编译原理扩展实验展示

于剑

2020 年 12 月 26 日

实验概览

实验概览

- 在一个自己规定的语言（mdverif）上实现了一个形式化验证的原型系统
- 程序员可以使用一套标注语法为程序添加标注，表示程序的期望行为，而验证器在编译器静态地尝试证明它
- 如果通过验证，则说明程序的运行时行为符合标注，即没有bug
- 它提供了对程序正确性的形式化证明

实验概览

- 程序员为每个函数标注 precondition 和 postcondition，即调用时需满足的条件和调用完成时应满足的条件
- 对于循环，还需标注循环不变式
- 用验证器验证程序，如果不能通过，可能是标注出了问题，也可能是程序本身有 bug

方法介绍

顺序执行路径上的验证

- 首先考虑一个顺序执行的函数，把执行路径上的动作分为三类：assume，assert 和程序本身的普通行为（如赋值）
- 程序的状态包含当前执行的位置和所有变量的值，可以用一阶逻辑公式来表示对某个状态的断言
- assume 和 assert 的对象都是对当时执行状态的断言，但前者提供信息，而后者要求验证断言
- 我们希望生成一个公式，称为 verification condition，它的永真性对应程序的正确性

公理系统

- 纯粹的一阶逻辑无法满足我们的要求：例如程序中用到整数运算，证明中往往需要考虑到整数运算的性质才行
- 引入相应的公理系统
- 在某个一阶理论中判断可满足性（判断 VC 的永真性等价于判断其否定的可满足性）
- 在这个实验中使用 Z3 Prover 来判断可满足性，并使用它预定义的 bit vector 提供有符号 32 位整数的性质

顺序执行路径上的验证

- 为了生成 verification condition，从后向前扫描程序，维护对当前位置程序状态的断言
- 最终得到的公式是关于函数执行的起始状态的
- 考虑对于 assume，assert 和赋值分别怎样维护 VC

控制语句

- 加入控制语句后，需要将函数的执行拆分为若干基本路径，每条路径使用前面的方法验证
- if 语句被基本路径跨过，分别生成条件为真和为假的路径，两种路径中分别 assume 条件为真/假
- 循环语句要求标注循环不变式，利用循环不变式将控制流拆开
- 在基本块间的控制流图上 dfs 以生成基本路径
- 循环要标注 rank function，一个固定长度的关于执行状态的整数序列
- 通过证明 rank function 中每个元素始终非负且字典序严格递减来证明程序会终止

成果展示

实验成果

- 支持 if, while 和 for 循环（因此支持循环不变式和 rank function 的标注）
- 验证时会输出 VC，以便程序员找到错误，调整标注或代码实现

运行展示

```
$ ./mdverif test.c
vc: (((ns0::n) >= (0)) && ((ns0::n) < (2147483647))) --> (((0) == ((1) - (1))) && ((((1) <= (1)) && ((1) <= ((ns0::n) +
(1)))) && ((((ns0::n) - (1)) + (1)) >= (0))))
vc: ((ns0::ret) == ((ns0::i) - (1))) --> ((((1) <= (ns0::i)) && ((ns0::i) <= ((ns0::n) + (1)))) --> ((((ns0::i) <= (ns0:
:n)) --> ((((ns0::ret) + (1)) == (((ns0::i) + (1)) - (1))) && ((((1) <= ((ns0::i) + (1))) && (((ns0::i) + (1)) <= ((ns0:
:n) + (1)))) && (((((ns0::n) - ((ns0::i) + (1))) + (1)) >= (0)) && ((((ns0::n) - ((ns0::i) + (1))) + (1)) < (((ns0::n) -
 (ns0::i)) + (1))))))) && ((((ns0::n) - (ns0::i)) + (1)) >= (0))))
vc: ((ns0::ret) == ((ns0::i) - (1))) --> ((((1) <= (ns0::i)) && ((ns0::i) <= ((ns0::n) + (1)))) --> (((! ((ns0::i) <= (n
s0::n))) --> ((ns0::ret) == (ns0::n))) && ((((ns0::n) - (ns0::i)) + (1)) >= (0))))
Verified.
```

```
$ ./mdverif test_fail.c
vc: ((ns0::n) >= (0)) --> (((0) == ((1) - (1))) && ((((1) <= (1)) && ((1) <= ((ns0::n) + (1)))) && ((((ns0::n) - (1)) +
(1)) >= (0))))
Verification failed in a basic path in function sum
Verification failed.
```

Q&A

Q&A