Correctness ("Hoare") Triples

Part 1: Definitions and Basic Properties

CS 536: Science of Programming, Fall 2021

A. Why

- To specify a program's correctness, we need to know its precondition and postcondition (what should be true before and after executing it).
- The semantics of a verified program combines its program semantics rule with the state-oriented semantics of its specification predicates.

B. Objectives

At the end of today you should know

- The syntax of correctness triples (a.k.a. Hoare triples).
- What it means for a correctness triples to be satisfied or to be valid.
- That a state in which a correctness triple is not satisfied is a state where the program has a bug.

C. Correctness Triples ("Hoare Triples")

- A ***correctness triple*** (a.k.a. "***Hoare triple***," after C.A.R. Hoare) is a program *S* plus its specification predicates *p* and *q*.
    - The ***precondition*** *p* describes what we're assuming is true about the state before the program begins.
    - The ***postcondition*** *q* describes what should be true about the state after the program terminates.
- ***Syntax of correctness triples***: *{p} S {q}* (Think of it as */* p */ S /* q */*)

  ⇒ *Note: The braces are not part of the precondition or postcondition* ⇐
- The precondition of *{p} S {q}* is *p*, not *{p}*. Similarly the postcondition is *q*, not *{q}*. Saying "The precondition is *{p}*" is like saying "In C, the test in '***if** (B) x++;*' is '***if** (B)*'".

D. Satisfaction and Validity of a Correctness Triple

- Informally, for a state to ***satisfy*** *{p} S {q}*, it must be that if we run *S* in a state that satisfies *p*, then after running *S*, we should be in a state that satisfies *q*. For a triple to be ***valid***, it must be satisfied in all states.
- ***Important***: If we start in a state that doesn't satisfy *p*, we claim nothing about what happens when you run *S*.

- In some sense, "the triple is satisfied" means "the triple is not buggy".
- Say you (as the user) have been told not to run *S* when $x < 0$ because *S* calculates $sqrt(x)$.
- And say the triple is $\{x \geq 0\}\ y := sqrt(x)\ \{y^2 \leq x < (y+1)^2\}$.
- You can't say this program has a bug when you start in a state with $x < 0$, even though the program fails, because you ran the program on bad input.

- Analogous to $\sigma \vDash p$ and $\vDash p$ for satisfaction and validity of predicates, we'll use the notations $\sigma \vDash \{p\}\ S\ \{q\}$ and $\vDash \{p\}\ S\ \{q\}$ for satisfaction and validity.

E. Simple Informal Examples of Correctness

- Before going to the formal definitions of partial and total correctness, let's look at some simple examples, informally.
- ***Example 1***: $\vDash \{x > 0\}\ x := x+1\ \{x > 0\}$. This is satisfied in all states, so the triple is valid.
- ***Example 2***: $\nvDash \{x > 0\}\ x := x\text{-}1\ \{x > 0\}$. This is not satisfied (= "has a bug") in the state where *x* is *1*. (That is, $\{x = 1\} \nvDash \{x > 0\}\ x := x\text{-}1\ \{x > 0\}$.) So this triple is not valid because it has a bug.
- There are a number of ways to fix the buggy program in Example 2:
  - ***Example 3***: Make the precondition "***stronger***" = "more restrictive":
    $\vDash \{x > 1\}\ x := x\text{-}1\ \{x > 0\}$ or $\vDash \{x\text{-}1 > 0\}\ x := x\text{-}1\ \{x > 0\}$
  - ***Example 4***: Make the postcondition "***weaker***" = "less restrictive":
    $\vDash \{x > 0\}\ x := x\text{-}1\ \{x > \text{-}1\}$
  - ***Example 5***: Change the program: E.g., $\{x > 0\}$ ***if*** $x > 1$ ***then*** $x := x\text{-}1$ ***fi*** $\{x > 0\}$
- ***Example 6***: $\vDash \{(x = 2*k \vee x = 2*k+1) \wedge x \geq 0\}\ x := x/2\ \{x = k \geq 0\}$
  (If *x* is nonnegative and equals *2*k* or *2*k+1* before dividing *x* by *2* then after the division, *x* equals *k*, which is nonnegative.)
- ***Example 7***: $\vDash \{s = 1+2+ \ldots +k\}\ s := s+k+1;\ k := k+1\ \{s = 1+2+ \ldots +k\}$
  (If *s* = the sum of *1* through *k*, then after adding *k+1* to *s* and *1* to *k*, *s* is still the sum of *1* through *k*.)
- ***Example 8***: $\vDash \{s = 1+2+ \ldots + k\}\ k := k+1;\ s := s+k\ \{s = 1+2+ \ldots + k\}$
  (This is like Example 7 but we increment *k* first and then update *s* by adding *k* (not *k+1*) to it.)
- ***Example 9***:

  $\vDash$ $\{s = 1 + 2 + \ldots + k\}$
  $k := k+1;$
  $s := s+k+1$
  $\{s = 1 + 2 + \ldots + (k - 1) + (k+1)\}$
- (This is like Example 8 but we increment *k* and then add *k* (not *k+1*) to *s*. Hope it's okay that *s* is not the sum of *1* through *k*.)
- ***Definition:*** For a triple $\{p\}\ S\ \{q\}$, a variable that appears in *S* is a ***program variable***; a variable that appears in *p* or *q* is a ***condition variable***. A ***logical variable*** is a condition variable that is not also

a program variable: It appears in the logical reasoning about the program but not the program itself. ("Logical" in this context doesn't mean "Boolean".)

- ***Example 10***: $\vDash \{x = c_0 \geq 0\}\ x := x/2\ \{c_0 \geq 0 \wedge x = c_0 / 2\}$
(If x is ≥ 0, then after the assignment $x := x/2$, the old value of x (call it c_0) was ≥ 0 and x = its old value divided by 2. Note c_0 is a ***logical constant***, a logical variable that is a named constant.

F. Having a Set of States that Satisfy a Predicate

- Before looking at the definitions of program correctness, it will help if we extend the notion of a single state satisfying a predicate to having a set of states satisfying a predicate.
- ***Notation***: Recall that $\Sigma_\perp = \Sigma \cup \{\perp\}$, where Σ is the set of all (well-formed, proper) states.
  - Then, $\sigma \in \Sigma_\perp$ allows $\sigma = \perp$, but $\sigma \in \Sigma$ implies $\sigma \neq \perp$.
  - Similarly for a set of states Σ_0, if $\Sigma_0 \subseteq \Sigma_\perp$, then we may have $\perp \in \Sigma_0$.
  - On the other hand, if $\Sigma_0 \subseteq \Sigma$, then $\perp \notin \Sigma_0$.
- ***Notation***: $\Sigma_0 - \perp$ means $\Sigma_0 \cap \Sigma$, which is the set of all non-$\perp$ members of Σ_0.
- ***Definition***: Let $\Sigma_0 \subseteq \Sigma_\perp$. We say Σ_0 ***satisfies*** p if every element of Σ_0 satisfies p. In symbols, $\Sigma_0 \vDash p$ iff for all $\tau \in \Sigma_0$, $\tau \vDash p$. (Note $\varnothing \vDash p$, since there exists no $\tau \in \varnothing$ where $\tau \nvDash p$.)[1]
- Some consequences of the definition:
  - If $\perp \in \Sigma_0$, then $\Sigma_0 \nvDash p$ and $\Sigma_0 \nvDash \neg p$.
  - $(\Sigma_0 \vDash p$ and $\Sigma_0 \vDash \neg p)$ iff $\Sigma_0 = \varnothing$.
    - Since $\perp \nvDash p$ (and $\nvDash \neg p$), we have $\perp \notin \Sigma_0$. If $\tau \neq \perp$ and $\tau \vDash p$ then $\tau \nvDash \neg p$, so $\tau \notin \Sigma_0$. So $\Sigma_0 = \varnothing$.
  - If Σ_0 has size ≥ 2 and $\perp \notin \Sigma_0$, then $\Sigma_0 \nvDash \neg p$ iff $\Sigma_0 \vDash p$.
    - Either $\tau \vDash p$ or $\tau \vDash \neg p$ but not both, so ($\tau \vDash p$ and $\tau \nvDash \neg p$) or ($\tau \nvDash p$ and $\tau \vDash \neg p$).
  - If Σ_0 has size ≥ 2 and $\perp \notin \Sigma_0$, then it is ***not*** the case that $\Sigma_0 \nvDash p$ iff $\Sigma_0 \vDash \neg p$.
    - ($\Leftarrow$) If $\tau \vDash \neg p$ then $\tau \nvDash p$, so if $\tau \in \Sigma_0$, then $\Sigma_0 \nvDash p$.
    - ($\Rightarrow$) If $\perp \notin \{\tau, \tau'\} \subseteq \Sigma_0$ where $\tau \vDash p$ and $\tau' \vDash \neg p$, then $\tau \nvDash \neg p$ (so $\tau \in \Sigma_0$ implies $\Sigma_0 \nvDash \neg p$) and $\tau' \nvDash p$ (so $\tau' \in \Sigma_0$ implies $\Sigma_0 \nvDash p$). So we have $\Sigma_0 \nvDash p$ and $\Sigma_0 \nvDash \neg p$ simultaneously.

G. Total Correctness

- Normally, we want our programs to always terminate in states satisfying their postcondition (assuming we start in a state satisfying the precondition). This property is called ***total correctness***.
- ***Definition***: The triple $\{p\}\ S\ \{q\}$ is ***totally correct in*** σ or σ satisfies the triple under ***total correctness*** iff it's the case that if σ satisfies p, then running S in σ always terminates in states satisfying q.[2]

---

[1] If you run across an old set of these notes, you should know I changed how the notation works in F'20.

[2] The sense of "implies" or "if… then…" used here is not like → (which appears in predicates) or ⇒ (which is a relationship between predicates). It's "if…then" at a semantic level: If this triple is satisfied or if this set is nonempty, then … holds.

- In symbols, $\sigma \vDash_{tot} \{p\}\ S\ \{q\}$ iff $\sigma \neq \bot$ and (if $\sigma \vDash p$ then $\bot \notin M(S, \sigma)$ and $M(S, \sigma) \vDash q$).
    - The $\bot \notin M(S, \sigma)$ clause is redundant because $M(S, \sigma) \vDash q$ implies $\bot \notin M(S, \sigma)$.
- For total correctness, we can't allow $\sigma = \bot$ because $\bot \nvDash p$ and $M(S, \bot) = \{\bot\} \nvDash q$, so ($\sigma \vDash p$ implies $M(S, \sigma) \vDash q$) would reduce to (false implies false), which is true.
- Note for $\sigma \vDash_{tot} \{p\}\ S\ \{q\}$ we specifically require $\sigma \neq \bot$ because $\bot \nvDash p$, so without banning $\bot$ explicitly, we'd have ($\sigma \vDash p \Rightarrow \ldots$) turn into (false implies …), which is true.
- ***Definition***: The triple $\{p\}\ S\ \{q\}$ is ***totally correct*** (is ***valid*** under ***total correctness***) iff $\Sigma \vDash_{tot} \{p\}\ S\ \{q\}$. I.e., $\sigma \vDash_{tot} \{p\}\ S\ \{q\}$ for all $\sigma \in \Sigma$ (Recall Σ is the set of well-formed proper states.)
    - ***Notation***: We also write $\vDash_{tot} \{p\}\ S\ \{q\}$ to mean that the triple is totally correct.

H. Partial vs Total Correctness

- It turns out that reasoning about total correctness can be broken up into two steps: Determine "partial" correctness, where we ignore the possibility of divergence or runtime errors, and then show that those errors won't occur.
- ***Definition***: The triple $\{p\}\ S\ \{q\}$ is ***partially correct in*** σ or σ satisfies the triple under ***partial correctness*** iff it's the case that if σ satisfies p, then whenever running S in σ converges to a memory state, that state satisfies q.
- In symbols, $\sigma \vDash \{p\}\ S\ \{q\}$ iff $\sigma \neq \bot$ and ($\sigma \vDash p$ implies (for every $\tau \in M(S, \sigma)$, if $\tau \in \Sigma$, then $\tau \vDash q$)).
- Equivalently, $\sigma \vDash \{p\}\ S\ \{q\}$ iff $\sigma \in \Sigma$ and ($\sigma \vDash p$ implies $M(S, \sigma) - \bot \vDash q$).
- As with total correctness, we can't allow $\sigma = \bot$ for partial correctness because $\bot \nvDash p$, which would make ($\sigma \vDash p \Rightarrow \ldots$) true.
- ***Definition***: The triple $\{p\}\ S\ \{q\}$ is ***partially correct*** (i.e., is ***valid*** under/for ***partial correctness***) iff $\Sigma \vDash \{p\}\ S\ \{q\}$. I.e, $\sigma \vDash \{p\}\ S\ \{q\}$ for all states σ. ***Notation***: We also write $\vDash \{p\}\ S\ \{q\}$.

I. More Phrasings of Total and Partial Correctness

- An equivalent way to understand partial and total correctness uses the property that if $\sigma \neq \bot$, then ($\sigma \vDash \neg p$ iff $\sigma \nvDash p$) and ($\sigma \vDash p$ iff $\sigma \nvDash \neg p$).
- For total correctness, if $\sigma \neq \bot$, then
    - $\sigma \vDash_{tot} \{p\}\ S\ \{q\}$
iff $\sigma \vDash p$ implies $M(S, \sigma) \vDash q$
iff $\sigma \vDash \neg p$ or $M(S, \sigma) \vDash q$
iff $\sigma \vDash \neg p$ or $\tau \vDash q$ for every $\tau \in M(S, \sigma)$
    - If S is deterministic, then for some τ, $M(S, \sigma) = \{\tau\}$ and $\tau \vDash q$ (so we know $\tau \neq \bot$).
    - If S is nondeterministic, then for every $\tau \in M(S, \sigma)$, we have ($\tau \neq \bot$ and) $\tau \vDash q$.
- For partial correctness, if $\sigma \neq \bot$, then
    - $\sigma \vDash \{p\}\ S\ \{q\}$
iff $\sigma \vDash p$ implies $M(S, \sigma) - \bot \vDash q$

iff $\sigma \models \neg p$ or $M(S, \sigma) - \perp \models q$

iff $\sigma \models \neg p$ or for every $\tau \in M(S, \sigma)$, either $\tau = \perp$ or $\tau \models q$.

- If S is deterministic, then there is only one τ in $M(S, \sigma)$, and either $\tau = \perp$ or $\tau \models q$.

J. Unsatisfied Correctness Triples

- It's useful to figure out when a state ***doesn't satisfy*** a triple because not satisfying a triple tells you that there's some sort of bug in the program.

Unsatisfied Total Correctness

- For a state $\sigma \neq \perp$ to not satisfy $\{p\}\ S\ \{q\}$ under total correctness, it must satisfy p and running S in it can cause an error or one of its final states does not satisfy q.
  - We have $\sigma \models_{tot} \{p\}\ S\ \{q\}$ iff $\sigma \models \neg p$ or $M(S, \sigma) \models q$
  - So $\sigma \not\models_{tot} \{p\}\ S\ \{q\}$ iff $\sigma \models p$ and $M(S, \sigma) \not\models q$

    iff $\sigma \models p$ and ($\perp \in M(S, \sigma)$ or for some $\tau \in M(S, \sigma)$, $\tau \neq \perp$ and $\tau \not\models q$ (i.e., $\tau \models \neg q$, since $\tau \neq \perp$).
- If S is deterministic, then $\sigma \models p$ and $M(S, \sigma) = \{\tau\}$ where $\tau = \perp$ or $\tau \models \neg q$.
- If S is nondeterministic, then $\sigma \models p$ and ($\perp \in M(S, \sigma)$ or for some $\tau \in M(S, \sigma)$, $\tau \models \neg q$).
  - Another characterization: $\sigma \models p$ and if $\perp \notin M(S, \sigma)$, then for some $\tau \in M(S, \sigma)$, $\tau \models \neg q$.

Unsatisfied Partial Correctness

- For a state $\sigma \neq \perp$ to not satisfy $\{p\}\ S\ \{q\}$ under partial correctness, it must satisfy p and running S in it always terminates in a state satisfying $\neg q$. In symbols
  - We have $\sigma \models \{p\}\ S\ \{q\}$ iff $\sigma \models \neg p$ or $M(S, \sigma) - \perp \models q$
  - So $\sigma \not\models \{p\}\ S\ \{q\}$ iff $\sigma \models p$ and $M(S, \sigma) - \perp \not\models q$
    - We know $M(S, \sigma) - \perp \models q$ holds iff for every $\tau \in M(S, \sigma)$, if $\tau \neq \perp$, then $\tau \models q$
    - So $M(S, \sigma) - \perp \models q$ holds iff for some $\tau \in M(S, \sigma)$, we have $\tau \models \neg q$ (since $\tau \not\models q$ with $\tau \neq \perp$)
  - Substituting back, $\sigma \not\models \{p\}\ S\ \{q\}$ iff $\sigma \models p$ and $\tau \models \neg q$ for some $\tau \in M(S, \sigma)$.
- If S is deterministic, then we need $\sigma \models p \wedge M(S, \sigma) = \{\tau\}$ where $\tau \models \neg q$.
- If S is nondeterministic, $M(S, \sigma)$ can include $\perp$ and states that satisfy q, but there must be at least one state in $M(S, \sigma)$ that satisfies $\neg q$.
- ***Note***: If S is nondeterministic and partial correctness of $\{p\}\ S\ \{q\}$ fails under σ, it's possible that some execution paths of S don't terminate or terminate in states satisfying q, but there must be some execution path that ends in a state satisfying $\neg q$.

K. Three Extreme (Mostly Trivial) Cases

- There are three edge cases where partial correctness occurs for uninformative reasons.. First recall the definition of partial correctness: $\sigma \models \{p\}\ S\ \{q\}$ means (if $\sigma \models p$, then $M(S, \sigma) - \perp \models q$).

  - ***p is a contradiction*** (i.e., $\models \neg p$). Since $\sigma \models p$ never holds, the implication (if $\sigma \models p$ then …) always holds, so partial correctness of *{p} S {q}* always holds. So for example, *{F} S {q}* is valid under partial correctness, for all *S* and *q*.
  - ***S always causes an error***. If $M(S, \sigma) = \{\perp\}$ then $M(S, \sigma) - \perp = \varnothing$, and $\varnothing \models q$, so again we get partial correctness of *{p} S {q}*.
  - ***q is a tautology*** (i.e., $\models q$). Then for any σ, $M(S, \sigma) - \perp \models q$, so whether σ satisfies *p* or not, we get partial correctness of *{p} S {q}*. So for example, *{p} S {T}* is valid under partial correctness for all *p* and *S*.
- For total correctness, recall $\sigma \models_{tot} \{p\}\ S\ \{q\}$ means (if $\sigma \models p$, then $M(S, \sigma) \models q$). (Also, recall that since $\perp \not\models q$, if $M(S, \sigma) \models q$, then $\perp \notin M(S, \sigma)$.)
  - ***p is a contradiction***. The argument here is the same as for partial correctness, so for all *S* and *q*, the triple *{F} S {q}* is valid under total correctness.
  - ***S always causes an error***. Since $M(S, \sigma) = \{\perp\}$, we know $M(S, \sigma) \not\models q$. So total correctness of *{p} S {q}* always fails.
  - ***q is a tautology***. $\sigma \models_{tot} \{p\}\ S\ \{T\}$ does says something interesting. Since $M(S, \sigma) \models T$ implies $\perp \notin M(S, \sigma)$, satisfaction of $\sigma \models_{tot} \{p\}\ S\ \{T\}$ requires *S* ***to always terminate*** under σ. So validity of $\models_{tot} \{p\}\ S\ \{T\}$ happens when *S* always terminates when started in a state satisfying *p*.
    - As a general principle, since total correctness is partial correctness plus termination, we have $\sigma \models_{tot} \{p\}\ S\ \{q\}$ iff $\sigma \models \{p\}\ S\ \{q\}$ and $\sigma \models_{tot} \{p\}\ S\ \{T\}$. Again, this means that to show that a triple is totally correct, we can prove partial correctness and termination separately.

