Mental Models of Recursion

Tina Götschi, Ian Sanders, Vashti Galpin
School of Computer Science
University of the Witwatersrand
Johannesburg,South Africa
tgotschi@cs.wits.ac.za, ian@cs.wits.ac.za, vashti@cs.wits.ac.za

Abstract

Recursion is a fundamental concept in Computer Science. A student's knowledge of recursion can be termed their *mental model* of recursion. A student's mental model is *viable* if it allows them to accurately and consistently represent the mechanics of recursion. Non-viable mental models are constructed if students have misconceptions about the mechanisms of recursion or have misconceptions about concepts fundamental to recursion. This paper presents a study of the mental models of recursion that first year students at the University of the Witwatersrand have constructed in 2000, 2001 and 2002. It was found that while the majority of students constructed the viable *copies* model, many non-viable models such as the *looping*, *active*, *step*, *magic*, *return-value* and various *odd* models were also constructed. Identifying the models that students have can allow lecturers to target individual students' specific problems and analysis of the models can provide insight into learning.

Categories & Subject Descriptors

K.3 *Computers & Education*: Computer & Information Science Education - Computer Science Education.

General Terms

Human Factors, Algorithms

Keywords:

Recursion, Mental Models, Constructivism, Programming, Learning, Pedagogy

SIGCSE '03, February 19-23, 2003, Reno, Nevada, USA.

1 Introduction

Recursion is a difficult concept to teach and learn [9, 15] but a fundamental concept in Computer Science. Students at the University of the Witwatersrand (Wits) are taught recursion in the first year Fundamental Algorithmic Concepts course [12] and demonstrate their understanding by implementing recursive programs in Scheme and by tracing the execution of recursive programs and algorithms in tests and exams. This research is concerned with studying those traces and identifying the mental model of recursion that Wits students in 2000, 2001 and 2002 have constructed.

The term mental model is one that is used by cognitive psychologists such as Johnson-Laird [7] and Norman [11] to define cognitive representations of knowledge (For a comprehensive survey on mental models see [13]). Mental models have been used in computer science education to describe students' knowledge of recursion [8, 4, 3]. Kahney defines recursion as "a process that is capable of triggering new instantiations of itself, with control passing forward to successive instantiations and back from terminated ones"[8, p.315]. George [5] named these two flows of control the *active* flow (when control is passed forward to new instantiations) and the *passive* flow (when control flows back from terminated ones). A viable mental model of recursion features both flows of control. Kahney called this the *copies* model [8].

While a "conceptual" model is the tool a teacher uses to teach the concept of recursion, a "mental" model is the understanding that an individual learner constructs [16]. Kahney found that novices possess mental models at variance with the conceptual model of recursion and thus are not able to accurately predict the behaviour of a recursive program. Non-viable models are not unexpected as constructivism, a philosophy of learning becoming increasingly influential in computer science [1], posits that learning is an active process where the learner is continually making sense of their experiential world. Thus, if experience changes, so does knowledge, and an educator's role is to guide students towards the shared common knowledge of recursion held by computer scientists.

In this research, mental models were identified from students' traces of the execution of recursive programs. A "trace" is a student's representation of the flow of control and the calculation of the solution of a recursive program. Traces were obtained from examinations and class tests. Each trace was coded into categories. The categories defined characteristics of recursion such as the active flow, the passive flow and the base case. As students' traces were examined, further categories were added to describe other

features of students' traces. Mental models were identified from the combination of categories a trace exhibited.

It was found that from 2000 to 2002, the majority of Wits first year students constructed Kahney's *copies* model, but students also constructed the non-viable *looping* and *magic* models. New models were identified, these are the *active*, *step*, and *return-value* models. This research also found that some non-viable models, such as the *magic* and *active* seemed to be precursors of the *copies* model and students who had these models would need only a little more guidance from lecturers in order to construct a viable model. Other models, such as the *step* and *return-value* models indicate that students had many misconceptions about the process of recursion and a non-viable model of computer behaviour.

The outcome of this research is a method that allows lecturers to identify students' mental models of recursion. Mental models can be used to identify students' misconceptions and this can provide lecturers with understanding of what specific help students need. It should also make lecturers reflect on their own teaching methods and the examples they use when explaining recursion. Already the course lecturer at Wits is taking a different approach to teaching recursion. The first example that students are presented with is a more complicated program with embedded recursion rather than a simple recursive program that could lead to a non-viable looping model. Also, when students demonstrate their understanding of recursion, the lecturer requires that they indicate an understanding the passive flow. It is hoped that the method to identify mental models can be applied elsewhere and that this gives lecturers a diagnostic tool that will aid in the development of teaching targeting specific students' non-viable mental models.

The next section, Section 2, provides a description of constructivism and the Wits pedagogy. Section 3 describes the process of data collection and analysis. A discussion of the results is presented in Section 4 and some ideas for the application of this research and further work are proposed in section 5.

2 Background

Constructivism is a philosophical perspective on knowledge and learning, hypothesising that learners actively construct their own knowledge, they don't passively receive it [14]. Furthermore, knowledge is seen to be "viable" rather than "true" as constructivism holds that the only world known is that of our experience [14]. Knowledge is "viable" if it fits with our real-world experiences [6] and allows us to make accurate predictions of phenomena. Social interaction and communication through language accounts for the fact that individuals construct knowledge that is common to a group, for example computer scientists. It is the shared common knowledge that computer scientists have of recursion that lecturers want their students to construct.

Constructivism influences pedagogy because it gives lecturers insights into what their role should be to foster successful learning. Teachers cannot "tell" and expect learners to "get" knowledge. Teachers should generate perturbations in students' existing conceptual structures and hence foster new combinations of concepts [14]. This means that lecturers should present students with problems and examples that challenge their current understanding and reveal non-viable constructions.

Shared common knowledge of recursion is embodied in the many conceptual models of recursion [16]. At Wits, process tracing models, showing the copies or successive instantiations of the recursive program, and the abstract mathematical basis of recursion, namely mathematical induction are the conceptual models used to teach recursion. Wits students are introduced to recursion with a simple dramatisation, an approach that Ben-Ari [2, p.47] recommends since: "The behaviour of elementary programs can be faithfully reflected in dramatisations". Once students have grasped the basic mechanisms of recursion (the diminishing arguments, the base case and the building of the solution) process tracing models are presented to give a more precise representation of the mechanisms of recursion. Students then trace recursive programs written in Scheme, the language of instruction at Wits [12]. Scheme is a functional language which isolates the abstract expression of algorithms from implementation details such as the stack [10]. Scheme is used so that students need not learn complex syntax or implementation details which are premature if introduced at the beginning of CS1 [2]. However, students do need to understand parameter passing and how a function's return value is evaluated, as misconceptions about these can lead to the construction of non-viable models of recursion.

3 Data Collection and Analysis

The research began in October 2001 and the first set of data was obtained from the June 2001 exams. Each student's trace of the given recursive program was coded into categories derived from the definition of recursion. New categories, describing features found in students' traces, were added as the analysis progressed. Exam data proved to be a suitable source for identifying students' understanding of recursion, thus the previous year's exams were also obtained for analysis. Coding of this second set of data (June 2000 exams) began with the categories used in stage 1. Again more categories were added to describe features of students' traces and the existing categories became more clearly defined. A student's mental model was then identified from the combination of categories that their trace. Since both exams contained only one recursive program it was not possible to draw any conclusions about whether students possess more than one mental model of recursion and apply them selectively depending on the program they are tracing. Thus both questions were incorporated into a class test, which was given to the first years in 2002. Mental models were identified directly from this third data set. The following sections discuss each stage in detail.

3.1 Data collection and analysis – stage 1

The June 2001 exam provided 172 traces of Question 1, given in the box below.

Development of coding method The initial categories were derived from conceptual models of recursion [8, 5] and as analysis progressed, more categories were added according to features found in the traces. The final categories used are defined in Table 1.

Emergence of mental models After this stage of analysis and coding, the copies and looping models could be identified. Data coded into *copy/switch/copy* categories indicated a copies model and those with *looping/stop/none*, a looping model. Some categories indicated misconceptions

Active flow	
Copy	a new invocation with a new argument shown
Loop	operation is done on the list element by element
Not shown	only answer given or the trace is not detailed
None	no recursion, one or two step evaluation (see Section 3.1)
Null	nothing can be concluded
Algebraic	algebraic manipulation of function call (see section 3.2)
Base case	
Stop	recursion stops once base case is reached
Switch	once base case reached, switch from active flow to passive flow
Check incorrect	incorrect test for base case
Base omitted	operation at base case omitted
Passive flow	
Copy	a partial solution is calculated at each level and returned to the previous invocation
None	solution evaluated at base case
Return values	each invocation's return value saved and used in calculating a solution
Return problem	misconceptions about parameter passing and return value evaluation
Operation changed	|| changed to + or × or combination, or order of operations changed

Table 1: Coding Categories

Suppose you are given the algorithm below.

```
Algorithm1(numlist)
 if nulist is empty
  then
   1
  else
   2*head(numlist) || Algorithm1(tail(numlist))
```

Note: Remember that || means concatenation (joining) of two lists.
What would the output of the algorithm be if the input list was (4 1 3 5)? Show your workings.

Question 1 (Q1)

a student could have even though their model of recursion was viable, for example, using an incorrect base case test or changing the operation. Other combinations indicated different models, but before defining these precisely, the coding method needed validation and additional data needed to be analysed.

Validation of coding The initial categories, defined by Götschi, were discussed with the other researchers and a course tutor. Sanders, the course lecturer, suggested a category for a "two step" evaluation where students evaluated a solution by combining the base case operation and the recursive call and associated operation in calculating a solution. Evidence for this (as well as a "one step" evaluation) was found and these categories were added.

Galpin found that some students did not trace the execution of the program but evaluated an answer. The data was re-examined and sorted into one of 3 categories: *trace*, *evaluation* and *answer*. A trace had a representation of the mechanics of recursion. Many traces used a graphical process tracing method or showed the flow of control using arrows. Most traces showed new instantiations of the recursive program with its associated argument value and many indicated the passive flow of control and the building of the solution. Since these traces showed the student's thought processes clearly, it was possible to use them to identify mental models.

An *evaluation* showed that a student had little or no understanding of the recursive process and therefore could not show the process of recursion. It became clear that evaluations were often indicators of non-viable models.

An *answer* had no indication of how the student calculated that answer and the student simply gave the final output of the program. These provided no information for identifying mental models and were excluded from the study.

3.2 Data collection and analysis – stage 2

The June 2000 exams provided 160 traces of Question 2, given in the box below.

You are given the algorithm

```
c(n)
  if n=1
     then 1
     else 4 c(n/2) + 3
```

What value would be returned if the algorithm was run with the initial value of $n = 8$. Show your workings.

Question 2 (Q2)

Coding began with the categories developed in stage 1. Since this recursive program defined a recurrence relation which manipulated numbers, traces had different features to the previous list manipulation question.

Traces where students attempted to derive a closed form algebraic expression and used this to calculate the solution were coded into a new category called "algebraic". Traces with an interpretation of the function call `c(n/2)` as `4*c*n/2+3 = 16c+3` or, ignoring the `c`, `16+3 = 19` were also coded as "algebraic". This category was an indicator of a step or return-value model (see Section 3.3).

It was found that, in this question, students did not change the adding and multiplying operations but changed the order of operations during the passive flow. Thus the "operation changed" category was reinterpreted. This category was an indicator of the magic model.

3.3 Mental models identified

A students' mental model was indicated by the combination of categories their trace or evaluation was coded into. Kahney's viable **copies (C)** model was identified and the following models were all thought to be non-viable models:
Looping model (L): With this model the recursive procedure is viewed as a single object rather than a series of instantiations and thus recursion is seen as a form of iteration. The solution is calculated once the base case is reached, thus the base case is seen as the stopping condition of the loop.
Active model (Ac): Although many students did show evidence of understanding the active flow of control and the instantiations of the recursive functions with smaller argument values, as well as reaching the base case correctly, they did not show the passive flow and simply calculated the solution at the base case. In the case of some recursive programs (such as Q1), the correct solution *can* be evaluated at the base case, but this is not always possible. For example, in Q2, because of the order and precedence of the operations, the base case must be evaluated and the solution passed back to previous invocations before a solution can be calculated. It is possible that students use the active model when it is viable and the copies model otherwise.
Step model (S): Students with these models have no concept of any recursive flow of control. With this model, students simply evaluate the IF-THEN-ELSE and execute either the recursive condition once (one-step), both the recursive condition and the base case (two-step).
Return value model (R): This model stems from misconceptions about when return values from a function call are evaluated. Many students hold the misconception that at each instantiation a value is evaluated, before the next instantiation is complete. These values are stored and all combined into a solution.
Magic or Syntactic model (M): This model was also defined by Kahney [8]. Students with this model recognise programs with recursive syntax and trace a program without a clear idea of how the program achieves its effect. Thus they do not have a viable model for the behaviour of a recursive program but recognise syntactic elements as indicants of recursive behaviour and use some idea of what recursion entails (instantiations with smaller arguments, a base case and finally the building of a solution) to determine a solution.
Algebraic Model (Al): Students with this model interpret the program as an algebraic problem. This model was only observed with the recurrence relation.
Odd models (O): These are also models described by Kahney and are held by students with idiosyncratic ideas about some features of the programs and thus do not correctly predict the behaviour of the programs. These students often had so many misconceptions that their traces showed aspects of looping, algebraic and return value models, or their trace was simply incomprehensible.

3.4 Data collection and analysis – stage 3

The same recurrence relation and list manipulation questions were put in a class test in 2002 written by 169 students. All 169 of the recurrence relation questions (Q1) were analysed, while 30 of the list manipulation questions were disregarded because they were simply answers (17) or left out (13). The students' traces of the recursive programs were coded directly into mental models. Table 2 gives a summary of the mental models of recursion identified from the 3 years.

Model	C	L	Ac	S	R	M	Al	O
Recurrence Relation (Q1)								
2000 n=160	50.6	8.8	2.5	17.5	5.6	10.0	3.1	1.9
2002 n=169	45.7	0.0	4.7	26.0	15.4	4.7	0.5	3.0
List Manipulation (Q2)								
2001 n=151	44.4	13.9	32.5	1.3	2.0	2.6	0.0	3.3
2002 n=139	26.0	13.7	25.0	11.5	5.0	8.0	0.0	10.8

Table 2: Mental Models of Recursion (values as %)

4 Discussion

This research presents a method for identifying mental models of recursion, and has identified new mental models of recursion. These new models clarify the range of understanding that students have of recursion. Students with odd and step models have been the least active in their learning and have no viable understanding of recursion. Those with algebraic models have applied their existing, often non-viable, knowledge of mathematics without success. Students with a return-value model could not construct a viable model of recursion because they did not have a viable model of parameter passing and return value evaluation, which are required before students can understand how a recursive program executes. Next in the range are those with magic models. These students have constructed some idea of recursion, usually the active flow and the base case, but need to be exposed to more examples that will demonstrate how recursive programs achieve their effect. Finally those with active and looping models might be using them selectively, when they are viable, but if not, need their understanding to be challenged with examples where active and looping models will not be viable.

It is pleasing to see that the majority of students over the 3 years constructed the viable copies model of recursion. The lower numbers in 2002 could be because it was earlier in the year so students had not had as much practice and because they had probably studied less for the class test than they would have for an exam. Another reason for the low 26% of copies models (Q2 in 2002), could be because looping

and active models would allow students to obtain correct solutions and thus were viable. This was because operation during the passive flow was simply the concatenation of the elements into the final list and thus the solution could be determined at the base case. So a high number of students had viable models (90.8% in 2001, 64.7% in 2002) for Q2.

Of the 45.7% students with a copies model for Q1 in 2002, 17% also had copies models for Q2 while 6% had a looping and 15% had an active model. This shows that students have more than one model and use them selectively depending on the problem at hand.

The many students with step (17.5% in 2000, 26% and 11.5% in 2002) and odd models (10.8% in 2002) is a cause for concern. These students have constructed no viable knowledge about a concept fundamental in their course of study. It is not possible to determine whether this is because they have not had enough practice or whether there are other barriers that prevent viable constructions.

The high number of magic models (10% in 2000, 8% in 2002) is not as problematic as these students do have some understanding of recursion and, with more instruction, their current understanding can be challenged and new viable models constructed. The relatively high number of return-value models in 2002 (15.4%) indicates that students do not understand how Scheme handles recursive function calls and how return values are evaluated. Lecturers should be aware that an understanding of this is a pre-requisite to a viable model of recursion and thus needs to be taught before recursion is.

5 Application and future research

Identifying students' mental models of recursion not only brings non-viable models and misconceptions to light, but can also encourage lecturers to reflect on their teaching. The Wits first year lecturer has reflected on his choice of examples used to illustrate the recursive process. He also places more emphasis on the passive flow and its role in recursion.

To further test this method it should be applied with a different first year group in another university. This could become a diagnostic tool, such as that described by Dicheva and Close [4], in which students are tested then placed into tutorial groups according to their mental models and then provided with specific help aimed at correcting their misconception.

Mental model research can be used to gain a greater understanding of what students learn about other fundamental concepts in computer science.

6 Conclusion

It is expected that some students will construct non-viable mental models of recursion. Identifying non-viable models provides insight into the misconceptions that can cause these non-viable models and gives lecturers ideas of how to teach recursion. Students with non-viable models need to be more active in their learning and need to be provided with specific instruction to facilitate the construction of the copies model. This could be done using examples and exercises aimed at challenging their current understanding which their existing mental models shows. Therefore a method to identify a students' mental model is useful as it provides lecturers with information that will allow them to address individual needs and enhance learning.

References

[1] Ben-Ari, M. Constructivism in Computer Science Education. *SIGCSE Bulletin 30*, 1 (1998), 257–261.

[2] Ben-Ari, M., and Reich, N. Recursion: From Drama to Program. *Aspects of Teaching Computer Science 7* (1996), 45–47.

[3] Bhuiyan, S., Greer, J., and McCalla, G. Supporting the learning of recursive problem solving. *Interactive Learning Environments 4*, 2 (1994), 115–139.

[4] Dicheva, D., and Close, J. Misconceptions in Recursion: Diagnostic Teaching. In *Proceedings of the Sixth EUROLOGO Conference – Learning and Exploring with Logo* (Budapest, Hungary, 1997), pp. 234–239.

[5] George, C. EROSI–visualising recursion and discovering new errors. In *Proceedings of the 31st SIGCSE Technical Symposium* (Austin, TX USA, Mar. 2000), pp. 305 – 309.

[6] Jaworski, B. *Investigating Mathematics Teaching*. The Falmer Press, London, 1994.

[7] Johnson-Laird, P. *Mental Models*. Cambridge University Press, Cambridge, 1983.

[8] Kahney, K. What do novice programmers know about recursion? In *Studying the Novice Programmer*, E. Soloway and J. Spohrer, Eds. L.Erlbaum, Hillsdale, New Jersey, 1989, pp. 315–323.

[9] Levy, D., and Lapidot, T. Recursively Speaking: Analyzing Students' Discourse of Recursive Phemomena. In *Proceedings of the 31st SIGCSE Technical Symposium* (Austin, TX, USA, Mar. 2000), pp. 315–319.

[10] Manis, V., and Little, J. *The Schematics of Computation*. Prentice Hall, 1995.

[11] Norman, D. Some observations on mental models. In *Mental Models*, D. Gentner and A. Stevens, Eds. L.Erlbaum, Hillsdale, New Jersey, 1983.

[12] Sanders, I., and Mueller, C. A Fundamentals-based First Year Computer Science Curriculum. In *Proceedings of the 31st SIGCSE Technical Symposium* (Austin, TX, USA, Mar. 2000), pp. 227–231.

[13] Schwamb, K. B. Mental models: A survey. URL:`citeseer.nj.nec.com/schwamb90mental.html`.

[14] Von Glasersfeld, E. Questions and answers about radical constructivism. In *Scope Sequence and co-ordination of secondary School Science Vol 2 Relevant Research.*, K. Pearsall, M., Ed. National Science Teachers Association, Washington DC, 1992, pp. 169–192.

[15] Wiedenbeck, S. Learning Recursion as a Concept and as a Programming Technique. *SIGCSE Bulletin 20*, 1 (Jan. 1988), 275–278.

[16] Wu, C., Dale, N., and Bethel, L. Conceptual models and cognitive learning styles in teaching recursion. In *Proceedings of the 29th SIGCSE technical symposium* (Atlanta, GA, USA, Feb. 1998), pp. 223–227.