Necessary Liberal Preconditions
A Master Thesis Proposal

Anran Wang

1 Introduction

With computer programs melded into almost every aspect of human life, we often not only require their efficiency, but also their correctness. We would like to know for sure that our programs deliver the desired results, and do not run forever. This is called *total correctness*. To know "for sure", we could verify programs using formal methods that have been developing for decades. One of such methods is *Hoare Triple*[3] proposed by Tony Hoare in 1969. A Hoare Triple contains three parts: a precondition, a program, a postcondition. They are written as such: $\{F\}C\{G\}$[1]. It states that if the precondition is met before the execution of the program, then the postcondition is guarantee to be satisfied after the execution. Using Hoare Triple, we can prove both the termination and the validity of the postcondition of a program in a top-to-down manner.

Another way to construct such proofs is to use *weakest precondition*(wp)[1] transformer by Dijkstra. Starting with a postcondition, it works backwards. Both wp and Hoare Triple guarantee total correctness.

Sometimes, however, we deem nontermination a good behaviour, and only want to prove that a program delivers correct output *if* it terminates. This is called *partial correctness*. The *weakest liberal precondition*[2] transformer can be used to prove partial correctness.

In this thesis, we would like to research this matter further. We aim to study wlp in a quantitive setting. Specifically, we provide a proof system for the *necessary liberal preconditions* and its correspondance with a (new) Hoare Triple.

2 Preliminaries

To simplify matters, we first look at logical predicates. The construction is similar with functions in a quantitive setting, but it is easier to understand.

2.1 Hoare Triple

A Hoare Triple is a triplet in such form: $\{G\}C\{F\}$ where G and F are logical predicates and C is a program. G is called a *precondition*, F a *postcondition*. This triple reads "if G is satisfied before the execution of C, then F is satisfied after the execution of C". Note that not only the correctness of the execution is guaranteed (via the postcondition), termination is as well, hence such triples prove total correctness.

2.2 Weakest Precondition

As the name suggests, a weakest precondition transformer translates a postcondition to a precondition. Written as a function, $wp(C,F)$ gives the "weakest" precondition that when satisfied, guarantees the

[1] Originally it was written as $F\{C\}G$, but now it is often written with brackets around conditions instead of the program.

implication	defines
$G \implies wp(C,F)$	total correctness
$G \implies wlp(C,F)$	partial correctness
$wp(C,F) \implies G$	partial incorrectness
$wlp(C,F) \implies G$	???

Table 1: implications using wp and wlp[4]

satisfaction of postcondition F after the execution of program C . A predicate G_1 is "weaker" than G_2, if G_2 implies G_1. Just like Hoare Triple, wp is concerned with total correctness. Since $wp(C,F)$ is the "weakest", wp gives rise to a valid Hoare Triple: $\{wp(C,F)\}C\{F\}$. Alternatively, wp is connected with Hoare Triple by an implication:

$$\forall G.\ G \implies wp(C,F) : \{G\}C\{F\}$$

G is then in this case the *sufficient precondition*.

2.3 Weakest Liberal Precondition

The *weakest liberal precondition*, on the other hand, is concerned with partial correctness. $wlp(C,F)$ returns then the "weakest" precondition with which all the executions of C that either

- guarantees the postcondition F ,
- or are non-terminating.

As a result, if all calculations of C terminate, $wp(C,F) \equiv wlp(C,F)$.

We can now use some implications to define correctness or incorrectness as shown in Table 1.

We are interested in the last implication labeled with ???, where G is called the *necessary liberal precondition*, in the sense that

- all preconditions that leads to non-termination of C imples G,
- a preconditions that does not imply G leads to a postcondition that does not imply F.

Note that $A \not\models F$ is not the same as $A \models \neg F$.

It is our goal to study this implication and its connection to correctness/incorrectness.

2.4 Quantitive Setting

In a quantitive setting, instead of logical predicates, we investigate functions that maps to $\mathbb{R}^{\pm\infty}$. Analogous to implications, conjunction, disjunction for predicates, we have ordering $\preceq$, join $\curlywedge$ and meet $\curlyvee$.

2.5 Guarded Command Language

We use Dijkstra's (nondeterministic) guarded command language[1] (nGCL) to concepturalize a computer program and to include nondeterminism. For better understanding, we use an equivalent form[4] of nGCL that is similar to modern psydo-code:

$$C ::= \ x := e \ \mid \ C;C \ \mid \ \{C\}\square\{C\} \ \mid \ if\ (\varphi)\ \{C\}\ else\ \{C\} \ \mid \ while\ (\varphi)\ \{C\}$$

3 A Proof System

The goal of this thesis is to study the necessary liberal precondition and its relation to Hoare Triple. It would be beneficial to come up with a (complete and sound) proof system and apply it to important algorithms.

References

[1] E.W. Dijkstra (1975): *Guarded commands, nondeterminacy and formal derivation of programs*. Communications of the ACM 18(8), pp. 453–457.

[2] E.W. Dijkstra & C.S. Scholten (1990): *On substitution and replacement*. Springer New York, New York, NY. Available at `https://doi.org/10.1007/978-1-4612-3228-5_2`.

[3] C.A.R. Hoare (1969): *An axiomatic basis for computer programming*. Communications of the ACM 12(10), pp. 576–580.

[4] L. Zhang & B. Kaminski (2022): *Quantitative Strongest Post*. arXiv preprint arXiv:2202.06765.