Thompson, P. W. (1985). Understanding recursion: Process ≈ Object. In S. Damarin (Ed.), *Proceedings of the Seventh Annual Meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*, pp. 357-362.

UNDERSTANDING RECURSION: PROCESS ≈ OBJECT

Patrick W. Thompson

Illinois State University

It is hypothesized that to recognize a computation as requiring recursion, students must conceptualize a reciprocal relationship between processes and their resulting objects. An example is given, along with a discussion of the role of recursion within a mathematics curriculum.

Ask any instructor of Pascal, Logo, LISP, data structures, or algorithms to name three topics that students find most difficult. Most probably his or her list will include *recursion.* In this brief paper I will propose an hypothesis for explaining students' extreme difficulty with recursion, and will justify the importance of recursion's place in a mathematics curriculum.

First, let us ensure a common vocabulary. The term *recursive process* will mean any process that employs itself as a subprocess. The term *recursive object* will mean any object which contains an instance of itself as a component. *Recursion* will mean the class of recursive processes and recursive objects.

AN HYPOTHESIS

The distinction I have drawn between recursive processes and recursive objects is essential to formulate my hypothesis concerning students' difficulties with recursion. The hypothesis is this: to be able to recognize a problem solution as one requiring a recursive process, students must formulate their solution as a recursive object. Conversely, to recognize an object as having a recursive structure, they must formulate their description of it so that it is the result of a recursive process. That is to say, students must approach a problem with the anticipation that every object is the result of a process and every process results in an object. An example will illustrate this point.[1]

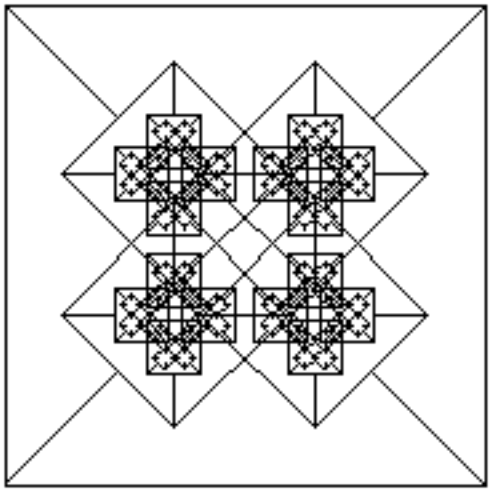
AN EXAMPLE

The figure below is one I regularly give my introductory Logo students with the intention that they write a procedure to construct a class of figures of which this is but one example.[2] Here is the kind of analysis that corresponds to this paper's hypothesis:

1. Name the class of objects: (Of those names that have been suggested, ASHTRAY is my favorite).
2. Describe an ASHTRAY: An ASHTRAY of order *n* and of size *s* is a square-with-tails with an ASHTRAY of order *n-1* and of size *s/3* at the tip of each tail.
3. State the minimal case: An ASHTRAY of order *0* is a point.

[1] The examples are written in ExperLogo, which in these examples is identical to Apple Logo.

[2] It actually requires a sequence of pictures to veridically suggest that the class has a recursive structure. To conserve space, I give a sequence of length one.



The description of the class ASHTRAY not only describes the class, it suggests a process by which to construct one. Since, in Logo, graphics is created by moving a turtle we also need to specify the relationship between the turtle and the to-be-drawn figure and to specify the relationship between the turtle's beginning and ending states when making an ASHTRAY.

4. The relationship between the turtle's initial state and a to-be-constructed ASHTRAY is that the turtle is, from its perspective, in the middle of the bottom side pointing perpendicularly toward the opposite side of the square-with-tails. (Other relationships are possible; this one merely turns out to be convenient.)
5. The effect upon the turtle of making an ASHTRAY is nil. That is, the turtle's beginning and ending states are identical. (This is merely a convenient assumption; any relationship between beginning and ending state is possible).

Notes 1 through 5 can be thought of as design specifications for making an ASHTRAY. To write the corresponding procedure, a student need only implement the description of the class as given in notes 1 through 3, keeping in mind the relationships specified by notes 4 and 5. However, to implement the description of an ASHTRAY *one must anticipate that the process one is describing produces an object of the described class, and that to obtain an object in the class one invokes the name of the process.* To make an ASHTRAY of order *n* and of size *s* , we will invoke the process named ASH.

```
TO ASH  :N  :S                          An ASHTRAY of order 0 is a point  (recall note 3).
   IF :N=0 [STOP]
   SQUARE.WITH.TAILS  :N  :S            SQUARE-WITH-TAILS will put an ASHTRAY at  the tip of
END                                     each tail  (recall note 2).
```

```
TO SQUARE.WITH.TAILS  :N  :S          [Assume turtle begins in middle of "bottom" side.]
  REPEAT 4 [ LT 90  FD :S/2           Go to end of next "tail" (recall note 4).
             RT 135  FD :S/3
             ASH  :N-1  :S/3          Make an ASHTRAY at the end of the tail  (recall note 2).
             BK :S/3  LT 45           Go to the middle of the next side.(Turtle ended where it
             FD :S/2  RT 90 ][3]        began—recall note  5).
END
```

The *decision* to write ASH :N-1 :S/3 in line 4 of SQUARE.WITH.TAILS is where it is essential to relate process and object as two sides of a coin. Many students, instead of writing ASH :N–1 :S/3, will begin to write LT 90 FD :S/2 RT 135 …. which is the beginning of another SQUARE.WITH.TAILS. That is, they become trapped in the *process* of constructing an ASHTRAY, do not recognize that what is required at that point is another *object* called an ASHTRAY, and that any ASHTRAY can be created by invoking ASH.

The necessity of relating process and object applies equally well to problems of writing recursive functions that operate upon data. However, students generally find writing recursive functions more difficult than writing recursive graphics. Apparently, when writing recursive graphics it is often sufficient to use an image of the finished product as a stimulant to cue themselves as to when to make recursive calls. When writing recursive functions, it is generally insufficient to imagine only the finished product (the function's output). Students must also reflect the data's structure in the function's structure. This is what Touretzky (1983) calls *structured recursion.* An example of reflecting a datum's structure within a function for processing it will be given in the presentation.

THE HYPOTHESIS REFORMULATED

To be able to write recursive procedures or functions (as distinct from merely reading a recursive procedure or function written by someone else), students must first describe the object to be created by the procedure in a way that reflects its recursive structure. They then can use that description as a guide for writing the procedure, keeping in mind that whenever they require an object of a particular class they invoke the name of the procedure that creates it, regardless of whether or not (at the time they invoke the name) the procedure has been completely defined. The cognitive prerequisites for this ability amount to a mindset, or belief system:

1. Any process produces an object.[4]
2. One obtains an object of a particular class by invoking the name of the process that creates it.
3. One can name (and hence invoke) a process before the process has been defined (with the intention that it will be defined eventually).

[3] In most versions of Logo, the commands in the REPEATed list must be typed as one logical line (i.e., without carriage returns).

[4] This includes "nonterminating" processes, which allows the set of natural numbers to be considered as an object. However, I would imagine that in practice most intended objects result from terminating processes.

RECURSION IN MATHEMATICAL UNDERSTANDING

The kind of object oriented thinking discussed above permeates theories of mathematical understanding. Skemp (1979) discussed a two-level model of mathematical thinking: at the lower level, one thinks by doing. At the higher level, one thinks *about* doing. Freudenthal (1972) discussed mathematical development in terms of progressively higher levels of object construction. Piaget's constructs of reflection and reflexion (Piaget & Inhelder, 1969) addressed the distinction between actions and represented actions. In each case, descriptions of intellectual advancement involve hypothesizing that what is process at one level becomes object at another level. Here, in students' study of recursion, we have the opportunity to provoke students' into making the relationship between process and object explict to themselves. One must be cautioned, however, by the empirical question as to whether or not such an awareness will actually assist students in their mathematical development.

RECURSION IN THE CURRICULUM

The school curriculum is rife with opportunities for casting mathematical content recursively. One example is given here. It defines a "grammar" for integers and integer operations (Dreyfus & Thompson, 1985). Here, the semantics of an integer is:

number	Do *number* steps in your current direction.
–number	Turn around, do *number*, turn back around.

The grammar for integers is:

1. A whole number is a number.
2. The negative of a number is a number.
3. The composition of two numbers is a number (one composes numbers by doing them consecutively).
4. The representation of a composition is equivalent to a number.
5. An operation defined as a composition of numbers is equivalent to a number.

The recursive property of the system (grammar and semantics) manifests itself when one evaluates espressions, as in **–[–70 30]**, which denotes the negative of the composition of **–70** and **30**. Our research suggests that for students to employ a rule of substitution when evaluating expressions, they must construct the distinction between process and object, as was hypothesized earlier in this paper for writing recursive procedures (Dreyfus & Thompson, 1985; Thompson & Dreyfus, 1985). Other examples can be found in topics ranging from whole number numeration to mathematical analysis.

CONCLUSION

It should be noted that the focus in this paper was explaining students' difficulties in *creating* recursive processes and objects. This is quite different from studies that focus on students' abilities to recognize already-written procedures as being recursive (cf., Kurland & Pea, no date) or their abilities to write iterative processes under the guise of recursion ("tail-end" recursion; cf., Anzai & Uesato, 1982). The ability to create recursive processes and

objects is much more difficult to cultivate than abilities to recognize "recursiveness" in already-written procedures, but at the same time once attained is much more useful.

REFERENCES

Anzai, Y., & Uesato, Y. (1982). Learning recursive procedures by middleschool children. *Proceedings of the Fourth Annual Conference of the Cognitive Science Society.* Ann Arbor, MI.

Dreyfus, T., & Thompson, P. W. (1985). Microworlds and van Hiele levels. *Proceedings of the Pshcyology of Mathematics Education.* The Netherlands: PME.

Freudenthal, H. (1972). *Mathematics as an educational task.* Dordrecht, The Netherlands: Reidel.

Kurland, D. M., & Pea, R. D. (No Date). Children's mental models of recursive Logo programs. Manuscript.

Piaget, J., & Inhelder, B. (1969). *The pschology of the child.* New York: Basic Books.

Skemp, R. R. (1979). *Intelligence, learning, and action.* London: John Wiley.

Thompson, P. W., & Dreyfus, T. (1985). Integers and algebra: Parallels in operations of thought. Manuscript (submitted).

Touretzky, D. S. (1983). *LISP — A gentle introduction to symbolic computation.* New York: McGraw-Hill.