Information Processing Letters 115 (2015) 468–474

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Automatic proving or disproving equality loop invariants based on finite difference techniques

Mengjun Li

School of Computer Science, National University of Defense Technology, Changsha, China

ARTICLE INFO

Article history:
Received 22 July 2014
Received in revised form 6 October 2014
Accepted 17 November 2014
Available online 3 December 2014
Communicated by J.L. Fiadeiro

Keywords:
Automatic theorem proving
Equality loop invariants
Formal characterization
Formal verification
Finite difference techniques

ABSTRACT

Loop invariants play a major role in software verification. In this paper, based on finite difference techniques, a formal characterization for equality loop invariants is presented. Integrating the formal characterization with the automatic verification approach in [5], the algorithm for automatic proving or disproving equality loop invariants is presented. The effectiveness of the algorithm is demonstrated with the experimental results.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

A key problem in automatic software verification is the inference of loop invariants. Recently, dynamic techniques have been applied to discover invariants rapidly. The Daikon approach of Ernst et al. [2] showed that dynamic inference is practical. In [6], the mathematical techniques equation solving, polyhedra construction and SMT solving are combined to detect invariants dynamically. In [11], a guess-and-check algorithm for computing algebraic equation invariants of the form $\wedge_i f_i(x_1, \ldots, x_n) = 0$ is presented, where each f_i is a polynomial over the variables $x_1, \ldots, x_n$ of the program.

The dynamic approach consists in testing a large number of candidate properties against several program runs, the properties that are not violated in any of the executions are retained as "likely" invariants [2]. Since the likely invariant may not be a real invariant, its validity needs to be proved. In [2,6], proving the validity of the discovered likely loop invariants is not considered. In [11], the validity of the discovered likely invariants is verified with the theorem prover Z3.

In [5], a practical approach for discovering likely equality loop invariants using parameterized templates, random testing and Gaussian elimination is presented, and the discovered likely loop invariants are proved with finite difference techniques. In this paper, also based on finite difference techniques, a formal characterization of equality loop invariants is presented. Integrating the formal characterization and the automatic verification approach in [5], an algorithm for straightly proving or disproving the likely equality loop invariants is presented, and the experimental results are shown also. The presented algorithm has the following features:

(1) With theorem provers, additional support invariants need to be provided in general to prove a loop invariant inductive. The presented algorithm is based on finite difference techniques, support invariants need not to be provided: when a proof or a disproof fails, it only requires to increase the threshold of the height of the decidable difference tree.

(2) Since the presented algorithm is not based on Gröbner bases like [1], besides the polynomial equality loop invariants, equality loop invariants of other forms can also be

E-mail address: mengjunli1975@gmail.com.

http://dx.doi.org/10.1016/j.ipl.2014.11.006
0020-0190/© 2014 Elsevier B.V. All rights reserved.

$\overline{x} := \overline{x_0}$;
while b **do**
  if b_1 **then** $\overline{x} := (f_1^1(\overline{x}), \cdots, f_n^1(\overline{x}))$;
  $\vdots$
  if b_m **then** $\overline{x} := (f_1^m(\overline{x}), \cdots, f_n^m(\overline{x}))$;
od
	$\overline{x} := \overline{x_0}$;
	while $true$ **do**
	  $\tau_1 : \overline{x} := (f_1^1(\overline{x}), \cdots, f_n^1(\overline{x}))$;
	  $\vdots$
	  $\tau_m : \overline{x} := (f_1^m(\overline{x}), \cdots, f_n^m(\overline{x}))$;
	od

(a) The while loop program with conditional branches	(b) The multi-path program

Fig. 1. The while loop program with conditional branches and its multi-path program abstraction.

$(x, r, s) := (a, 1, \frac{13}{4})$
while $(x - s > 0)$ **do**
  $(x, s, r) := (x - s, s + 6 * r + 3, r + 1)$
od
	$(x, r, s) := (a, 1, \frac{13}{4})$
	while $true$ **do**
	  $\tau : (x, s, r) := (x - s, s + 6 * r + 3, r + 1)$
	od

(a) the cubic-root algorithm program	(b) the multipath program abstraction

Fig. 2. The cubic-root algorithm program and its multi-path program abstraction.

proved or disproved, such as the loop invariant $f - n! = 0$ of the program factorial.c in [5]. In Section 5, it is pointed out that the approaches in [10] and [4] are not fit for inferring loop invariants of the Babylonian algorithm program in [3]. The Babylonian algorithm program has a loop invariant $result * y = a$, which can be discovered by the approach in [5] and proved by the presented algorithm.

The rest of this paper is structured as follows: Section 2 presents some preliminary definitions, Section 3 presents a formal characterization of equality loop invariants, Section 4 presents the algorithm for automatic proving or disproving the likely equality loop invariants, the experimental results are shown in Section 5, Section 6 discusses the related work, and Section 7 concludes this paper.

2. Preliminaries

Definition 1 *(Assignment transition).* An assignment transition τ is a sequent algebraic assignment of the form $\bar{x} := (f_1(\bar{x}), \ldots, f_n(\bar{x}))$, where $\bar{x} = (x_1, \ldots, x_n)$, and $f_1(\bar{x}), \ldots, f_n(\bar{x})$ are n arithmetic expressions over $x_1, \ldots, x_n$.

Definition 2 *(Multi-path program).* For variables $\bar{x} = (x_1, \ldots, x_n)$, a multi-path program with m paths has the form shown in Fig. 1(b), where $\overline{x_0}$ expresses the initial value of $\bar{x}$, and $\tau_1, \ldots, \tau_m$ are assignment transitions.

The tests in the loop and in the conditional branches are ignored, the multi-path program is the abstraction of the while loop with conditional branches shown in Fig. 1(a). The program model in [10] and [4] is also the multi-path program.

Definition 3 *(Loop invariant).* For a multi-path program, a loop invariant is an equality $E(\bar{x}) = 0$ that satisfies $E(w(\overline{x_0})) = 0$ for each word w over the alphabet $\Sigma = \{\tau_1, \ldots, \tau_m\}$, where $w(\overline{x_0})$ is defined inductively as follows:

(1) $\varepsilon(\overline{x_0}) = \overline{x_0}$.
(2) $\tau \cdot \sigma(\overline{x_0}) = (f_1(\sigma(\overline{x_0})), \ldots, f_n(\sigma(\overline{x_0})))$,

where ε denotes the empty word, and σ is a word over the alphabet Σ, and τ is an assignment transition $\bar{x} := (f_1(\bar{x}), \ldots, f_n(\bar{x}))$. And $E(\overline{x_0}) = 0$ is called the initial condition of loop invariant.

It is easy to prove that if $E(\bar{x}) = 0$ is a loop invariant of the multi-path program in Fig. 1(b), then it is also a loop invariant defined in the Hoare calculus of the while loop in Fig. 1(a).

Definition 4 *(Difference).* The difference of an expression $E(\bar{x})$ over an assignment transition τ is the expression $E(\tau(\bar{x})) - E(\bar{x})$, denoted by $\Delta_\tau E(\bar{x})$.

For example, for the loop program in Fig. 2(b), $\Delta_\tau(\frac{1}{4} + 3r^2 - s) = \frac{1}{4} + 3(r+1)^2 - (s + 6r + 3) - (\frac{1}{4} + 3r^2 - s) = 0$.

3. Formal characterization of equality loop invariants

The following theorem gives a formal characterization of equality loop invariants.

Theorem 1. *$E(\bar{x}) = 0$ is a loop invariant of a multi-path program if and only if, $E(\overline{x_0}) = 0$ and for each transition τ_i ($1 \le i \le m$), $\Delta_{\tau_i} E(\bar{x}) = 0$ is also a loop invariant.*

Proof. We prove the necessary part first. Assume that $E(\bar{x}) = 0$ is a loop invariant, then $E(\overline{x_0}) = 0$ by the definition. And we prove that for each transition τ_i ($1 \le i \le m$), $\Delta_{\tau_i} E(\bar{x}) = 0$ is also a loop invariant. For each word w over the alphabet $\Sigma = \{\tau_1, \ldots, \tau_m\}$, $\Delta_{\tau_i} E(w(\overline{x_0})) = E(\tau_i \cdot w(\overline{x_0})) - E(w(\overline{x_0}))$. Since $E(\bar{x}) = 0$ is a loop invariant, then $\Delta_{\tau_i} E(w(\bar{x_0})) = 0 - 0 = 0$. So $\Delta_{\tau_i} E(\bar{x}) = 0$ is also a loop invariant of the multi-path program.

Now we prove the sufficient part. For each word w over the alphabet $\Sigma = \{\tau_1, \ldots, \tau_m\}$, when $w = \varepsilon$, by assumption $E(\overline{x_0}) = 0$, so $E(w(\overline{x_0})) = E(\varepsilon(\overline{x_0})) = 0$. When $w = \tau_i \cdot \sigma$, assuming that $E(\sigma(\overline{x_0})) = 0$, $E(\tau_i \cdot \sigma(\overline{x_0})) = E(\sigma(\overline{x_0})) + \Delta_{\tau_i} E(\sigma(\overline{x_0}))$. Since $\Delta_{\tau_i} E(\bar{x}) = 0$ is a loop invariant, so $\Delta_{\tau_i} E(\sigma(\overline{x_0})) = 0$ and $E(\tau_i \cdot \sigma(\overline{x_0})) = 0 + 0 = 0$. Thus $E(w(\overline{x_0})) = 0$ for each word w over the alphabet Σ, so $E(\bar{x}) = 0$ is a loop invariant of the multi-path program. □

Download English Version:

https://daneshyari.com/en/article/427253

Download Persian Version:

https://daneshyari.com/article/427253

Daneshyari.com