Qualitative insights into abstraction skills from exam scripts

Julia Crossley[1] Marjahan Begum [2] Joseph Wood [2]

Department of Computer Science, City University of London

Introduction

This poster describes a study that applies a framework to a sample of exam scripts from a second year cohort. The chosen module is an Algorithm and Data Structure course that takes an implementation approach. The aim of the study is to explore the extent to which abstraction skills are revealed in this study.

The Framework

The framework classifies abstraction skills into three types and links them to the following inference types (Crossley, 2023; Kallia, 2023; Rappaport, 2023):

- Reductive abstraction $\longmapsto$ Abductive reasoning $\longmapsto$ Reducing information
- Contextual abstraction $\longmapsto$ Inductive reasoning $\longmapsto$ Extrapolating a pattern
- Constructive abstraction $\longmapsto$ Deductive reasoning $\longmapsto$ Constructing an idea

Description of study

Qualitative study investigating how the framework can be applied and refined is a first step to validating it. In order to use a range of data that is informative across the competency spectrum, a sample of naturally occurring data was selected and is being analysed (Lister,2010). Twenty two scripts were chosen out of a total of 165. In order to select a sample that would give interesting insights, a broad pattern was identified through regression analysis using the total marks on each script and the marks on the question. Scripts that fit the of the following criteria in order of priority, were selected:

- Had the strongest deviation from the regression line.
- Fit on the regression line. Differences between marks on the set question were attributed to a pattern of achievement that manifested in differences in the full exam mark.
- Had a similar overall mark to another chosen script and a significantly different mark on the chosen question.

If scatter clusters were identified, only one script was chosen from the cluster. Correlation between mark on the question and overall mark on the exam was strong enough to use a regression line as indicative of a general pattern.

Description of question: Creating an algorithm

Write a method/function in pseudo-code that receives a binary search tree and a circular list as its parameters, both containing positive integer keys. Your algorithm should return the sum of the smallest key from the circular list and the smallest key from the binary search tree. If either input structure is empty, then return the smallest key from the other. If both input parameters are empty, then return zero. Feel free to use auxiliary functions/methods if these help you. You do not need to redefine any basic operation that you call. The use of programming language specific library functions, other than the basic operations taught to you in this module, is not allowed. Explain how your method works by means of comments in the algorithm.

Why the question is interesting to study

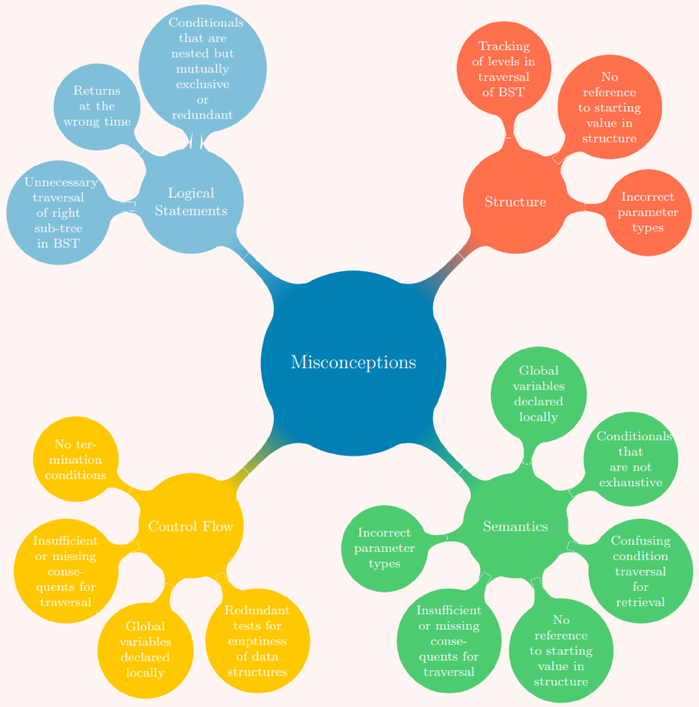
There is always variation in how a computational problem can be solved, even when controlling for purpose. When combined with module content and lateral knowledge, the variation we would see in approach also gives scope for solving questions in ways that may be less efficient, for example by storing information about states that do not need to be retrieved. This particular exam question demonstrates this well.It also gives opportunity to identify features that in this setting look stylistic but could highlight misconceptions at later educational stages.

Applying the model

The following describes behaviours that can be linked to each reasoning type. Due to the nature of the question, the scope for using abduction is limited.

Inference type	Behaviour
Abductive	Recognising appropriate seen code; placing appropriate seen code.
Inductive	Writing down an instance of a type; tracing a specific instance(s) of a type/types in the algorithm; adapting pieces of code that have been seen elsewhere using tracing.
Deductive	Adapting seen code to the purposes of the algorithm; meta-tracing.

Mindmap of misconceptions



Example of misconceptions

Here is an example of a made up response that is representative of the range of misconceptions in the sample.

```
Int MinBST ← 0
Int MinCLL ← 0
Node p ← BST      ▷ Incorrect type, unless specific implementation is assumed.
if CLL.isEmpty() then
    while p! = NULL do MinBST = p.key
    end while      ▷ No traversal of structure; due to this the condition does not change and the while never terminates.
    if CLL! = 0 AND BST == 0 then return MinCLL      ▷ Nested if statement means we have mutually exclusive conditions.
    end if
else if BST.length! = 0 then return MinBST      ▷ Incorrect condition and type.
else
    Node q ← CLL.tail
    if q.key < MinCLL then
        q ← q.next      ▷ Confusing condition for traversal and retrieval of key.
    end if
end if
MinCLL ← q      ▷ Reference to local variable outside nested statement. return MinCLL + MinBST
```

Conclusions

The methods of these studies need to remain qualitative in order to pick out enough detail. Although a single question may be the focus of the analysis, it is necessary to place it relative to other tasks, particularly if they are in the same setting. This offers more conclusive insights into the nature of students' misconceptions, as mistakes cannot always be attributed to a single misconceptions. This is particularly the case in code or algorithm creation: what is a mistake in one approach isn't necessarily in another. It is hoped that insights that come out of this study contribute towards developing a framework that can be used both diagnostically and predictively.

References

[1] Thornton C.
Quantitative abstraction theory.
Journal of Artificial Intelligence and Simulation Behaviour., 1:281–290, 2003.

[2] Crossley J.
How do students conceptualize and represent abstract ideas? an initial exploration.
In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 2*, ICER '23, page 82–86, New York, NY, USA, 2023. Association for Computing Machinery.

[3] Maria Kallia.
The search for meaning: Inferential strategic reading comprehension in programming.
In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1*, ICER '23, page 1–14, New York, NY, USA, 2023. Association for Computing Machinery.

[4] Raymond Lister, Tony Clear, Simon, Dennis J. Bouvier, Paul Carter, Anna Eckerdal, Jana Jacková, Mike Lopez, Robert McCartney, Phil Robbins, Otto Seppälä, and Errol Thompson.
Naturally occurring data as research instrument: Analyzing examination responses to study the novice programmer.
SIGCSE Bull., 41(4):156–173, jan 2010.

[5] Sara Nurollahian, Matthew Hooper, Adriana Salazar, and Eliane Wiese.
Use of an anti-pattern in cs2: Sequential if statements with exclusive conditions.
In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2023, page 542–548, New York, NY, USA, 2023. Association for Computing Machinery.

[6] Kaasenbrood E. Perrenet J.
Levels of abstraction in students' understanding of the concept of algorithm: the qualitative perspective.
SIGCSE Bulletin, 38:270–274, 2006.

[7] Turner R.
Computational intention.
Studies in logic, Grammar and Rhetoric., 63:19–30, 2020.

[8] Rapaport W.
Philosophy of Computer Science: an introduction to the issues and the literature.
Wiley-Blackwell, Hoboken, NJ, USA, 2023.