Software Testing, Quality Assurance & Maintenance—Lecture 5b

Patrick Lam
University of Waterloo

January 22, 2025

Part I

Structural Induction Example

How Structural Induction Works

Our structures are defined by grammars.

Use the grammar to provide proof obligations for structural induction.

WHILE language: terminals, arithmetic expressions, Boolean expressions, statements.

Goal

Proposition. All WHILE programs that do not contain any `while` statements always terminate.

Terminate = there is a finite derivation that finishes with some final state.

We'll use the big-step semantics to prove this.

WHILE programs

A WHILE program is an `slist`,
which is one or more statements.

We need to prove the proposition for statements.

Statements may contain arithmetic or boolean expressions.

Subgoal

Lemma. Evaluation of any boolean or arithmetic expression always yields a value and terminates.

Proof of Lemma

Start with arithmetic expressions.

Base case: integers n and variables x. From the semantics we have rules:

$$\frac{}{\langle n, q\rangle \Downarrow n} \qquad\qquad \frac{}{\langle x, q\rangle \Downarrow q(x)}$$

which clearly yield values n and $q(x)$ & terminate.

Inductive cases for lemma: binary

Rules: negation, parentheses, arithmetic.

Inductively assume all smaller expressions yield values & terminate.

Let's see $+$.

$$\frac{\langle e_1, q\rangle \Downarrow n_1 \qquad \langle e_2, q\rangle \Downarrow n_2}{\langle e_1 + e_2, q\rangle \Downarrow n_1 + n_2}.$$

By induction, e_1 and e_2 have the property.

This derivation tree shows: if you build an expr wih $+$, it also has the desired property.

Can say “similarly for + and *”.

Inductive cases for lemma: unary

Let's do (e). Assume property holds for e. Then:

$$\frac{\langle e, q\rangle \Downarrow n}{\langle (e), q\rangle \Downarrow n}.$$

Can conclude that (e) also yields a value and terminates.
Unary negation is the same.

Boolean Expressions

For these expressions, base cases are `true` and `false`.

Quote the inference rules

$$\frac{}{\langle \texttt{true}, q \rangle \Downarrow \texttt{true}} \qquad \frac{}{\langle \texttt{false}, q \rangle \Downarrow \texttt{false}}$$

to conclude termination.

Boolean Expressions: Relational Operators

For $<$, $\leq$, etc, we rely on termination for arithmetic expressions.

$$\frac{\langle e_1, q\rangle \Downarrow n_1 \quad \langle e_2, q\rangle \Downarrow n_2}{\langle e_1 < e_2, q\rangle \Downarrow (n_1 < n_2)}.$$

e_1 and e_2 evaluate to integers, and then we apply the rule and return `true` if $n_1 < n_2$ and `false` otherwise.

Boolean Expressions: Boolean Operators

For `and` and `or`, we rely on the induction hypothesis.

$$\frac{\langle e_1, q\rangle \Downarrow b_1 \quad \langle e_2, q\rangle \Downarrow b_2}{\langle b_1 \;\texttt{and}\; b_2, q\rangle \Downarrow (b_1 \wedge b_2)}.$$

By IH, e_1 and e_2 evaluate to b_1 and b_2.
The quoted rule yields a value for $b_1 \;\texttt{and}\; b_2$.

You should also mention `not` and the parenthesized (e) here. $\square$

Back to Statements

The base cases for this proof are `skip`, assignment statements, and the `print_state`, `assert`, `assume`, and `havoc` statements.

We only gave semantics for `skip` and assignment, so we'll not talk about the other statements here either.

Base Case: Assignment Statement

$$\frac{\langle e, q \rangle \Downarrow n}{\langle x := e, q \rangle \Downarrow q[x := n]}.$$

This is a base case: no statements in the hypothesis.

Per lemma, e evals to value n and terminates.

This rule shows that we can evaluate an assignment statement and terminate. (Big-step semantics: evaluation just changes the state.)

Inductive case: if Statement

Termination because we inductively assume termination for then and else clauses.

$$\frac{\langle s_1, q\rangle \Downarrow q' \quad \langle e, q\rangle \Downarrow \text{true}}{\langle \texttt{if}\ e\ \texttt{then}\ s_1\ \texttt{else}\ s_2, q\rangle \Downarrow q'}.$$

s_1 terminates by IH and so the `if` also terminates by this derivation.

"Similiarly for `else`".

Inductive case: statement list

$$\frac{\langle s_1, q\rangle \Downarrow q^1 \quad \cdots \quad \langle s_n, q^{n-1}\rangle \Downarrow q^n}{\langle \{s_1; \cdot; ss_n\}, q\rangle \Downarrow q^n}.$$

Because, inductively, all of the s_i terminate, then so does the list of s. $\square$