Introduction to Mathematical Logic

For CS Students

CS104/CS108

Yida TAO (陶伊达)

2024 年 5 月 20 日

Table of Contents

1 Warm up

- Warm up
- Program Verification
- Partial and Total Correctness
- Program variables and logical variables

Program Correctness

How do we show that a program works correctly (it does what it is supposed to do)?

- Manually review the code
  - Expensive and error-prone
- Testing
  - Check a program for carefully chosen inputs.
  - Cannot be exhaustive ("Testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence". E. Dijkstra.)
- Formal verification (形式化验证)
  - Mathematically prove that a program satisfies some properties for all inputs.

Why formal verification?

- There is a great advantage in being able to verify the correctness of computer systems, whether they are hardware, software, or a combination.
- Most obvious in the case of safety-critical systems, but also applies to those that are commercially critical, such as mass-produced chips, mission critical, etc.

Without formal verification, what could go wrong?

- Therac-25, X-ray, 1985
  - Overdosing patients during radiation treatment, 5 dead
  - Reason: race condition between concurrent tasks
- AT&T, 1990
  - Long distance service fails for 9 hours.
  - Reason: wrong BREAK statement in C code
- Patriot-Scud, 1991
  - 28 dead and 100 injured
  - Reason: rounding error
- Pentium Processor, 1994
  - lost millions of dollars
  - Reason: incomplete entries in a look-up table

Without formal verification, what could go wrong?

- Ariane 5, 1996
  - Exploded 37 seconds after takeoff
  - Reason: data conversion of a too large number
- Mars Climate Orbiter, 1999
  - Destroyed on entering atmosphere of Mars
  - Reason: mixture of pounds and kilograms
- Power black-out, 2003
  - 50 million people in Canada and US without power
  - Reason: programming error
- Royal Bank, 2004
  - Financial transactions disrupted for 5 days
  - Reason: programming error

Without formal verification, what could go wrong?

- UK Child Support Agency, 2004
  - Overpaid 1.9 million people, underpaid 700,000, cost to taxpayers over $ 1 billion
  - Reason: more than 500 bugs reported
- Science (a prestigious scientific journal), 2006
  - Retraction of research papers due to erroneous research results
  - Reason: program incorrectly flipped the sign (+ to -) on data
- Toyota Prius, 2007
  - 160,000 hybrid vehicles recalled due to stalling unexpectedly
  - Reason: programming error
- Knight Capital Group, 2012
  - High-frequency trading system lost $ 440 million in 30 min
  - Reason: programming error

Table of Contents

2 Program Verification

- ▶ Warm up
- ▶ Program Verification
- ▶ Partial and Total Correctness
- ▶ Program variables and logical variables

Process

The process of program verification:

1. Convert an informal description $\mathcal{R}$ of requirements for a program into a logical formula ϕ_R.
2. Write a program $\mathcal{P}$ which is meant to satisfy the requirements $\mathcal{R}$ above.
3. Prove that program $\mathcal{P}$ satisfies the formula ϕ_R.

We will focus on the third part.

Programming Language

2 Program Verification

We will use a programming language that is a subset of C/C++ and Java, whose core features include:

- integer and boolean expressions, with symbols of $+$, $-$, $\leq$......
- assignment statements
- conditional statements
- while loops

Imperative Programs

- Executing an imperative (命令式) program has the effect of changing the state, i.e. the values of program variables.
- To use an imperative program, first establish an initial state, i.e. set some variables to have values of interest
- Then, execute the program to transform the initial state into a final one.
- One then inspects the values of variables in the final state to get the desired results.

Program States

2 Program Verification

Consider a program that computes the factorial of input x and store it in y. What are program states at “while”?

```
y = 1 ;
z = 0 ;
while ( z != x ) {
   z = z + 1 ;
   y = y * z ;
}
```

- Initial state s_0: $z = 0, y = 1$
- Next state s_1: $z = 1, y = 1$
- State s_2: $z = 2, y = 2$
- State s_3: $z = 3, y = 6$
- State s_4: $z = 4, y = 24$ ......

Formal specification

Requirements for a program (程序需求): describe what customers want (e.g., “I want a program that sorts an array in descending order”).

But, natural language requirements are often:

- inconsistent, ambiguous, hard to follow, cannot be reasoned about

Specification for a program (程序规范): formalize the requirements. A formal specification can be considered as a *contract*.

A formal specification:

- describes a program’s effect in a mathematically precise notation
- can be used to prove the correctness of a program

Formal Specification

2 Program Verification

Two important components of a specification:

- Precondition: The state before the program executes.
- Postcondition: The state after the program executes.

Our formal notation:

- is based on FOL logic
- applies to imperative programs
- We want to develop a notion of proof that will allow us to prove that a program satisfies the specification given by the precondition and the postcondition. This is done using Hoare triples.

Hoare Triples

- Proposed by Sir Charles Antony Richard Hoare. British computer scientist.
- Developed the QuickSort algorithm and the Hoare logic for verifying program correctness.
- Won Turing award in 1980.

Hoare Triples

A Hoare triple $\llparenthesis P \rrparenthesis\ C\ \llparenthesis Q \rrparenthesis$

- Precondition: P (FOL formula)
- Code to be verified: C
- Postcondition: Q (FOL formula)

A specification of a program C is a Hoare triple with C as the second component $\llparenthesis P \rrparenthesis\ C\ \llparenthesis Q \rrparenthesis$, which means:

> If the state of program C before execution satisfies P, then the ending state of C after execution will satisfy Q.

Examples

2 Program Verification

The requirement

I want a program that swaps the values in x and y

might be expressed as:

$$(\!| x = a \wedge y = b |\!)\ C\ (\!| x = b \wedge y = a |\!)$$

执行代码

Is the specification true when C is:

- $r = x; x = y; y = r$ ✓
- $x = y; y = x$ $x=b$ $y=b$ ✗

Examples

The requirement

If the input x is a positive integer, compute a number whose square is less than x

might be expressed as:

$$(\!|\, x > 0 \,|\!)\ C\ (\!|\, y * y < x \,|\!)$$

Specification is NOT behavior

Consider the program P:

```
y=0;
```

Is the Hoare triple

$$(\!|\, x > 0 \,|\!)\ P\ (\!|\, y * y < x \,|\!)$$

satisfied?

Specification is NOT behavior

Consider the program P:

```
y = 0 ;
while ( y * y < x ) {
    y = y + 1 ;
}
y = y - 1 ;
```

Is the Hoare triple

$$\llparenthesis x > 0 \rrparenthesis \; P \; \llparenthesis y * y < x \rrparenthesis \quad ✓$$

satisfied?

Specification can be tricky

If the requirement "the program must set y to the maximum of x and y" is expressed as:

$$(\!|\, True \,|\!)\ C\ (\!|\, y = max(x, y) \,|\!)$$

Then the following programs are all "correct" w.r.t this specification:

- if $x >= y$ then $y = x$ else skip
- if $x >= y$ then $x = y$ else skip
- $y = x$

cuz the postcondition $y = max(x, y)$ says "y is the maximum of x and y in the final state."

Specification can be tricky

2 Program Verification

For the requirement “the program must set y to the maximum of x and y”, rather than being expressed as:

$$(\!|\, True \,|\!)\ C\ (\!|\, y = max(x, y) \,|\!)$$

the correct formalization of what was intended is probably:

这样就不一定对 (x=y)

$$(\!|\, x = a \wedge y = b \,|\!)\ C\ (\!|\, y = max(a, b) \,|\!)$$

如果 intial state 是 false

就不能用来证

Specification can be tricky

The lessons:

- It is easy to write the wrong specification!
- A proof system will NOT help since the incorrect program could have been proved “correct”.

Table of Contents

3 Partial and Total Correctness

- ▶ Warm up
- ▶ Program Verification
- ▶ Partial and Total Correctness
- ▶ Program variables and logical variables

Partial Correctness

Hoare triple $\llparenthesis P \rrparenthesis\ C\ \llparenthesis Q \rrparenthesis$ is satisfied under partial correctness iff

- for every state s_1 that satisfies P,
- if execution of C starting from state s_1 terminates in a state s_2,
- then state s_2 satisfies condition Q.

which is denoted as

$$\models_{par} \llparenthesis P \rrparenthesis\ C\ \llparenthesis Q \rrparenthesis$$

(In other sources, we may also describe it as “$\llparenthesis P \rrparenthesis\ C\ \llparenthesis Q \rrparenthesis$ holds/is valid/is true”)

Examples

Is the following Hoare triple satisfied under partial correctness?

- $\llparenthesis x = 1 \rrparenthesis\ y = x\ \llparenthesis y = 1 \rrparenthesis$ ✓
- $\llparenthesis x = 1 \rrparenthesis\ y = x\ \llparenthesis y = 2 \rrparenthesis$ ✗
- $\llparenthesis x = 1 \rrparenthesis$ while True do SKIP $\llparenthesis y = 2 \rrparenthesis$ ✓ 不会 terminate ⇒ Assumption is false
- $\llparenthesis x = 1 \rrparenthesis$ while $x = 1$ do $y = 2$ $\llparenthesis x = 3 \rrparenthesis$ ✓
- $\llparenthesis i > j \rrparenthesis\ i = i + 1; j = j + 1;\ \llparenthesis i > j \rrparenthesis$ ✓
- $\llparenthesis i \geq 0 \rrparenthesis\ i = i - 1;\ \llparenthesis i \geq 0 \rrparenthesis$ ✗ $i=0$
- $\llparenthesis True \rrparenthesis$ if $i = j$ then $i = i + 1$ else $y = 0$ $\llparenthesis i \neq j \rrparenthesis$ ✗

Examples

3 Partial and Total Correctness

Consider the Hoare triple

$$(\!| x > 0 |\!)\ C\ (\!| y * y < x |\!)$$

If we run C starting with the state $x = 5; y = 5$, C terminates in the state $x = 5; y = 0$.

Is the Hoare triple satisfied under partial correctness?

- **A.** Yes
- **B.** No
- [✓ **C.** Not enough information to tell]

Examples

Consider the Hoare triple

$$\llparenthesis x > 0 \rrparenthesis \; C \; \llparenthesis y * y < x \rrparenthesis$$

If we run C starting with the state $x = 5; y = 5$, C terminates in the state $x = 5; y = 3$.

Is the Hoare triple satisfied under partial correctness?

A. Yes

B. [✓ No]

C. Not enough information to tell

Examples

Consider the Hoare triple

$$\llparenthesis x > 0 \rrparenthesis \; C \; \llparenthesis y * y < x \rrparenthesis$$

If we run C starting with the state $x = -3; y = 5$, C terminates in the state $x = -3; y = 0$.

Is the Hoare triple satisfied under partial correctness?

- **A.** Yes
- **B.** No
- [✓ **C.** Not enough information to tell]

Examples

3 Partial and Total Correctness

Consider the Hoare triple

F

$$\llparenthesis x > 0 \land x < -1 \rrparenthesis \; C \; \llparenthesis y * y < x \rrparenthesis$$

Is the Hoare triple satisfied under partial correctness?

A. Yes

B. [✓ No]

C. Not enough information to tell

Examples

要知道任意起始状态

Consider the Hoare triple

$$(\!|\, x > 0 \,|\!)\; C \;(\!|\, y * y < x \,|\!)$$

If we run C starting with the state $x = 2; y = 5$, C does not terminate.

Is the Hoare triple satisfied under partial correctness?

A. Yes

B. No

[✓ **C.** Not enough information to tell]

Partial Correctness

3 Partial and Total Correctness

Partial correctness is rather a weak requirement, since any program which does not terminate at all satisfies its specification.
For example, an endless loops that never terminates satisfies all specifications, since partial correctness only says what must happen *if* the program terminates.

We say a program is partially correct if it gives the right answer whenever it terminates. It never gives a wrong answer, but it may give no answer at all.

Total Correctness

3 Partial and Total Correctness

A Hoare triple $\llparenthesis P \rrparenthesis\ C\ \llparenthesis Q \rrparenthesis$ is satisfied under total correctness iff.

- for every state s_1 that satisfies condition P,
- execution of C starting from state s_1 terminates in a state s_2,
- and state s_2 satisfies condition Q.

which is denoted as

$$\vDash_{tot} \llparenthesis P \rrparenthesis\ C\ \llparenthesis Q \rrparenthesis$$

Intuitively,

Total correctness = Partial correctness + Termination

Examples

Is the following Hoare triple satisfied under partial and/or total correctness?

$$\llparenthesis\, x = 1 \,\rrparenthesis\; y = x \;\llparenthesis\, y = 1 \,\rrparenthesis$$

A. Neither satisfied

B. Only partial correctness satisfied

[✓ **C.** Total correctness satisfied]

Examples

Is the following Hoare triple satisfied under partial and/or total correctness?

$$(\!|\, x = 1 \,|\!)\ y = x\ (\!|\, y = 2 \,|\!)$$

A. [✓ Neither satisfied]

B. Only partial correctness satisfied

C. Total correctness satisfied

Examples

3 Partial and Total Correctness

Is the following Hoare triple satisfied under partial and/or total correctness?

$$\llparenthesis\, x = 1 \,\rrparenthesis\ \underline{\text{while True do } x = 0}\ \llparenthesis\, x > 0 \,\rrparenthesis$$

A. Neither satisfied

B. [✓ Only partial correctness satisfied]

C. Total correctness satisfied

Examples

Is the following Hoare triple satisfied under partial and/or total correctness?

$$(\!|\, x \geq 0 \,|\!)\; C\; (\!|\, y = x! \,|\!)$$

in which C is:

```
y = 1 ; z = 0 ;
while ( z != x ) { z = z + 1 ; y = y * z ; }
```

只包含整数

A. Neither satisfied

B. Only partial correctness satisfied

[✓ **C.** Total correctness satisfied]

Examples

3 Partial and Total Correctness

Is the following Hoare triple satisfied under partial and/or total correctness?

$$\underline{(\!|\, True \,|\!)}\; C \;(\!|\, y = x! \,|\!)$$

$x \geq 0$ ✓

$x < 0$ 不终止

in which C is:

```
y = 1 ; z = 0 ;
while ( z != x ) { z = z + 1 ; y = y * z ; }
```

A. Neither satisfied

[✓ **B.** Only partial correctness satisfied]

C. Total correctness satisfied

Table of Contents

4 Program variables and logical variables

- Warm up
- Program Verification
- Partial and Total Correctness
- Program variables and logical variables

Definitions

4 Program variables and logical variables

The variables which we have seen so far in the programs that we verify are called program variables. They can also appear in the preconditions and postconditions of specifications.

Sometimes, in order to formulate specifications, we need to use other variables which do not appear in programs.

Examples

4 Program variables and logical variables

The following program C also computes the factorial of input x.

```
y = 1;
while (x != 0) {
    y = y * x;
    x = x - 1;
}
```

x的值会不断更新

x=0 终止

$y = 0! = 1$

It is not a good idea to express the Hoare triple as $\llparenthesis\, x \geq 0 \,\rrparenthesis\ C\ \llparenthesis\, y = x! \,\rrparenthesis$, why?

Examples

The following program C also computes the factorial of input x.

```
y = 1;
while (x != 0) {
    y = y * x;
    x = x - 1;
}
```

Instead, we can use $(\!|\, x = x_0 \land x \geq 0 \,|\!) \; P \; (\!|\, y = x_0! \,|\!)$.

Examples

4 Program variables and logical variables

```
y = 1;
while (x != 0) {
    y = y * x;
    x = x - 1;
}
```

In $\llparenthesis\, x = x_0 \wedge x \geq 0 \,\rrparenthesis\ C\ \llparenthesis\, y = x_0! \,\rrparenthesis$, x_0 is a logical variable and This specification reads: for all integers x_0, if x equals x_0, $x \geq 0$ and we run the program such that it terminates, then the resulting state will satisfy y equals $x_0!$.

This works since x_0 cannot be modified by C as x_0 does not occur in C.

Examples

The following program C adds up the first x integers and stores the result in z.

```
z = 0;
while (x > 0) {
    z = z + x;
    x = x - 1;
}
```

How to express the program as a Hoare triple?

Examples

The following program C adds up the first x integers and stores the result in z.

```
z = 0;
while (x > 0) {
    z = z + x;
    x = x - 1;
}
```

How to express the program as a Hoare triple?

$\llparenthesis\, x = x_0 \wedge x \geq 0 \,\rrparenthesis\ C\ \llparenthesis\, z = x_0(x_0+1)/2 \,\rrparenthesis$

Definitions

4 Program variables and logical variables

For a Hoare triple $(\!| P |\!)\ C\ (\!| Q |\!)$, its set of logical variables (also called auxiliary variables or ghost variables) are those variables that are free in P or Q, and don't occur in C.

The state of the system gives a value to each program variable, but not for the logical variables.

Readings

4 Program variables and logical variables

- TextB: chapter 4.1, 4.2

Introduction to Mathematical Logic

Thank you for listening!
Any questions?