About the Author

Seth Bergmann *is chairperson of the Computer Science Department at GSC and has been a member of the faculty since 1980. His main interests are programming languages, formal languages, compilers, and computer architecture. He has published in the area of computer algorithms.*

Seth's undergraduate degree in Physics is from Rensselaer; he holds an M.S.E. in Computer Science from the University of Pennsylvania (and is A.B.D. at Penn). He has been a consultant to Univac, U. of P., Philadelphia Child Guidance Clinic, and other institutions. He has taught Computer Science at Oberlin College and at the University of Auckland in New Zealand.

Seth lives in Glassboro with his wife, Sue, and his children, Aaron, 10, and Sarah, 8. Sue is a social work administrator with the Family Counseling Service in Camden.

Audience Participation in Computer Science Courses

Seth Bergmann

Audience Participation, How and Why

Over the years it has become clear that students learn by actively participating in problem solution or discussion rather than by listening to a lecture. I believe this to be especially true in the mathematical sciences. In a lecture environment, the student tends to produce, at best, an autonomic copy of the lecture in his or her notebook. The information seems to enter through the eyes and ears, bypass the brain entirely, and go directly to the fingers of the writing hand.

To combat this problem, I find it necessary to force the students to take an active part in the presentation. I generally refer to this as *audience participation*, to borrow a term from television game shows. I start the process by stating a problem, beginning a computer program, beginning the construction of a data structure, or somehow getting things going. Once it has become clear that we will be producing a result through a series of logical steps, I turn it over to the students, who are called upon in the order in which they sit in the class, up and down the rows of seats. Each student gets an opportunity to fill in a small piece of the puzzle solution. If a student has trouble coming up with a response,

a little help is in order, and if this doesn't work, I move on to the next student but always make sure an explanation is provided to help those who are struggling to keep up. When finished, I make it clear that the problem was solved by the *students*. I did not do it—they did—and if they can do it in class, they can do the homework, and they can solve the problems on the exam. This builds self-confidence and reduces fear of or intimidation by the difficult subject material.

The first time a class is confronted with audience participation, there is always some tension and nervousness. This tension can be lessened in the following ways. Before starting audience participation, I explain that the purpose of this exercise is not to embarrass students. They will not be graded or in any way judged on their responses. Its purpose is simply to make sure that they are thinking critically and logically in my class. Whenever a student comes up with a correct response, no matter how simple the problem, I commend the student. Also, I have found that it is important to use a casual or friendly tone of voice; speaking sternly or harshly only intimidates the students further. Occasional humor injected in the right way also helps.

When a student's response is incorrect, I usually include it in the solution being constructed on the chalk board, responding mechanically to the student. It is inevitable that a contradiction will soon become obvious, and the hands go up or students call out the correct response. In this way students continue to think even when it is not their turn to respond.

Subject Matter for Audience Participation

In computer science there is no end to the topics which lend themselves to this kind of exposition. Anything which involves the construction of the solution to a problem through a series of logical steps will usually work, as long as there has been some preliminary explanation and the problem is not excessively difficult. For example, in teaching formal

languages in my Compiler Design class, the concepts of *grammar* and *derivation tree* are essential. I give the students the following grammar for arithmetic expressions involving only addition and multiplication:

1. Expr → Expr + Term
2. Expr → Term
3. Term → Term * Factor
4. Term → Factor
5. Factor → (Expr)
6. Factor → constant

The problem is to construct a derivation tree for the expression (2 + 3) * 4 beginning with the syntactic type Expr and applying the rules of the grammar until the leaves of the tree represent the desired expression, in which 2, 3, and 4 are examples of constant. This process is shown in figure 1.

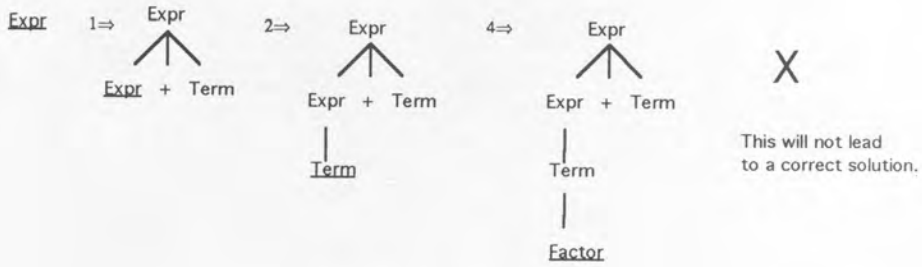

Figure 1

The students are asked which grammar rule to apply at a particular underlined point in the tree. The student's response is shown in front of the ⇒ symbol. Note that the first response, rule 1, was incorrect and was subsequently corrected to rule 2 when starting over from the beginning, as shown in Figure 2 on the next two pages.

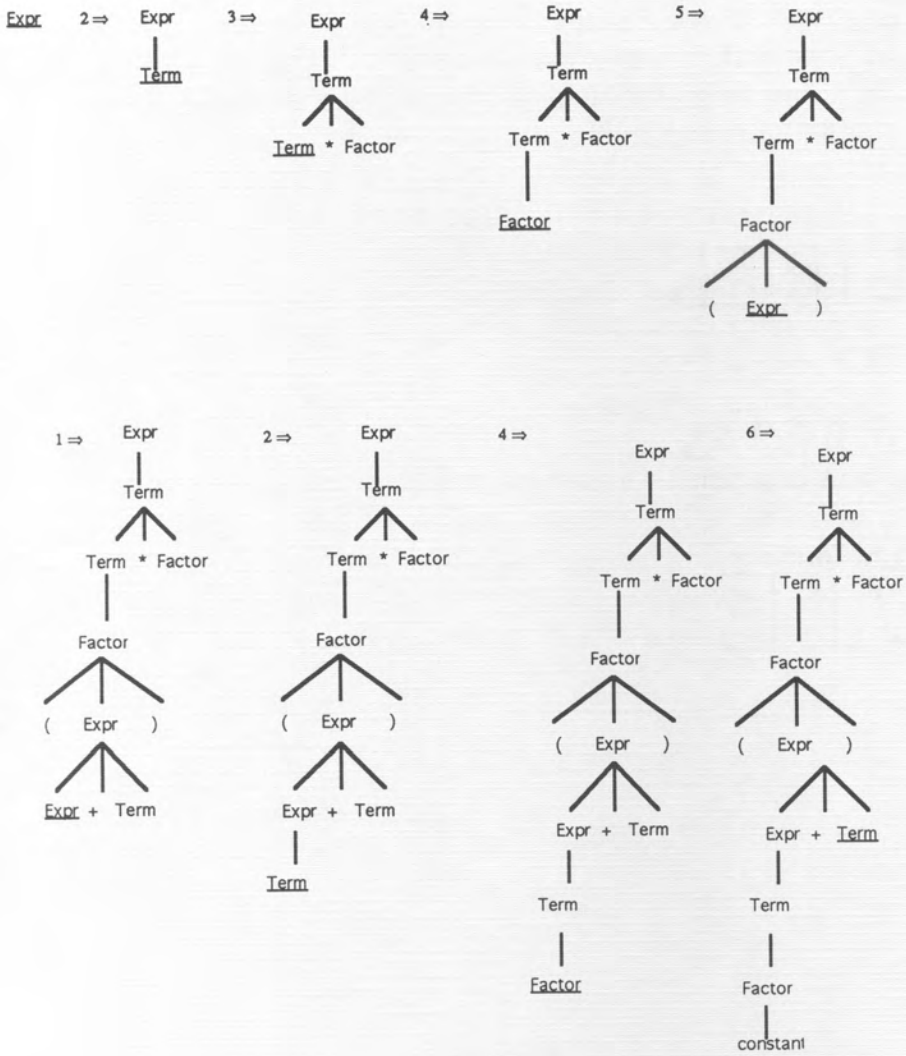
Expr
2⇒
Expr
Term
3⇒
Expr
Term
Term * Factor
4⇒
Expr
Term
Term * Factor
Factor
5⇒
Expr
Term
Term * Factor
Factor
(Expr)
1⇒
Expr
Term
Term * Factor
Factor
(Expr)
Expr + Term
2⇒
Expr
Term
Term * Factor
Factor
(Expr)
Expr + Term
Term
4⇒
Expr
Term
Term * Factor
Factor
(Expr)
Expr + Term
Term
Factor
6⇒
Expr
Term
Term * Factor
Factor
(Expr)
Expr + Term
Term
Factor
constant

Figure 2

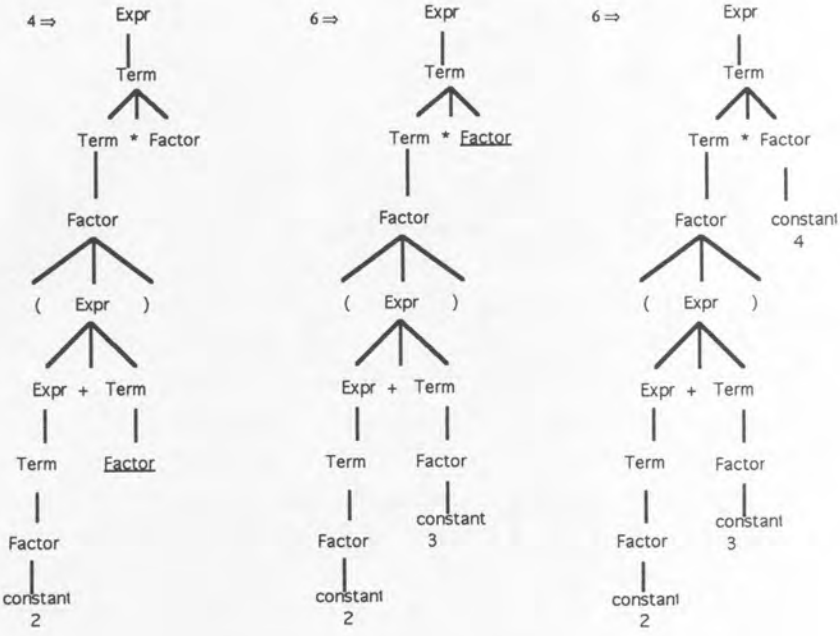

Figure 2 continued

There are many other topics in computer science which are suitable for this kind of exposition. For example, when constructing a computer program, the instructor can present a general framework, flowchart, or pseudocode for the program and have the students complete the program with audience participation. For more complex programs, it often helps to do a trace of the program's execution with audience participation. In a data structures course, after a program which builds a structure has been presented, such as a binary search tree, the construction of the tree for particular input values can be done with audience participation.

I am sure that this technique can be applied in other disciplines. For example, in mathematics the construction of a proof or the derivation of a formula can be outlined by the instructor and completed by the students. In chemistry the students could fill in a blank periodic table, and in biology

the students could label a blank anatomy chart. In general, any problem solution which involves a series of relatively small steps will be a likely candidate for audience participation.

Tools for Audience Participation

The simplest way to implement audience participation is with the chalk board in the classroom. As each student responds, the instructor fills in the response on the board (I like to move fairly quickly, and don't like to wait for students to come to the board themselves, which can also be intimidating). The chemistry and biology charts referred to above are almost as simple. One way to do this is to put the blank chart on a transparency and project it on a white board (a blackboard doesn't work quite as well) and fill in the students' responses on the board.

One of my favorite tools is a computer with monitor or monitors that can be seen by the entire class. When developing a program, the students suggest ways of filling in the missing parts of the program, and I type them into the computer. This method always catches the students' mistakes (as well as my own).

Finally, a course such as Computer Literacy, which is taught in a room with a computer for each student and an overhead display for the instructor's computer, is designed for audience participation. In this case, all the students can respond simultaneously at their own stations. The instructor, as well as students who solve the problem quickly, can help those who have trouble.