# CMinor 程序验证工具实验报告

**2021011823 张国威**

## 一、 实现过程

根据课上所讲的基于基本路径分析的程序验证方法，一种合理的实现方案是：对每个函数，首先根据控制流图，求出所有基本路径。再对每一条基本路径 $\{\varphi\}\ st_1; st_2; ...; st_n\{\psi\}$，判断验证条件 $\varphi \to wp(st_1; st_2; ...; st_n, \psi)$ 是否成立。如果该验证条件对所有基本路径都成立，则程序具有部分正确性。然后，如果基本路径的首位置有秩函数，首先判断它有下界 $0$。若基本路径的末位置也有秩函数，则需要判断从首位置到末位置该秩函数的值是减小的，即 $\varphi \to wp(st_1; ...; st_n, \delta(x) \prec \delta(x'))[x' \mapsto x]$。如果该验证条件对所有基本路径都成立，则程序具有完全正确性。

**求基本路径的方法：** 维护一个队列，初始时只有函数头。先让函数头出队，通过深度优先搜索获得所有从函数头开始的基本路径。这些基本路径终止于循环头，函数调用前或函数出口。对于终止于循环头的基本路径，再将循环头全部入队。当这些循环头出队时，若未曾访问过，就又得到了从这些循环头开始的新的基本路径。以此类推。在深度优先搜索中，我也加入了对于已搜索过的位置的缓存，以提高效率。

**部分正确性的验证方法：** 先求出每条基本路径后置条件。其中，终止于函数出口和循环头的情况较为简单：前者访问 `conditions` 属性，后者访问 `invariants` 属性即可。对终止于函数调用前的情况，需要将被调用函数的前置条件中的形参替换为调用函数中的实参。然后，根据 `assume` 语句， `assert` 语句，以及变量和数组赋值语句的最弱前置条件计算规则，迭代地从后往前计算最弱前置条件，直到到达基本路径的初始位置。此时得到 $wp_0 = wp(st_1; st_2; ...; st_n, \psi)$，将 $\varphi \to wp_0$ 交给 `solver` 求解即可。

值得注意的是，对于函数调用语句 $st_i$，还需要判断将形参替换为实参后被调用函数的后置条件 $\psi_i'$ 满足 $\psi_i' \to wp(st_{i+1}; ...; st_n, \psi)$。由于有终止于函数调用前的基本路径，因此不需要在这里再次判断函数的前置条件是否满足。另外，我在求基本路径时，并没有考虑终止于 `assert` 语句的情况，而是利用 $wp(\textbf{assert}\ e, \psi) = e \wedge \psi$，像其他语句一样直接求最弱前置条件。这与单独拆出终止于 `assert` 语句的情况是等价的。

**完全正确性的验证方法：** 首先判断基本路径的首位置有没有秩函数。若有，则先判断每个秩函数是否都以 $0$ 为下界。若是，则继续判断末位置是否有秩函数。如果两端都有秩函数，则判断末位置和首位置是否满足字典序关系。在部分正确性验证中实现了计算 $wp$ 的函数后，我们只需要将部分正确性验证中 $wp$ 的后置条件变为这里的字典序关系条件即可。由于局部变量在首位置和末位置的值需要分开表示，因此需要为每个局部变量 $i$ 建立一个末位置表示 $i$ 和首位置表示 $i'$ 的双向映射。程序中，这可以用 `Dictionary` 来实现。

## 二、遇到的问题及解决方案

1. 求基本路径的方法比我一开始想的要复杂。我本来觉得仅靠一个简单的深度优先搜索就能实现，但其实对开和结束位置的限制也让问题变得复杂了一些。最终我采用了前面所述的一边维护一个队列，一边进行深度优先搜索的方式较为高效地实现了从控制流图到基本路径的构造。

2. 基本路径终止于函数调用前的情况其实与终止于函数出口和循环头的情况写起来不太一样。因为后两者可以分别直接访问该**位置**的属性（ `conditions` 和 `invariants` ）获得后置条件；对于终止于函数调用前的情况，后置条件则需要根据函数调用**语句**中右式的属性才能获得。因此，我在实现时额外在基本路径中加入了函数调用后的位置，以通过调用前和调用后的位置唯一确定函数调用语句。同时，为了标识这样的路径而不与其他情况混淆，我又在基本路径末尾再加入了一次调用后的位置。这样做是安全的，因为程序位置的后继位置不会是它本身。

3. 测例中存在这种较为隐蔽的不包含基本路径的情况：

   ```
   /*@
       requires \true;
       decreases -1;
       ensures \true;
   */
   void fun() {}
   ```

   需要单独判断秩函数以 $0$ 为下界且 $\varphi \to \psi$。

4. 由于已经有一段时间没写过 C 语言的代码了，对于面向对象程序设计中的一些知识已经有些生疏。不过，在本项目完善的报错工具，以及 ChatGPT 的帮助下，我最终解决了这些问题。

5. 本次大作业用到的 API 较多，各种类的属性和继承关系也十分复杂。最终实现的代码中有不少很长的属性访问序列，比如 `stmt.rhs.function.entryLocation.rankingFunctions` ，因此刚开始写的时候举步维艰。不过，在经过一段时间的 coding 和 debugging 后，就比较得心应手了。