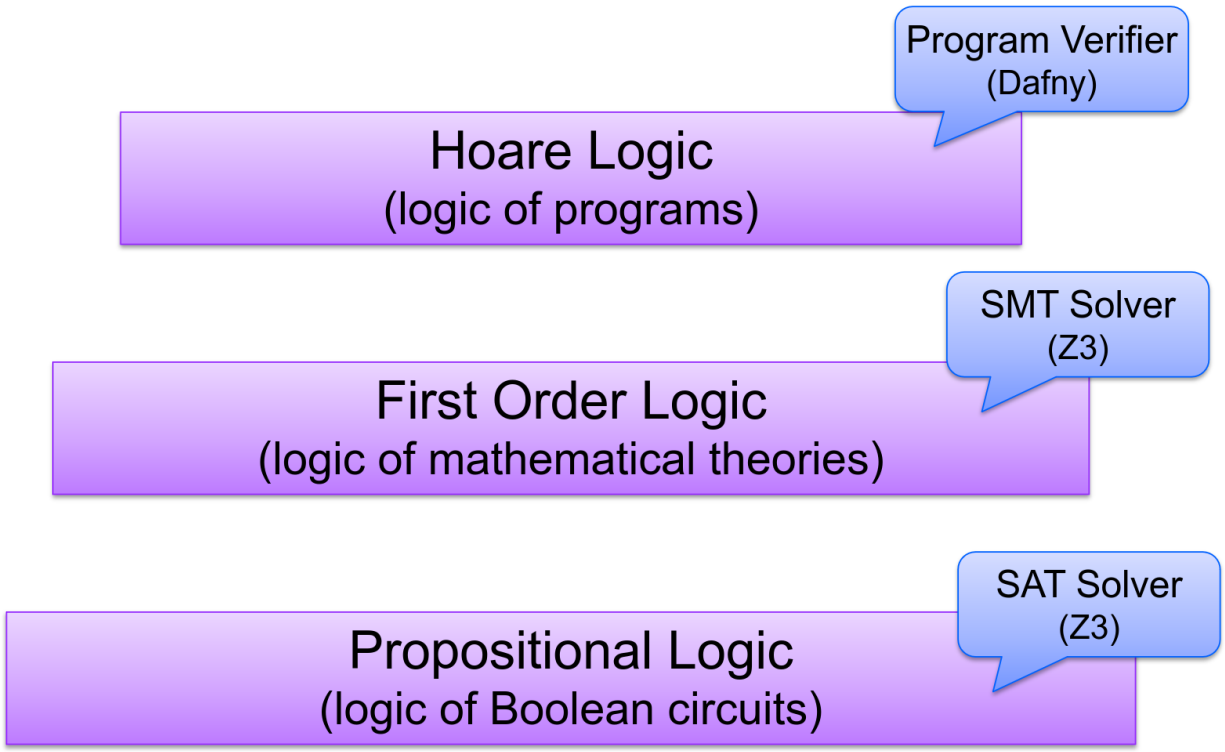
Software Testing, Quality Assurance and Maintenance	Winter 2025
Lecture 12 — February 26, 2025	
Patrick Lam	*version 1*

We've seen this picture on the slides.



This lecture is about Hoare Logic. Again, you've seen this in SE 212, but it's worth a review as we prepare to move onto Dafny. Some of the concepts are subtle!

Axiomatic Semantics. You used axiomatic semantics in SE 212. It consists of:

- a language for stating assertions about programs; and
- rules for establishing the truth of assertions.

You have used axiomatic semantics to make and prove assertions. Some examples:

- this program terminates;
- if this program terminates, the variables `x` and `y` have the same value throughout the execution of the program;
- array accesses are within array bounds.

One can use first-order logic for assertions. There are also special-purpose specification languages, which I'll name-check but not discuss: Z (the OG specification language), Larch, JML. And there

are other logics as well: temporal logic, especially useful for specifying properties of concurrent systems; linear logic, which has been used under-the-hood for Rust uniqueness; and separation logic, for reasoning about the structure of the heap.

Hoare Triples. We write assertions about WHILE programs using Hoare triples:

$$\{A\}\ c\ \{B\}$$

which means that (1) if A holds in state q, and (2) if the semantics says $q \to q'$, then (3) B holds in q'.

One can thus write the valid assertion

$$\{y \leq x\} z := x; z := z + 1 \{y < z\}$$

which means that if $y \leq x$ and you run those two statements, you know that $y < z$ at the end of them.

Partial versus total correctness. More specifically, $\{A\}\ c\ \{B\}$ is a *partial* correctness assertion, and does not imply termination of c.

If A holds in state q, and if there exists q' such that $q \to q'$, then B holds in state q'. So, if there is no q' (i.e. c does not terminate, or gets stuck), then there is, vacuously, no obligation to show B.

There is the notion of *total* correctness as well, though much less often used. $[A]\ c\ [B]$ is the notation for *total* correctness.

If A holds in state q, **then** there exists q' such that $q \to q'$, and B holds in state q'.

More formal Hoare logic

We have assertions like A and B. We'll formalize the language we use for them, define when an assertion holds in a state, and define rules for deriving valid Hoare triples.

Our assertion language. We use *first-order predicate logic* and use WHILE expressions as atoms.

$$\begin{array}{rcl} A & ::== & \mathsf{true} \mid \mathsf{false} \mid e_1 = e_2 \mid e_1 \geq e_2 \\ & \mid & A_1 \wedge A_2 \mid A_1 \vee A_2 \mid A_1 \Rightarrow A_2 \mid \forall x.A \mid \exists x.A \end{array}$$

There are logical variables (e.g. $\forall x$) and program variables. We don't distinguish them. For our purpose, all WHILE variables range over integers.

It is always valid to use a WHILE Boolean expression as an assertion.

Semantics of assertions. We write

$$q \models A$$

to mean that assertion A holds in state q. (It is well-defined when q is defined on all variables occuring free in A).

We define $\models$ inductively on the structure of assertions. (Start with the atomic assertions and arithmetic expressions from WHILE, and then build up the compound assertions like $\wedge$ etc.)

$q \models \mathsf{true}$	always
$q \models e_1 = e_2$	iff $\langle e_1, q\rangle \Downarrow = \langle e_2, q\rangle \Downarrow$
$q \models e_1 \geq e_2$	iff $\langle e_1, q\rangle \Downarrow \geq \langle e_2, q\rangle \Downarrow$
$q \models A_1 \wedge A_2$	iff $q \models A_1$ and $q \models A_2$
$q \models A_1 \vee A_2$	iff $q \models A_1$ or $q \models A_2$
$q \models A_1 \Rightarrow A_2$	iff $q \models A_1$ implies $q \models A_2$
$q \models \forall x.\ A$	iff $\forall n \in \mathbb{Z}.\ q[x := n] \models A$
$q \models \exists x.\ A$	iff $\exists n \in \mathbb{Z}.\ q[x := n] \models A$