Plan Creation

Avoid some of these errors. From on-the-fly decomposition while programming could help overuse affordances of the scaffolds in ways that lead to plan-composition errors. We propose that steering students away from on-the-fly decomposition while programming could help avoid some of these errors.

Methodology

Research Questions
1. When do HTDP-trained students use templates?
2. How do focal computations manifest in HTDP programs?
3. How and when do HTDP students integrate focal computations into existing code?

Data Collection
- Spring 2015 CS1 course using HTDP in Racket
- Participants worked on the Adding Machine problem during a weekly lab session
- Video captured activity within the IDE window
- 25 (of 138) submissions analyzed

Data Coding
- Problem: Adding Machine
- Design a program called adding-machine that consumes a list of numbers and produces a list of the sums of each non-empty sublist separated by zeros. Ignore input elements that occur after the occurrence of two consecutive zeros.

Key Issues
- Students struggled to decompose the problem and compose plans – they were not taught a systematic process for doing these
- Students retrieved plans in the form of (a) operational expressions and (b) entire functions
- Key issues: on-the-fly problem decomposition around existing code and the retrieval of contexts that aren’t well suited to the problem
- Data suggests that students largely work through problem tasks and store both solutions, fitting learned solutions into the context of new problems [2,4]

Plan Retrieval

Novices rely heavily on previously learned program plans, examples, or solutions, fitting learned solutions into the context of new problems [2,4] (2) plan creation to describe novice programmer behavior when encountering a programming problem [3]

Findings
- Evidence of HTDP template use, development of foci, and task decomposition
- Students created helpers but failed to use them to effectively decompose the problem, attempting various task combinations and replicating tasks within and across functions
- Students struggled with problem decomposition and plan composition, resulting in output inconsistencies and errors

References

Acknowledgments
We thank Joe Beck for letting us collect data in his course and Mike Clancy for pointing us to the Adding Machine problem. This research is partially funded by the US NSF under grant number CCF1116539.