracil). Although this is an interesting detail, it is irrelevant to the central understanding of the genetic code.

Figure 5 shows a portion of the ***transcription*** procedure, which simulates the process of transcription. The procedure makes the assignment rule explicit. There are two data boxes at the top of the figure: ***DNA-symbol** and ***RNA-symbol.** The student has used **ifs,** a multiple "if" statement, which takes a data box as an argument. Inside the data box are four lines of code. In each line, there is a test to determine the value of ***DNA-symbol,** and then an assignment. For any symbol, only one of the four tests will be true. In this case, the third test is true (the value of ***DNA-symbol** is X), and therefore ***RNA-symbol** will be assigned the value R. This procedure will assign, for each symbol in ***DNA,** the appropriate value in ***RNA.** All that is then needed is to organize that procedure into a larger procedure that examines each item of ***DNA** and builds the appropriate values for ***RNA.**

Translation. Figure 6 shows the result of the simulation of translation. Note that the symbols in ***RNA** have been grouped in pairs. This is because the student has selected a codon (which stands for word) to consist of two symbols. (This was at the investigator's suggestion. It makes no difference whatever to the abstract problem. It does, however, make the representation much easier to observe.) The ***translation*** procedure has just been called. For each codon in ***RNA,** a word is assigned in ***protein.** For instance, the first codon in ***RNA** is (R R). The word assigned to this codon is "a" which is the first word in ***protein.** Likewise, the second codon in ***RNA** (X X) is assigned to "better" and the third codon in ***RNA** (B R) is assigned to "language."

Figure 7 shows a portion of the ***translation*** procedure. At the top row is a command, **ifs,** which tests the four possible values for the first symbol of a codon. Below this row, there are four boxes, **nouns, verbs, adjectives,** and **articles,** corresponding to the four possible outcomes. The contents of the **nouns** procedure are visible in the figure. This **ifs** command tests for the second symbol of the codon and assigns the appropriate word. Consequently, the ***translation*** procedure assigns a word to each possible codon, based upon the rules the student has devised.

Refining the Representation (1 Week Later)

Figure 8 shows the graphical output of a program written by the student to explain the process of translation. At the top of the figure, there is a graphics box which has a tree diagram. There are four major branches to the tree. At the end of each of these branches are the letters R, A, N, or V. (The investigator had

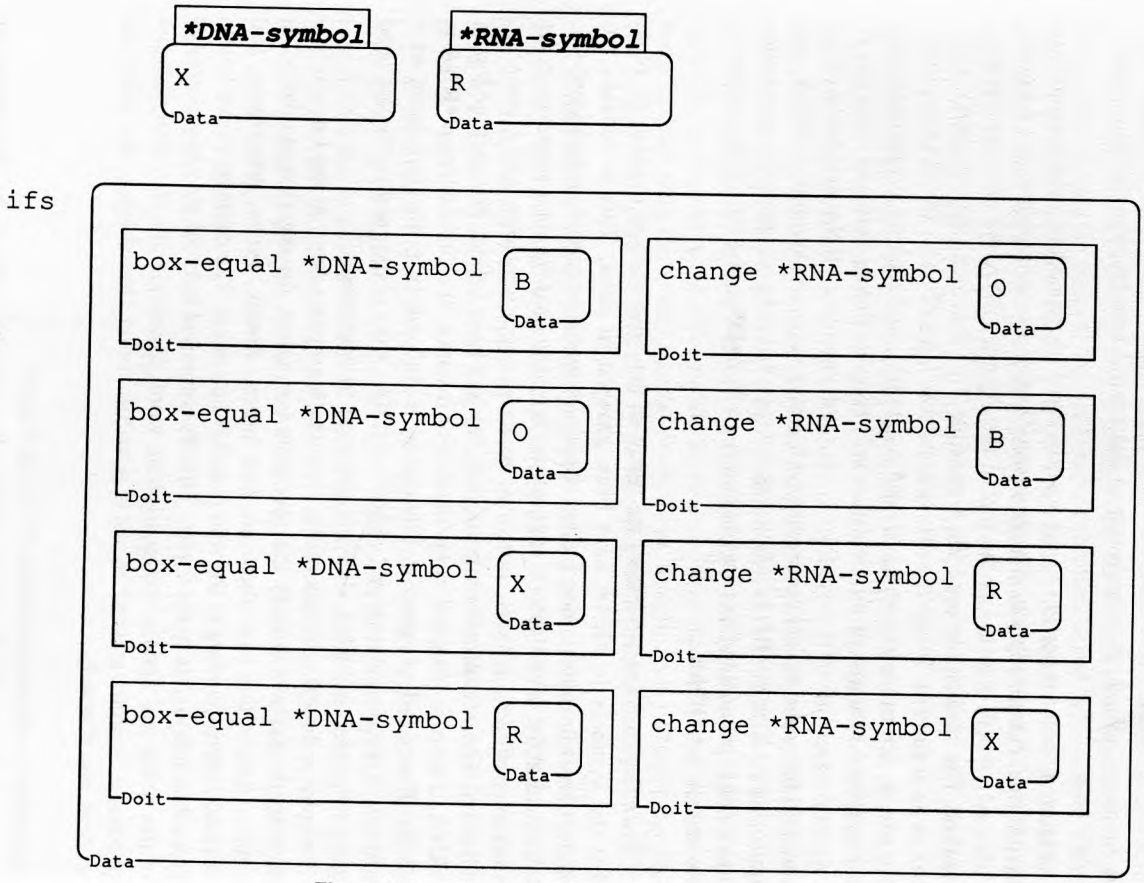

Figure 5. A portion of the ***transcription*** procedure.

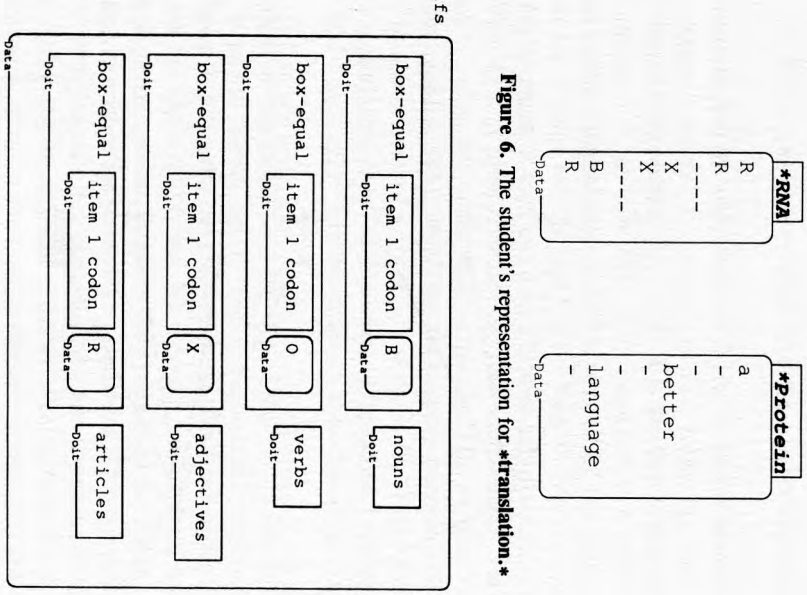

Figure 6. The student's representation for ***translation.***

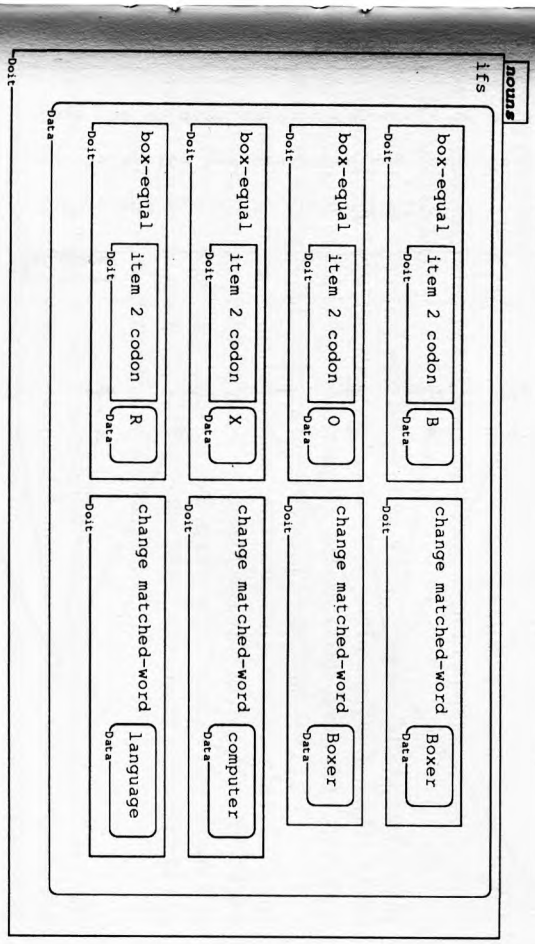

Figure 7. A portion of the ***translation*** procedure.

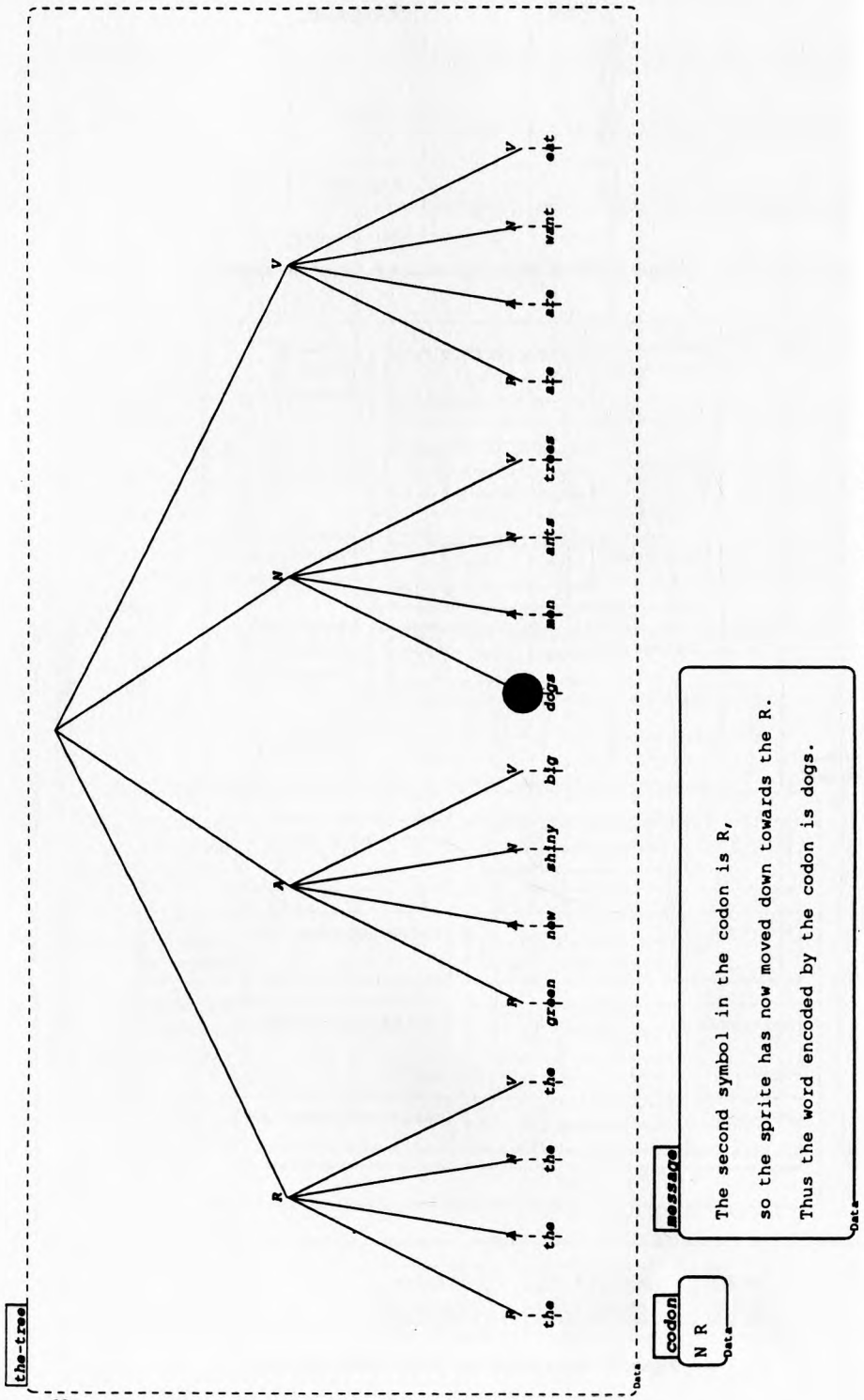

Figure 8. The student's graphical representation for translation.

difficulty remembering the assignment of B, O, X, and R to the parts of speech. At the investigator's request, the student used the following four symbols: R, A, N, and V. This is a mnemonic device. If R is the first symbol of a codon, the word is an article; if A, an adjective; if N, a noun; if V, a verb. The process of transcription in this simpler microworld is essentially the same as before. The pairing rules are: R pairs with A, and N pairs with V. This change was effortless for the student to make. It took less than 1 minute to replace the old symbols with the new.) From each of these branches are four subbranches. Therefore, each of the 16 possible codons are represented in the tree. (The student's code is simpler than the actual genetic code, which has 64 possible codons.) The processes are represented by the actions of a sprite. (A sprite is the Boxer version of a Logo turtle. It is a graphics object that moves in response to commands such as "forward" and "turn-right.") In the graphics box, the sprite is on the A in the first row.

The process of translation is a two-step process: (1) use the first codon to determine the part of speech, and (2) use the second codon to determine the exact word. In Figure 8, the sprite has reached the second level of the tree (the symbol R underneath the top-level symbol N). Beneath the graphics box are two data boxes. One data box, **codon,** contains the symbols in the codon. The other data box, **message,** describes the state of the simulation. The information in **message** is the output of a program that uses standard data construction commands. The student was easily able to adapt his data construction procedures to build a wide range of messages. The structure of the diagram is *precisely* the structure of the calling program.

An Assessment Interview, Using the Boxer Representation (3 Months Later)

The student was satisfied with his program when it had reached the state shown in Figure 8. He turned his attention to other topics in Boxer. For 3 months, he did not engage in any activity concerning the genetic code. Then, the investigator attempted to assess the student's understanding in an interview.

At the beginning of the assessment interview, the investigator asked the student to describe the genetic code. The student was able to reply directly: "Each block of three nucleotides codes for one amino acid." (Recall that it took him over 13 minutes to reach this answer in the initial assessment.) The student was then asked about the properties of his simulation.

Student: In mine, to make it more simple—each nucleotide is only . . . they only come in blocks of two. And then instead of coding for amino acids, they code for words in a sentence. So . . . that's just to lower the sample space, so there won't be as many possibilities. And the rest of it works the same—I have four, and each one is paired. So they work like that.

Investigator: Ok, good. Now, there is one part that I guess is a relatively fine feature, but

your model actually captured it. That is—how many possible codons are there and how many . . .

Student: Right. That there are more possible codons than things to be coded for—both in the real world and in the model. There is some duplication. There are two different codes that code for the same amino acid, or three even. And it works the same way in my model.

The student's explanation indicates that he understands what a codon is. He gives a precise example of a codon from his own simulation, and then describes the difference between his simulated codon and the codon in the genetic code. Then, the investigator attempted to determine whether the student understood the importance of the position of a symbol in a codon. The student was able to recall what his simulation did.

Investigator: So in your actual simulation, if they had a mutation of the first symbol in a codon, versus the second symbol, how would that affect that?

Student: If it was the first one, it would disrupt the entire . . . the meaningfulness of the sentence, probably. But if it was the second one, the sentence would be different; and might not make perfect, but it would be grammatically correct.

Investigator: Ok, and what's the analogy to the real code?

Student: I'm not really sure about that. I just assumed it basically works the same way; but I'm not real sure about that.

Then, the investigator showed the student Figure 9 and asked him to explain the process.

Student: That, in this case, the third choice really doesn't matter; because glycine will be chosen no matter what. But that's pretty rare—that might be the only one, is it? Where the third codon doesn't even matter?

This reply indicates that the student is able to use his representation to understand a feature of the genetic code that he did not know before. In the genetic code, the third symbol in a codon is often irrelevant. In many cases, it can be changed without affecting the amino acid being coded for. The student did not know this as fact, but he was able to understand the concept because in his simulation, the different positions in a codon had different affects on the word that was coded for.

A Program to Generate the Standard Representation

The student wrote a program to create the matrix, previously shown in Figure 2, which is the standard representation for translation. He had access to the relevant figure (Lehninger, 1982). The central procedure is shown in Figure 10. Notice how the context-dependent meaning of A is captured explicitly in locations in the hierarchical structure of the program. The first line of the figure shows a com-

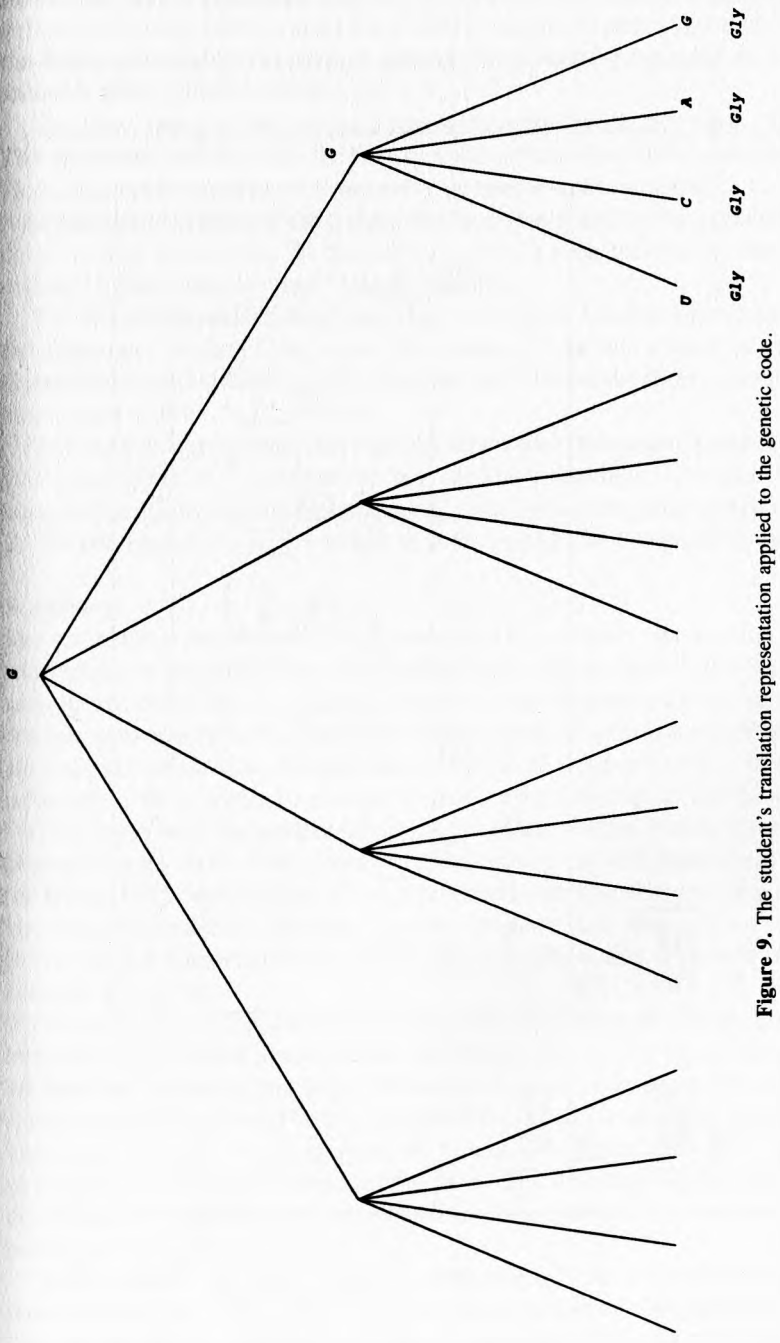

Figure 9. The student's translation representation applied to the genetic code.

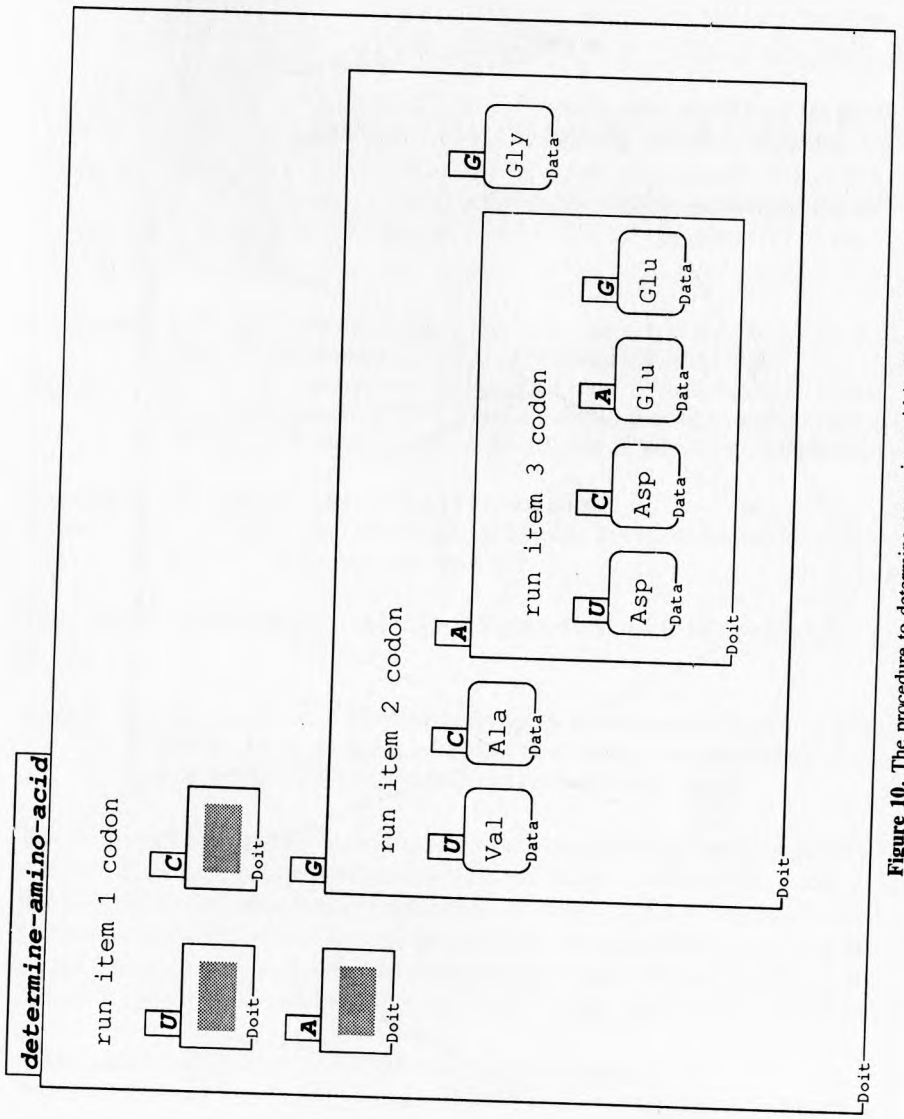

Figure 10. The procedure to determine an amino acid in the matrix.

mand that runs a procedure for the first letter of the codon. The student has defined the four possible values for a symbol as procedures. (The student wrote the Boxer command: **run [item 1 codon].** This can be interpreted as: **run-the-value-of [first-symbol-of codon].**)

The functioning of this program can be illustrated with the codon "G A C." The procedure will first run the first symbol in the codon in the outermost box. Here, there is a procedure defined as G, that is the open doit box. In that box, it runs the second symbol of the codon, A. Here, that is defined as a procedure that contains four data boxes. In that environment, it runs the third symbol of the codon, C. The value is "asp" [aspartic acid].

The **determine-amino-acid** procedure provides a concrete representation for the redundancy of the genetic code. For instance, if the first symbol in a codon is G and the second symbol is U, it does not matter what the third symbol is. The amino acid will be "val" (valine).

The student's procedure, takes apart a data object (representing a codon), uses that information to determine the appropriate procedures, and then builds a complex data object (the complete matrix, which was previously shown in Figure 2). He completed this entire program, *from scratch,* in 1 hour 20 minutes.

Summary

The student was initially unable to give a concise summary of a centrally important biological process, even when he had more domain knowledge than necessary. In particular, he was not able to recall protein synthesis until he recalled the processes associated with a particular cell structure (the ribosomes). More generally, he was not able to describe the essentials of the genetic code in terms of information flow, without referring to details of the biological mechanisms.

That same day, the student created a simulation of the genetic code in two phases of work. In the first phase, he created a program with three boxes to stand for DNA, RNA and protein. His simulation was simpler than the genetic code, but it represented the important features. He used four arbitrary symbols, selected pairing-assignment rules, and wrote procedures that simulated transcription and translation.

One week later, the student created a tree representation for the process of translation. He created a tree that had four branches at the top level, and a total of 16 branches at the second level. When a codon was translated into a word, a sprite traversed the tree. The tree modelled the calling structure of his program. This is no accident: His program could be easily modified to show its progress with sprite movement in the tree diagram. The selection of the first branch determined the part of speech; the selection of the second branch determined the precise word.

Three months later, the student was able to recall the properties of his Boxer representation in detail, and to describe the relationship between his representation and the genetic code. In particular, he remembered that the two symbols of

his codons had different meaning. He was able to use his representation as the basis for understanding an important property of the genetic code. In the real code, the third base of a codon often does not determine the amino acid. In sharp contrast to the initial attempt, the student now can describe the genetic code accurately in terms of information flow.

Finally, after this preparation, it was easy for the student to program the representation for the standard representation. In so doing, his program captured some of the patterns of the genetic code at a fine level of detail.

DISCUSSION

The student used Boxer to abstract, elaborate, and generalize the central principles by making explicit representations of the most important components and processes. Programming makes it easier for students to build and explore flexible and useful representations. This activity was supported by Boxer's structural features, especially the capability to take apart data objects easily and to build more complex data structures. It is easy to begin programming in Boxer, because variables are visible. It is easy to make simple extensions because variables can be modified by program control. It is possible to make major changes because variables can be constructed to any level of complexity the student wishes. Boxer programs provide immediate feedback, which can be used to refine the representation. Furthermore, when the representation is in a computational medium, students can easily modify and extend their work.

This study demonstrates the value of learning subject matter via programming. The act of programming forces students to choose what to represent for a particular program, encouraging them to abstract the problem from the details of the *particular situation*. Students become highly motivated to write programs that work. Consequently, they pay close attention to details of the *program representation*—not because the teacher told them—but because they want to create a good piece of work. And to complete that work, they must not only observe the representations, they must actively create them.

This representational capability is especially useful for learning this crucially important topic in biology. The discovery itself depended upon two conceptual shifts in the representation. The first shift was to abstract the problem from the biological details. The second shift was to focus on the details of information flow. The problem has been solved by biologists, and the principles are simple. Despite this simplicity, it is very easy for students—even very good students—to become mired in the wealth of details surrounding those general principles.

During the activity, the student wrote programs to represent the process of information flow. In so doing, he made explicit the rules of transcription and translation. He spontaneously created a representation for the process of translation that was useful for him. The student's representation was a tree, which differs from the one used in standard texts (Crick's matrix). There is, of course,

nothing wrong with Crick's matrix. But because it is virtually the only representation of translation used in standard texts, it is easy for students to lose sight of the fact that it is a *representation,* not the process. By creating their own representation, students can gain an important understanding of the distinction between a representation and the thing itself. Moreover, they can understand the process as something *dynamic*.

After 3 months, he was able to remember the details of his representations. This fact does not, of course, prove that Boxer representations are easy for students to remember. It does, however, suggest that students may remember best what they actually do.

Although the results are encouraging, I do not wish to overstate the claims. This student was certainly above average in intelligence and motivation. But he had incomplete knowledge of a crucially important principle, and he was able to integrate his knowledge in a long-term project, which he undertook on his own initiative. Although there is an investment in time and resources in teaching a student Boxer, there is also a large dividend in spontaneous activity leading to productive thinking. This is a dividend that can be repaid with compound interest as the student applies his Boxer representational expertise to other subject areas, such as physics or mathematics, per se.

Towards Cross-Disciplinary Research

In both mathematics and biology, a problem can appear difficult when the student cannot find a good representation for it. It becomes easy when the student finds an appropriate representation. This representation emphasizes the symbolic nature of certain areas of biology.

One issue where mathematics and biology educators could productively collaborate is the concept of a *function*. The flow of information in both transcription and in translation can be understood as a function because each DNA sequence determines one and only one RNA sequence, and RNA sequence uniquely determines a protein sequence. The transcription function has an inverse (each RNA sequence determines one and only one DNA sequence), but the translation function does not (any given protein sequence could be coded for by a number of RNA sequences). Hence, the genetic code provides a useful context for explorations that can be essentially mathematical.

On the flip side, the mathematics education community suggests pathways for the integration of descriptions of problem solving into research. George Polya (1945/1957) described an approach to problem solving that makes good sense to experienced problem solvers. The *direct* application of Polya's ideas to education has been problematical, however. The suggestions (as presented) did not seem to help students who were not familiar with them. Polya's ideas do provide guidelines, but it is necessary to provide much more support than the text of *How to solve it* and to customize that educational support to the student (Schoenfeld, 1990).

Biology has its own elder statesman. Like George Polya, Francis Crick made many interesting observations about the role of problem solving. Crick developed a very powerful representation for the genetic code. He also noted some of the educational implications: "Some people still find DNA hard to understand. . . . Really the ideas needed to grasp the structure are, if properly presented, ridiculously easy, since they do not violate common sense, the way quantum mechanics and relativity do" (Crick, 1988, p. 48).

By themselves, Crick's observations are not likely to be of direct use in biology education. But, in an important way, Crick's ideas have many potential educational benefits. The genetic code, which 40 years ago was beyond anyone's comprehension, is now within the grasp of a diligent high school student. Integrating Crick's ideas into the biology curriculum is a major undertaking. Biology education is not, in my opinion, simply a matter of finding the right way to *present* the information: It depends upon giving students the opportunity to *represent* the information in a way that is meaningful to them.

In light of the many differences between biology and mathematics, it is not surprising that, to date, mathematics educators and biology educators have had very little to say to each other. But there are some interesting parallels between the two, and this situation can change for the better. This study is advanced to encourage communication between mathematics educators and biology educators for the investigation of how students learn by constructing useful representations.

REFERENCES

Carpenter, T.P., Lindquist, M.M., Matthews, W., & Silver, E.A. (1983). Results of the third NAEP mathematics assessment: Secondary school. *Mathematics Teacher, 76,* 652–659.

Crick, Francis (1988). *What mad pursuit: A personal view of scientific discovery.* New York: Basic Books.

Davis, Robert B. (1984). *Learning mathematics: The cognitive science approach to mathematics education.* Norwood, NJ: Ablex.

Davis, Robert B. (1985). The role of representations in problem solving: Case studies. *Journal of Mathematical Behavior, 4,* 85–97.

Imrey, Lee (1988). The Isis problem revisited. *Journal of Mathematical Behavior, 6,* 381–382.

Lehninger, Albert (1982). *Principles of biochemistry.* New York: Worth.

Patel, Vimla L., Evans, David A., & Groen, Guy J. (1989). Biomedical knowledge and clinical reasoning. In D.A. Evans & V.L. Patel (Eds.), *Cognitive science in medicine: Biomedical reasoning.* Cambridge, MA: MIT Press.

Ploger, Don (1988). Reasoning and the structure of knowledge in biochemistry. *Instructional Science, 17,* 57–76.

Ploger, Don (1990, June). Problem solving with a flexible computer system: Reconstructing a discovery in biology. *American Educational Reasearch Association Special Interest Group for Artificial Intelligence and Education Newletter,* pp. 5–9.

Polya, George (1957). *How to solve it* (2nd ed.). Princeton: Princeton University Press. (Original work published 1945)

Schoenfeld, Alan (1990). Learning to think mathematically: Problem solving, metacognition, and sense-making in mathematics. In D. Gouws (Ed.), *Handbook for research on mathematics teaching and learning.* New York: MacMillan.

Stryer, Lubert (1981). *Biochemistry.* San Francisco: Freeman.

Thomson, N., & Stewart, James (1985). Secondary schools genetics instruction: Making problem solving explicit and meaningful. *Journal of Biological Education, 19,* 53–62.

Touretzky, David S. (1984). *LISP: A gentle introduction to symbolic computation.* New York: Harper & Row.

Watson, James (1968). *The double helix: A personal account of the discovery of the structure of DNA.* New York: Mentor.

Wertheimer, Max (1959). *Productive thinking* (2nd ed.). New York: Harper & Row. (Original work published 1945)