Hoare Logic Notes

Jerry



1 公理系统

霍尔逻辑被用于形式化证明程序正确性，是一种公理系统。公理系统的详细内容可参见课本第 3 章，这里仅摘取课本中的一些重点内容。

公理系统指从一些公理出发，根据演绎规则推导出一系列定理形成的演绎体系。公理系统中包含：

- 初始符号，即公理系统所允许出现的全体符号集合。
- 形成规则，即公理系统中合法的符号序列。
- 公理，即公理系统中可以推演其他所有重言式的基本重言式。
- 变形规则，即公理系统中规定的推演规则。
- 建立定理，即公理系统中所有的重言式和对它们的证明。

公理系统有两个重要的性质：完备性和可靠性。公理系统的完备性是指所有的重言式都可以由公理系统中的公理和变形规则推导出来；公理系统的可靠性是指通过公理系统推导出的所有公式都是重言式。命题逻辑公理系统是完备且可靠的，但并不是所有的公理系统都是完备且可靠的。

2 霍尔三元组

要证明程序的正确性，首先要形式化定义什么是正确。霍尔逻辑使用霍尔三元组定义程序正确性。

$$\{P\}C\{Q\}$$

其中 P 是前置条件，C 是代码，Q 是后置条件。前置条件表示执行程序前的程序状态满足什么条件，后置条件表示程序执行后满足什么条件。例如 $\{a > 1\}a{+}{+}\{a > 2\}$ 表示如果程序执行前 $a > 1$，那么执行 $a{+}{+}$ 后满足 $a > 2$。

注意，这里讲的霍尔逻辑只能保证 partial correctness，意思是如果 $\{P\}C\{Q\}$ 成立，也不能保证程序 C 必定终止。

霍尔三元组只是定义如果程序能终止，那么程序会实现什么功能。也就是说，对任意的 P 和 Q，$\{P\}while(1);\{Q\}$ 都成立，因为 $while(1);$ 根本不能终止。

前置条件和后置条件常常表现为谓词逻辑公式。

3 公理和推理规则

并不是所有的霍尔三元组都成立，可以通过下面的公理和推理规则来证明一个霍尔三元组成立。

3.1 skip

skip 语句是什么都不做的意思，如果在执行 *skip* 前满足条件 P，那么执行 *skip* 后，因为 *skip* 什么都没做，所以 P 仍然成立。公式 1表示了这种关系。

$$\{P\}skip\{P\} \tag{1}$$

3.2 赋值

$$\{P\}a := e\{(\exists v)((a = e[v/a]) \wedge (P[v/a]))\} \tag{2}$$

公式 2是关于赋值语句的定理。其中，$a := e$ 表示将变量 a 赋值为表达式 e，$e[v/a]$ 表示将表达式 e 中的变量 a 替换为变量 v，$P[v/a]$ 表示将前置条件 P 中的变量 a 替换为变量 v。

该公式的含义是，如果在执行程序前 P 为真，则在将变量 a 赋值为表达式 e 后，存在一个值 v（即执行代码前 a 的旧值），使得执行代码后变量 a 的值为 $e[v/a]$，且 $P[v/a]$ 为真。例如，$\{x = 1\}x := x + 1\{(\exists v)(x = (x + 1)[v/x] \wedge (x = 1)[v/x])\}$ 成立，简化后为 $\{x = 1\}x := x + 1\{(\exists v)(x = v + 1 \wedge v = 1)\}$。

$$\{P[e/a]\}a := e\{P\} \tag{3}$$

公式 3是关于赋值语句的另一个定理，$P[e/a]$ 是把 P 中的 a 全部换成 e。意思是如果执行代码前 $P[e/a]$ 成立，那么执行 $a := e$，现在 a 的值变成了 e，所以 P 肯定成立。例如，$\{x + 1 = 2\}x := x + 1\{x = 2\}$ 成立。

3.3 推理

$$\frac{\{P\}C\{Q\}, Q \to R}{\{P\}C\{R\}} \tag{4}$$

公式 4的意思是，如果横线上面的式子都成立，那么横线下面的式子也都成立。如果执行前满足 P，执行 C 后满足 Q，Q 成立一定推出 R 成立，那么执行 C 后一定也能满足 R。

$$\frac{P \to Q, \{Q\}C\{R\},}{\{P\}C\{R\}} \tag{5}$$

类似的，公式 5表示如果 P 成立一定推出 Q 成立，执行前满足 Q 能保证执行 C 后满足 R，那么只要执行前满足 P，执行 C 后也一定能保证 R 成立。

3.4 顺序

$$\frac{\{P\}C1\{R\}, \{R\}C2\{Q\}}{\{P\}C1; C2\{Q\}} \tag{6}$$

公式 6表示了顺序结构代码的推理规则，$C1; C2$ 是说代码分两部分，前一部分是 $C1$，后一部分是 $C2$。如果 $\{P\}C1\{R\}$ $\{R\}C2\{Q\}$ 都成立，那么如果执行 $C1; C2$ 前 P 成立，执行完 $C1; C2$ 后 Q 一定成立。

3.5 条件

$$\frac{\{P \wedge istrue(b1)\}C1\{Q\}, \{P \wedge isfalse(b1)\}C2\{Q\}}{\{P\}if(b1)\ then\ C1\ else\ C2\{Q\}} \tag{7}$$

公式 7对应 if 语句的推理规则。if 语句有两条分支，$b1$ 成立时执行 $C1$，$b1$ 不成立时执行 $C2$。$istrue(b1)$ 表示 $b1$ 为 T，$isfalse(b1)$ 表示 $b1$ 为 F。

3.6 循环

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}while(b1)\ C\{I \wedge isfalse(b1)\}} \tag{8}$$

公式 8对应循环的推理规则。霍尔逻辑可以处理带有循环的程序，unbounded loop 也是可以处理的。

核心是要手动定义循环不变式（loop invariant），循环不变式是一个公式，在执行循环前成立，在每次执行完循环体后成立，在整个循环结束后成立。有了循环不变式，就可以刻画在第 n 次执行循环体后退出时满足的条件。

所以上面的推理规则中，横线上的 $\{I \wedge istrue(b1)\}C\{I\}$，实际是在说 I 确实是循环不变式，横线下面是说只要循环前循环不变式成立，那么循环执行结束后，I 一定成立并且循环条件 $b1$ 是 F。

4 Weakest Precondition

证明一个程序的正确性，需要通过编写前置条件、后置条件、循环不变式形式化定义什么是正确，然后通过推理证明霍尔三元组的正确性。这个推理过程如果通过手动完成就太麻烦了，最弱前置条件可以把霍尔逻辑公式转成谓词逻辑公式，进而利用 SMT solver 去自动化求解，实现自动化的形式化证明。

如果 $P1 \to P2$ 成立，那么就说 $P1$ 比 $P2$ 强，$P2$ 比 $P1$ 弱。给定程序 C 和后置条件 Q，$wp(C,Q)$ 是最弱前置条件意味着 $(\forall P)(\{P\}C\{Q\}) \Leftrightarrow (P \to wp(C,Q))$（$\Leftrightarrow$ 是等价）。根据最弱前置条件的定义，证明霍尔三元组 $\{P\}C\{Q\}$ 成立就等价于证明 $P \to wp(C,Q)$ 成立，这样就把霍尔逻辑的公式转成了谓词逻辑公式，接下来就交给强大的 SMT solver 来处理。

相似的，给定程序 C 和前置条件 P，也可以定义最强后置条件 $sp(C,P)$。$sp(C,P)$ 是最强后置条件意味着 $(\forall Q)(\{P\}C\{Q\}) \Leftrightarrow (sp(C,P) \to Q)$。根据最强后置条件的定义，证明霍尔三元组 $\{P\}C\{Q\}$ 成立就等价于证明 $sp(C,P) \to Q$ 成立，这样就把霍尔逻辑的公式转成了谓词逻辑公式，接下来就交给强大的 SMT solver 来处理。

因为计算最强后置条件会引入量词，增加 SMT solver 的求解难度，所以通常使用最弱前置条件实现自动化证明。

接下来使用 $wlp(C,Q)$ 而不是 $wp(C,Q)$，它们的区别是 wlp 不考虑程序是否终止，对应前面说的 partial correctness 的霍尔逻辑。实现自动化最后的问题就是如何自动计算 $wlp(C,Q)$，下面讲解给定程序 C 和后置条件 Q，如何自动化计算最弱前置条件。程序包含 $skip$、赋值语句、顺序语句、if 语句、循环语句。计算公式如下：

$$
\begin{aligned}
wlp(skip, P) &= P \\
wlp(a := e, P) &= P[e/a] \\
wlp(C1; C2, P) &= wlp(C1, wp(C2, P)) \\
wlp(if(b)\ then\ C1\ else\ C2, P) &= (istrue(b) \to wlp(C1, P)) \wedge (isfalse(b) \to wlp(C2, P))
\end{aligned}
$$

如果程序中有循环，那么需要用户手动定义循环不变式 I，然后证明 $\{P\}while(b)C\{Q\}$ 就变成证明下面 3 个公式都成立。

$$
\begin{aligned}
&P \to I \\
&\{I \wedge iftrue(b)\}C\{I\} \\
&(I \wedge isfalse(b)) \to Q
\end{aligned}
$$

这样循环就去掉了，接下来就可以按照前面的做法生成谓词逻辑公式，交给 SMT solver 验证。