Tracing Programs with Assertions

Venkatesh Choppella

2023-11-18 10:19:17+05:30

IIIT Hyderabad

Contents

What is a Program Trace?

- A program trace is a sequence of observations made during the execution of a program.
- Each observation records the value of a set of program variables or the relation between the program variables.

What is the use of a Program Trace?

- A program trace is a representation of one's mental model of how a program runs.
- A program trace could be used to understand the running of a program and isolate bugs either in your program or your understanding of it.

How does one represent a Program Trace?

- There is no standard way.
- Many people use tables, or diagrams.
- The purpose of this lecture is to propose a format that makes the program trace resemble a mathematical proof.

Contents

What are Assertions?

An assertion is a judgement or a claim about something. When programming, the assertions usually involve program variables and their current values.

1. $x = 2$
2. $a = 5$
3. $f(3) = 7$
4. $a = 3 \land x = 2$
5. $a > 3 = \mathit{true}$

Tracing *fact_iter*(1) with assertions

```
1  int fact_iter(int n) {
2      int i = n;
3      int a = 1;
4      while (i > 0) {
5          a = a * i;
6          i = i - 1;
7      };
8      return a;
9  };
```

1		
2	*fact_iter*(1)	Given
3	$n = 1$	L1, 2
4	$i = 1$	L2, 3
5	$a = 1$	L3
6	$i > 0 = \mathit{true}$	L4, 4
7	$a = 1$	L5, 4, 5
8	$i = 0$	L6, 4
9	$i > 0 = \mathit{false}$	L4, 8
10	*ret* 1	L8, 7
11	$\mathit{fact\_iter}(1) = 1$	Call, 2—10

Structure of an Assertion Trace

1. A trace of a program where each line is an assertion.
2. The assertion is justified by the program line number L and previous assertions.
3. A function call opens a new block of assertions.
4. A return closes the current block of assertions.
5. Upon return, an assertion $fn\_call(args...) = return\_value$ is made.
6. Once a block is closed, none of its assertions are available as justifications for any subsequent assertion.

Tracing $fact\_rec(1)$ with assertions

```
1  int fact_rec(int n) {
2      int a;
3      if (n > 0)
4          a = n * fact_rec(n-1);
5      else
6          a = 1;
7      return a;
8  };
```

1		
2	$fact\_rec(1)$	Given
3	$n = 1$	L1, 2
4	$n > 0 = true$	L3, 3
5	$fact\_rec(0)$	L4, 3
6	$n = 0$	L1, 5
7	$n > 0 = false$	L3, 6
8	$a = 1$	L6
9	$ret\ 1$	L7, 8
10	$fact\_rec(0) = 1$	Call, 5—9
11	$a = 1$	L4, 3, 10
12	$ret\ 1$	L7, 11
13	$fact\_rec(1) = 1$	Call, 2—12