Journal of Symbolic Computation 48 (2013) 1–36

Contents lists available at SciVerse ScienceDirect

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

Invariant functions and invariant relations: An alternative to invariant assertions

Lamia Labed Jilani [a], Olfa Mraihi [b], Asma Louhichi [c], Wided Ghardallou [c], Khaled Bsaies [c], Ali Mili [d]

[a] *Institut Superieur de Gestion, RIADI Lab, Bardo, Tunisia*
[b] *Institut Superieur de Gestion, Bardo, Tunisia*
[c] *Faculte des Sciences de Tunis, LIPAH Lab, El Manar, Tunisia*
[d] *New Jersey Institute of Technology, Newark, NJ, United States*

ARTICLE INFO

Article history:
Received 24 May 2011
Accepted 9 April 2012
Available online 4 May 2012

Keywords:
Invariant assertions
Invariant relations
Invariant functions
While loops
Reasoning about loops
Invariant generation
Loop functions

ABSTRACT

Whereas the analysis of loops in imperative programs is, justifiably, dominated by the concept of invariant assertion, we submit a related but different concept, of invariant relation, and show how it can be used to analyze diverse aspects of a while loop. We also introduce the concept of invariant function, which is used to generate a broad class of invariant relations.

1. Invariant assertions, invariant relations, and invariant functions

Despite several decades of research, the development and maintenance of software products under acceptable standards of quality and reliability remain an unfulfilled challenge, so much so that the global research community is rallying around an international initiative to focus on this elusive goal (Hoare et al., 2009; Woodcock and Banach, 2007). Known as *The Verified Software Initiative*, this enterprise aims to achieve three goals, namely the development of theoretical foundations, the generation of industrial strength tools, and the production of verified software applications in selected application domains.

E-mail addresses: lamia.labed@isg.rnu.tn (L.L. Jilani), olfa.mraihi@yahoo.fr (O. Mraihi), louhichiasma@yahoo.fr (A. Louhichi), wided.ghardallou@gmail.com (W. Ghardallou), khaled.bsaies@fst.rnu.tn (K. Bsaies), mili@cis.njit.edu (A. Mili).

0747-7171/$ – see front matter
http://dx.doi.org/10.1016/j.jsc.2012.04.001

Despite the proliferation of programming languages and programming paradigms, the vast majority of software being developed and maintained nowadays is written in C-like imperative languages, where loops constitute the main locus of complexity, hence the main source of program faults. Object oriented languages shift the focus towards data organization and interprocess coordination, but loops play an important role there as well. The analysis of while loop has, justifiably, been dominated by the concept of invariant assertions, since its introduction by C.A.R. Hoare in 1969 (Hoare, 1969). This concept has influenced much of the research on program analysis and design since its introduction (Dijkstra, 1976; Gries, 1981), and continues to do so to this day; the generation of invariant assertions has remained a topic of great interest during the seventies and early eighties (Cousot and Cousot, 1977; Cheatham and Townley, 1976; Cousot and Halbwachs, 1978), then has emerged again in the last decade (Carbonnell and Kapur, 2004; Colon et al., 2003; Denney and Fischer, 2006; Ernst et al., 2007; Fahringer and Scholz, 2003; Fu et al., 2008; Jebelean and Giese, 2007; Hu et al., 2004; Kovacs and Jebelean, 2004, 2005; Podelski and Rybalchenko, 2004; Sankaranarayana et al., 2004; Hoder et al., 2010; Kovacs and Voronkov, 2009a; Maclean et al., 2010; Zuleger and Sinn, 2010; Kroening et al., 2010; Iosif et al., 2010; Furia and Meyer, 2010).

In this paper we present two distinct but related concepts, namely invariant relations and invariant functions, and show how these can provide complementary insights to those that invariant assertions afford us. Specifically, we find that invariant functions can be used to generate invariant relations, and that invariant relations can be used to elucidate a broad range of functional properties of loops, including:

- Computing or approximating the function of a loop (Mili et al., 2009).
- Computing or approximating the termination condition of a loop (i.e. the condition under which the loop is guaranteed to terminate).
- Computing an invariant assertion of the loop for a given precondition (Louhichi et al., 2011).
- Computing the weakest precondition of a loop for a given postcondition (Mraihi et al., 2011a).
- Computing or approximating the strongest postcondition of a loop for given precondition (Mraihi et al., 2011a).
- Computing conditions of correctness of the loop with respect to relational specifications (Ghardallou et al., 2011; Parnas, 2011).

In this paper, we focus on the use of invariant relations to compute loop functions and termination conditions of loops; also, to compare our approach to approaches that are based on invariant assertions, we use invariant relations to generate invariant assertions. As for the relationships between invariant relations, invariant functions, and invariant assertions, they are beyond the scope of this paper, and are discussed in some detail in Mraihi et al. (2011b).

To give the reader some intuition for these three concepts, we present them below by means of a simple example, namely the following loop on natural variables n, f, k, such that k is non-zero.

```
{k:=1; f:=1; while (k!=n+1) {f:=f*k; k:=k+1;}}.
```

We propose the following characterizations/illustrations.

- *Invariant Assertion*: An invariant assertion is a predicate on program variables that holds after any number (including zero) of iterations. For the program above, we propose

$$f = (k-1)!.$$

- *Invariant Function*: An invariant function is a function that takes the same value before and after application of the loop body (assuming the loop condition holds). For the above program, we propose

$$V(n, f, k) = \frac{f}{(k-1)!}.$$