Loop invariants

László Babai

Last updated 1-31-2009

Given an algorithm, a configuration is an assignment of values to each variable. A configuration is feasible if it can actually occur during the execution of the algorithm. Let \( C \) denote the set of all configurations, whether feasible or not. We refer to \( C \) as the configuration space.

A predicate over a set \( A \) is a Boolean function \( f: A \rightarrow \{0, 1\} \) (1: “true,” 0: “false”). A transformation of \( A \) is a function \( g: A \rightarrow A \).

Let \( P \) and \( Q \) be predicates over the configuration space \( C \) and let \( S \) be a set of instructions, viewed as a transformation of \( C \). Consider the loop “while \( P \) do \( S \).” We say that \( Q \) is a loop-invariant for this loop if for all configurations \( X \in C \) the following implication is correct:

\[
P(X) \land Q(X) \Rightarrow Q(S(X)).
\]

(1)

In other words, if \( P \land Q \) holds for the configuration \( X \) then \( Q \) also holds for the configuration \( S(X) \), where \( S(X) \) is the configuration obtained from \( X \) by executing \( S \).

Note that the highlighted statement has to hold even for infeasible configurations. This is analogous to chess puzzles: when showing that a certain configuration leads to checkmate in two moves, you do not investigate whether or not the given configuration could arise in an actual game.

Exercise 1. Prove: if \( Q_1 \) and \( Q_2 \) are loop-invariants for the loop “while \( P \) do \( S \)” then \( Q_1 \land Q_2 \) is also a loop-invariant.

A configuration for Dijkstra’s algorithm consists of a status value (white, grey, black), a cost value (a real number or \( \infty \)), and a parent link (possibly NIL) for each vertex, and a set \( Q \) (the priority queue; here we treat it as a set of nodes; priority is based on the cost value).

Dijkstra’s algorithm consists of iterations of a single “while” loop. Consider the following statements:

\[
Q_1 : \quad (\forall u \in V)(u \in Q \text{ if and only if } u \text{ is grey})
\]

\[
Q_2 : \quad (\forall u \in V)(\text{if } u \text{ is white then } c(u) = \infty).
\]

\[
Q_2^* : \quad (\forall u \in V)(u \text{ is white if and only if } c(u) = \infty).
\]
\[ Q_3 : (\forall u, v \in V)(\text{if } u \text{ is black and } v \text{ is not black then } c(u) \leq c(v)). \]

\[ Q_4 : (\forall v \in V)(c(v) \text{ is the minimum cost among all } s \to \ldots \to v \text{ paths that pass through black vertices only}). \]

(We say that the path \( s = v_0 \to v_1 \to \ldots \to v_k = v \) passes through black vertices only if for \( 0 \leq i \leq k - 1 \), the vertex \( v_i \) is black.)

**Exercise 2.**

(a) Prove that \( Q_1 \) and \( Q_2 \) are loop-invariants.

(b) Let \( R = Q_1 \& Q_2 \). Prove that \( R \& Q_3 \) is a loop-invariant.

(c) Prove that \( R \& Q_3 \& Q_4 \) is a loop-invariant.

(d) Prove that \( R \& Q_4 \) alone is not a loop-invariant. **Explanation.** You need to construct a weighted directed graph with nonnegative weights, a source, and assignments of all the variables (parent pointers, status colors, current cost values) such that \( R \& Q_4 \) holds for your configuration, but \( Q_4 \) will no longer hold after executing Dijkstra’s **while** loop. Your graph should have very few vertices.

**Exercise 3.** Infer from Exercise 2(c) that Dijkstra’s algorithm is correct.

**Exercise 4.** (a) Prove that \( Q_2^* \) is not a loop-invariant.

(b) Prove that \( Q_1 \& Q_2^* \) is a loop invariant.