HW 4 – Abstract Interpretation

CS 477 – Spring 2025

Assigned March 28, 2025, 11:59 PM
Due April 7, 2025, 11:59 PM

Objectives and Background

The purpose of this HW is to test your understanding of

- Galois Connection, Interval Analysis, Widening
- Reaching Definitions

This homework will contribute to 18% of the grade (from 100 pts scaled).

Policy

For all theory questions, including proofs, only typewritten solutions in LaTeX are allowed. Handwritten solutions will not be considered and will be awarded zero marks.
This homework is an individual assignment. In addition, directly copying from LLMs, including ChatGPT and Gemini, is strictly forbidden. In this homework, you can use these tools to gain insights; however, stringent measures will be implemented to detect any instances of direct replication from these sources. Any identified instances of suspicion will prompt an invitation to an oral examination.
Late policy: You have a total of four additional days throughout the term to accommodate late submissions for homework assignments. This cumulative allowance can be used at your discretion across different assignments. If you submit more than 5 minutes late for an assignment, then it is considered as using one full day of the late submission allowance. Once you exhaust the four-day late submission allowance, any further late homework submissions will result in a penalty of 33% grade reduction/day.
Please do not submit any code or use code to accomplish the non-programming problems unless we specifically instruct you to do so.

Turn-In Procedure (PLEASE READ CAREFULLY)

Use `https://www.gradescope.com/` to submit your homework. There are two assignments on Gradescope for this homework; one is designated for theory problems, and the other is for programming assignments. For the theory section of this assignment, you can submit a single PDF file or a set of photos just like HW 1.
For the coding problems, submit **exactly one file** to Gradescope, named <*net_id*>_interval.py, with <*net_id*> replaced by your own NETID. Failure to name your code files correctly will result in a 0 grade for the problem. Some tutorials on how to use Gradescope are available here `https://www.gradescope.com/get_started#student-submission`.

Galois Connection (20 pts)

Consider the monotonic abstraction function $\alpha : C \to A$ and concretization function $\gamma : A \to C$ forming a Galois connection between the complete lattices C and A. Let $\bigsqcup, \bigsqcap$ respectively be the least upper bound and the greatest lower bound operators in A. $\bigcup, \bigcap$ are similarly defined for C. $\subseteq$, and $\sqsubseteq$ are the ordering relations between the elements in C and A respectively.

1. Prove that for every $L_C \subseteq C$, $\alpha(\bigcup L_C) = \bigsqcup_{c \in L_C} \alpha(c)$ holds. **[5 pts]**

2. Prove that for every $L_A \subseteq A$, $\gamma(\bigsqcap L_A) = \bigcap_{a \in L_A} \gamma(a)$ holds. **[5 pts]**

3. Prove that $\gamma(a) = \bigcup\{c \in C \mid \alpha(c) \sqsubseteq a\}$ holds for all $a \in A$. **[5 pts]**

4. Let $F_0, F_1 : C \to C$ be two monotone functions, and let ${F_0}^{\#}, {F_1}^{\#} : A \to A$ be two functions that over-approximate them, that is such that $F_0 \circ \gamma \subseteq \gamma \circ {F_o}^{\#}$ and $F_1 \circ \gamma \subseteq \gamma \circ {F_1}^{\#}$. Then prove that $F_0 \circ F_1$, can be over-approximated by ${F_0}^{\#} \circ {F_1}^{\#}$. **[5 pts]**

Interval Transformers (20 pts)

5. Let $f : \mathbb{R} \cup \{-\infty, +\infty\} \to \mathbb{R} \cup \{-\infty, +\infty\}$ be a monotonically increasing function with $f(-\infty) = -\infty$ and $f(\infty) = \infty$. Write sound and most precise interval abstract transformer $f^\sharp : A_I \to A_I$ for f where A_I denotes the set of all intervals over extended real numbers ($\mathbb{R} \cup \{-\infty, +\infty\}$). Also formally prove the soundness of $f^\sharp$ and show it is indeed the most precise interval transformer.
   Hint: To prove soundness, first define the setwise concrete function $f_C : 2^{\mathbb{R}} \to 2^{\mathbb{R}}$ as $f_C(\mathscr{X}) = \{f(x) \mid x \in \mathscr{X}\}$, and then show that $\forall a \in A_I, f_C(\gamma(a)) \subseteq \gamma(f^\sharp(a))$. **[10 pts]**

6. Let $f : \mathbb{R} \cup \{-\infty, +\infty\} \to \mathbb{R} \cup \{-\infty, +\infty\}$ be defined as $f(x) = a \times x^2 + b \times x + c$, where $a, b, c \in \mathbb{R}$ are constants with $a \neq 0$. Write a sound and most precise interval abstract transformer $f^\sharp : A_I \to A_I$ for f, where A_I denotes the set of all intervals over the extended real numbers ($\mathbb{R} \cup \{-\infty, +\infty\}$). Additionally, formally prove the soundness of $f^\sharp$ and show that it is indeed the most precise interval transformer.
   Hint: To prove soundness, first define the setwise concrete function $f_C : 2^{\mathbb{R}} \to 2^{\mathbb{R}}$ as $f_C(\mathscr{X}) = \{f(x) \mid x \in \mathscr{X}\}$, and then show that $\forall a \in A_I, f_C(\gamma(a)) \subseteq \gamma(f^\sharp(a))$. **[10 pts]**

Reaching Definitions (20 pts)

7. Consider the following program defined over the variables `x, y, z`:

```
1 var x,y,z;
2 x = input;
3 while (x>1) {
4     z = x/2;
5     if (z>2) {
6         x = x+z;
```

```
    }
    y = x+2;
    if (y>3) {
        x = x/4;
    }
    z = z-4;
}
output z;
```

For each point in the program, determine a precise but sound approximation of the set of reaching definitions at that point. Show each iteration of the analysis at each point (Do not forget the entry and exit points).

Interval Analysis Extended - While (40 pts)

Please begin by downloading the necessary starter codes at Here. We also provided black-box solutions for the functions that you needed to implement from homework 3. The functions are presented in an encrypted way located at the bottom of the `interval.py`; you should call them as if they exist normally without any issue. **Using these functions is not required**; you can still use your version of these functions from homework 3 to do this homework. If you do want to use your own functions, please remember to comment out the provided solution at the bottom of the file.

In this assignment, you are tasked with constructing an interval analysis engine. The files `expression.py` and `command.py` contain the definitions of an expression and commands, respectively. These are consistent with the definitions provided in Homework 3, with the addition of a new command: `while`. Your objective is to perform interval analysis for this new command, utilizing a hyperparameter `max_iterations` that is a strictly positive integer to determine the point at which widening should be applied during the looping process. Further details are provided below.

The obfuscated code located at the bottom of the file `netid_interval.py` are black-box functions from last homework you can use to for this assignment. You can call these functions in your implementation, though you are welcome to use your implementation (please comment out the black-box code if you use your own). You may introduce any necessary helper functions.

Your task is to review the starter file `netid_interval.py` and complete the functions designated as TODO. Specifically, you need to implement the `executeWhile` function. (Note that you may have to implement helper functions for *widening* or splitting the store on the while condition. This problem is divided into two parts, each with its own point allocation. **(40 pts total)**

8. For the first part, assume `max_iteration` $= 1$ (the default): When the max iteration for widening is 1, you should start performing the widening operation immediately after the first iteration. **(30 pts)**

9. For the second part, the assumption is that `max_iteration` ≥ 1: When the max iteration for widening is greater than 1, you should start performing widening only after the `max_iteration`-th iteration. You should try a combination of values of `max_iteration` by setting them using the provided setter function or by modifying the global variable provided in the starter code. **(10 pts)**

A sample test case is provided in `test.py`. This test should output the correct results to the standard output—as specified in the function header comments—if your implementation is accurate. To execute the test case, use the command `python test.py`. Note that this problem defines all variables to be real variables, not integer variables. You should add more test cases, including different `while` programs and different `max_iteration` parameters, to ensure your program is correct.
Hint: Review the while iteration example from the lecture slides carefully. Pay particular attention to how and when the merge and widening operations are conducted.