Graphical Loop Invariant Programming in CS1

Simon Liénardy, Lev Malcev, Benoit Donnet

simon.lienardy@uliege.be
Agenda

- Context
- Programming Methodology
- GLI
- CAFÉ
- Preliminary Evaluation
- Conclusion & Future Work
Context: First Year Students in CS

- Introduction to programming course
  - University of Liège, Belgium
  - Open access to the University (and Higher Ed. in general)
    - No background required, esp. in Mathematics
- ~ 80 Students in Computer Science (Bloc 1)
- Programming skills required by next courses
Programming Methodology

INIT
{Invariant}
while (B) {
    {Invariant \land B}
    LOOP BODY
    {Invariant}
}
{Invariant \land \neg B}
END

Deriving the code:

• Based on Dijkstra, *A Discipline of Programming* (1976)
• Graphical representation
• Represent what has already been computed
⇒ Strategy to solve the problem
    (*Metacognition*)
Graphical Loop Invariant: example
Binary search in a sorted Array

Value searched : X

A : 0 | l | u | N

· < X  To do  · > X

Sorted and unmodified
Graphical Loop Invariant: Guidelines

1. Drawing must be relevant and **named**
2. The **boundaries** of the problem are provided
3. There must be one (or more) **dividing line(s)**
4. Each dividing line should be labeled (w/ **variables**)
5. Label(s) about **what has been achieved so far**
6. Label(s) about **what should be done**
7. All the named elements and variables are present in the code
Graphical Loop Invariant: Rules

- Rules can be used to assess a Student’s Invariant
- Mistakes can be sorted into 3 categories:
  - Syntax
    E.g. missing elements
  - Semantic
    E.g. labels that do not make sense or not relevant w/ the problem
  - Matching with the code
    The Invariant should be used to write the code
Deriving the code from the Invariant
Introducing GLI

GLI
Graphical Loop Invariant

Choisissez un outil  Ajouter  Définir une zone  Valider  Delete
GLI: Patterns available

Line: 0 1 2 3 ... i-1 i ... n-1 n

Text: $a_0 + a_1x + a_2x^2 + \ldots + a_i x^i + \ldots + a_n x^n$

Array: ...0 ...i ...N
GLI: Syntax checks
GLI: Deriving the code

As soon as the checks are passed:

- One can move the dividing bars to get the init state:

  ![Diagram](image)

  ```
  int l = 0;
  int u = N-1;
  ```

- Or move them to find a condition under which the loop must be stopped

  → Infer the loop condition
Introducing CAFÉ

- French acronym for Correction Automatique et Feedback des Étudiants
- Students submit on a web platform 5 Challenges of increasing difficulty during the semester
  - Plus Challenge 0 to learn how to submit
- Assessment for Learning oriented [Sambel et al., 2013, Wiliam, 2011]

Challenges?

→ Cumulative difficulty

Nov 22 - GDD 2019

Loop Invariant Programming in CS1
Introducing CAFÉ

Day 1

Instructions
Template

Blackboard

Challenge

5/20 + Feedback and Feedforward

Nov 22 - GDD 2019
Introducing CAFÉ

Day 1
Challenge
Day 2

Blackboard

???

Challenge

CAFÉ

Int = 0;
jnt = N;
while (i)
{
}

Nov 22 - GDD 2019

+ Feedback and Feedforward

10/20
Introducing CAFÉ

Blackboard

→ Closing the feedback loop [Boud, 2000]

Challenge

CAFÉ

+ Feedback and Feedforward
CAFÉ: Submitting Invariant

- Graphical Loop Invariant is the **corner stone** of our methodology
- How to make students work with the Invariant during the Challenges?
- Blank Invariant
  - To be filled by the student
  - Bootstrap effect
### CAFÉ: Submitting Invariant
(Challenge : compute $C = A \cap B$)

|----|------------|----|------------|----|

And the values [16] to the [17] and to the [18] are in the [19].

1. -> 15. : Replace by variables, constants, numerical values
16. : Replace by “different from”, “common to “, etc.
CAFÉ: Invariant checking

- Variables present in the Inv are in the code too
  - And initialized according to the Invariant
- Array Indices in the Inv are used to index arrays
  - Out of Bound check
- Loop Variant correction
- Iterations count (if complexity constrains)
- Unit tests
- Feedback & Feedforward added after correction
CAFÉ: Preliminary Evaluation

Data about:
- Students participation
- Performance
- Perception

Over multiple years

Work in progress
Conclusion

- **GLI:**
  - Evaluation = work in progress

- **CAFÉ:**
  - Takes time but efficient and scalable
  - Language independent (modulo slight mods)
  - Lot of data to be analyzed

- Questions ?