Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Generalized homogeneous polynomials for efficient template-based nonlinear invariant synthesis

Kensuke Kojima *, Minoru Kinoshita, Kohei Suenaga

Department of Communications and Computer Engineering, Graduate School of Informatics, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan

ARTICLE INFO

Article history:
Received 19 May 2017
Received in revised form 5 June 2018
Accepted 5 June 2018
Available online xxxx
Communicated by D. Sannella

Keywords:
Static analysis
Polynomial invariants
Homogeneous polynomial

ABSTRACT

The *template-based* method is one of the most successful approaches to algebraic invariant synthesis. In this method, an algorithm designates a *template polynomial* p over program variables, generates constraints for $p = 0$ to be an invariant, and solves the generated constraints. However, this approach often suffers from an increasing template size if the degree of a template polynomial is too high.

We propose a technique to make template-based methods more efficient. Our technique is based on the following finding: If $p = 0$ is an algebraic invariant, then p can be decomposed into the sum of specific polynomials that we call *generalized homogeneous* polynomials, that are often smaller. This finding justifies using only a smaller template that corresponds to generalized homogeneous polynomials.

Concretely, we state and prove our finding above formally. Then, we modify the template-based algorithm proposed by Cachera et al. so that it generates only generalized homogeneous polynomials. This modification is proved to be sound. Furthermore, we also empirically demonstrate the merit of the restriction to generalized homogeneous polynomials. Our implementation outperforms that of Cachera et al. for programs that require a higher-degree template.

1. Introduction

We consider the following *postcondition problem*: Given a program c, discover a fact that holds at the end of c regardless of the initial state. This article focuses on a postcondition written as an *algebraic condition* $p_1 = 0 \wedge \cdots \wedge p_n = 0$, where $p_1, \ldots, p_n$ are polynomials over program variables; this problem is a basis for static verification of functional correctness.

One approach to this problem is *invariant synthesis*, in which we are to compute a family of predicates P_l indexed by program locations l such that P_l holds whenever the execution of c reaches l. The invariant associated with the end of c is a solution to the postcondition problem.

Because of its importance in static program verification, algebraic invariant synthesis has been intensively studied [1–4]. Among these proposed techniques, one successful approach is the constraint-based method in which invariant synthesis is reduced to a constraint-solving problem. During constraint generation, this method designates *templates*, which are poly-

* Corresponding author.
E-mail address: kozima@fos.kuis.kyoto-u.ac.jp (K. Kojima).

https://doi.org/10.1016/j.tcs.2018.06.005
0304-3975/

Please cite this article in press as: K. Kojima et al., Generalized homogeneous polynomials for efficient template-based nonlinear invariant synthesis, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.06.005

```
1: x := x_0; v := v_0; t := t_0;
2: while t − t_1 ≠ 0 do
3:     (x, v, t) := (x + v dt, v − g dt, t + dt);
4: end while
5:
```

Fig. 1. Program c_{fall}, which models a falling mass point. The symbols in the program represent the following quantities: x is the position of the point, v is its speed, t is time, x_0 is the initial position, v_0 is the initial speed, t_0 is the initial value of the clock t, g is the acceleration rate, t_1 is the time limit, and dt is the discretization interval. The simultaneous substitution in the loop body numerically updates the values of x, v, and t. The values of x, v, and t are numerical solutions of the differential equations $\frac{dx}{dt} = v$ and $\frac{dv}{dt} = -g$.

nomials over the program variables with unknown parameters at the coefficient positions [1]. The algorithm generates constraints that ensure that the templates are invariants, and obtains the invariants by solving the constraints.[1]

Example 1. The program c_{fall} in Fig. 1 models the behavior of a mass point with weight 1 and with a constant acceleration rate.[2] For this program, the postcondition $-gt + gt_0 - v + v_0 = 0$ holds regardless of the initial state.

We describe how a template-based method computes the postcondition in Example 1. The method described here differs from the one we explore in this article; this explanation is intended to suggest the flavor of a template method.

A template-based method generates a *template polynomial* over the program variables that represent a postcondition. For example, the template polynomial of degree 2 is

$$p(x_0, v_0, t_0, x, v, t, t_1, dt, g) := a_1 + a_{x_0}x_0 + a_{v_0}v_0 + \cdots + a_{dtg}dtg,$$

where $a_1, a_{x_0}, a_{v_0}, \ldots, a_{dtg}$ are unknown parameters representing coefficients of $1, x_0, v_0, \ldots, dtg$, respectively. The procedure then generates constraints under which $p(x_0, v_0, \ldots, g) = 0$ is indeed a postcondition of c_{fall}. The method proposed by Sankaranarayanan et al. [1] that uses Gröbner bases [5] in invariant synthesis generates the constraints as equations over the parameters; in this case, a solution to the constraints gives $-gt + gt_0 - v + v_0 = 0$, which indeed holds at the end of c_{fall}.

One of the drawbacks of the template-based method is excessive growth of the size of a template. Blindly generating a template of degree d for a degree parameter d makes the algorithm less scalable for higher-degree postconditions. For example, the program in Example 1 has a postcondition $-gt^2 + gt_0^2 - 2tv + 2t_0v_0 + 2x - 2x_0 - tgdt + t_0gdt = 0$ at Line 4. Template-based invariant synthesis procedures proposed so far, which typically increase the degree d from smaller ones iteratively, eventually discover this invariant when they try $d = 3$. However, a degree-3 template contains as many as $\binom{9+3}{3} = 220$ monomials. Computation with such large templates is often prohibitively expensive.

We propose a hack to alleviate this drawback in the template-based methods. Our method is inspired by a rule of thumb in physics called the *principle of quantity dimension*: A physical law should not add two quantities with different *quantity dimensions* [6]. If we accept this principle, then, at least for a physically meaningful program such as c_{fall}, relevant relations among variables that hold at the end of the program (and therefore a template) should consist of monomials with the same quantity dimensions.

Indeed, the polynomial $-gt + gt_0 - v + v_0$ calculated in Example 1 consists only of quantities that represent velocities. The polynomial $-gt^2 + gt_0^2 - 2tv + 2t_0v_0 + 2x - 2x_0 - tgdt + t_0gdt$ above consists only of quantities corresponding to the length. If we use L and T to represent quantity dimensions for lengths and times (the notation in physics), the first and second polynomials consist only of monomials with the quantity dimension LT^{-1} and L, respectively.

By leveraging the quantity dimension principle in the template synthesis phase, we can reduce the size of a template. For example, we could use a template that consists only of monomials for, say, velocity quantities

$$a_{v_0}v_0 + a_v v + a_{tg}tg + a_{dtg}dtg + a_{t_1g}t_1g$$

instead of the general degree-2 polynomial, which consists of 55 monomials.

The idea of the quantity dimension principle can be nicely captured by generalizing the notion of *homogeneous polynomials*. A polynomial is said to be *homogeneous* if it consists of monomials of the same degree; for example, the polynomial $x^3 + x^2y + xy^2 + y^3$ is a homogeneous polynomial of degree 3. We generalize this notion of homogeneity so that a *degree* is an expression corresponding to a quantity dimension (e.g., LT^{-1}).

Let us describe our idea using an example, deferring formal definitions. Suppose we have the following *degree assignment* for each program variable:

[1] The constraint-based method by Cachera et al. [4], which is the basis of the current article, uses a template also for other purposes. See Section 5 for details.

[2] Although the guard condition $t - t_1 \neq 0$ should be $t - t_1 < 0$ in a real-world numerical program, we use the current example for presentation purposes.

Please cite this article in press as: K. Kojima et al., Generalized homogeneous polynomials for efficient template-based nonlinear invariant synthesis, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.06.005