Abstract. In this paper, we present methods for checking and inferring all valid polynomial relations in \( \mathbb{Z}_{2^w} \). In contrast to the infinite field \( \mathbb{Q} \), \( \mathbb{Z}_{2^w} \) is finite and hence allows for finitely many polynomial functions only. In this paper we show, that checking the validity of a polynomial invariant over \( \mathbb{Z}_{2^w} \) is, though decidable, only \( \text{PSPACE} \)-complete. Apart from the impracticable algorithm for the theoretical upper bound, we present a feasible algorithm for verifying polynomial invariants over \( \mathbb{Z}_{2^w} \) which runs in polynomial time if the number of program variables is bounded by a constant. In this case, we also obtain a polynomial-time algorithm for inferring all polynomial relations. In general, our approach provides us with a feasible algorithm to infer all polynomial invariants up to a low degree.

1 Introduction

In reasoning about termination of programs, the crucial aspect is the knowledge about program invariants. Therefore, it is not surprising that the field of checking and finding of program invariants has been quite active, recently.

Many analyses interpret the values of variables regarding the field \( \mathbb{Q} \). Modern computer architectures, on the other hand, provide arithmetic operations modulo suitable powers of 2. It is well-known that there are equalities valid modulo \( 2^w \), which do not hold in general. The polynomial \( 2^{31}x(x + 1) \), for example, constantly evaluates to 0 modulo \( 2^{32} \) but may show non-zero values over \( \mathbb{Q} \). Accordingly, an analysis based on \( \mathbb{Q} \) will systematically miss a whole class of potential program invariants.

Example 1. As an example, consider the program from figure 1. This program repeatedly increases the value of program variable \( x \) in line 4 if arithmetic is modulo \( 2^{32} \). Therefore, the program \texttt{powersum()} computes a square sum. Thus, at program line 6 the polynomial invariant \( 2 \cdot x^3 + 3 \cdot x^2 + x - 6 \cdot y = 0 \) holds modulo \( 2^{32} \) — but not over the field \( \mathbb{Q} \).

© Springer-Verlag Berlin Heidelberg 2008
An exact analysis of the example program should take into account the structure of polynomials over the domain $\mathbb{Z}_{2^{32}}$: The right hand side in the assignment in line 4 of the example can be rewritten as $2^{31}x(x+1) + x + 1$ where the first summand $2^{31}x(x+1)$ is equivalent to the zero polynomial over $\mathbb{Z}_{2^{32}}$. Such polynomials are called vanishing. 

Singmaster [17] investigates the special structure of univariate vanishing polynomials over $\mathbb{Z}_m$ and provides necessary and sufficient conditions for a polynomial to vanish over $\mathbb{Z}_m$. Hungerbühler and Specker extend this result to multivariate polynomials and introduce a canonical form for polynomials in quotient rings [3]. Shekhar et al. present an algorithm to compute this canonical representation over the quotient ring $\mathbb{Z}_{2^w}$ [16]. A minimal Gröbner base characterising all vanishing polynomials in arbitrary quotient rings is given by Wienand in [18]. In contrast to the infinite field $\mathbb{Q}$, the ring $\mathbb{Z}_{2^w}$ is finite. Therefore, there are just finitely many distinct $k$-ary polynomial functions. In fact, it will turn out that we can restrict ourselves to polynomials in $k$ variables up to a total degree $1.5(w+k)$. Due to this upper bound on the total degrees of the polynomials of interest, the problem of checking or inferring of polynomials over $\mathbb{Z}_{2^w}$ becomes an analysis problem over finite domains only and therefore trivially is computable. Hence, the key issue is to provide tight upper complexity bounds as well as algorithms which also show decent behaviour on practical examples.

In this paper, we first consider the problem of checking whether a given polynomial relation is valid at a given program point. While being decidable over $\mathbb{Q}$, we show that this problem becomes $\text{PSPACE}$-complete over $\mathbb{Z}_{2^w}$. Furthermore, we present a practical algorithm for this problem which is based on effective precise weakest precondition computation. In case that the number of variables is bounded by a (small) constant, this algorithm even runs in polynomial time.

Secondly, we consider the problem of inferring all polynomial relations which are valid at a given program point. This problem, though not known to be computable in $\mathbb{Q}$, turns out to be computable in exponential time over $\mathbb{Z}_{2^w}$. Again, we present an algorithm for inferring all polynomial invariants of a given shape, whose runtime turns out to be polynomial given that the number of variables is bounded by a constant. Both algorithms have been implemented, and we report on preliminary experiments.

Related Work

The pioneer in the area of finding polynomial relations was Karr [4] who inferred the validity of polynomial relations of degree at most 1 (i.e., affine relations) over programs using affine assignments and tests only. An algorithm for checking validity of polynomial relations over programs using polynomial assignments is provided by Müller-Olm and Seidl [7] and was extended later to deal with disequality guards as well [9]. Their approach is based on effective weakest precondition computations where conjunctions of polynomial relations are described by polynomial ideals. Termination of a fixpoint computation in $\mathbb{Q}$ thus is guaranteed by Hilbert’s base theorem. In [9], the authors also observe that their method for checking the validity of polynomial relations can be used to construct an algorithm for inferring all polynomial invariants up to a fixed degree. In [13], Rodriguez-Carbonell et al. pick up the idea of describing invariants by polynomial ideals and propose a forward propagating analysis, based on a constraint system over these ideals. As infinite descending chains of polynomial