On Computing Solutions of Linear Diophantine Equations with One Non-linear Parameter

Stefan Schuster and Armin Größlinger

Abstract. We present an algorithm for solving Diophantine equations which are linear in the variables, but non-linear in one parameter. This enables us to compute data dependences in more general situations than is possible with current algorithms.

1. Introduction

Finding data dependences, i.e., determining which operations of a program access the same memory cells, is the basis for automatic parallelisation and cache optimisation. For example, in the code shown in Figure 1(a), we can ask which loop iterations (i, j) and (i', j') access the same elements of the array A. In algorithms proposed and implemented so far, this analysis is restricted to the case that loop bounds and array index expressions are terms which are linear in the variables (i, j in the example) and the structure parameters (m, n in the example). Syntactic treatment of non-linearities has been suggested [PW95], but our aim is to lift the restriction to linear parameters algebraically. Banerjee's data dependence analysis [Ban93] consists of two steps. First, it handles equations arising from the array accesses in the program. Second, it takes inequalities in the loop bounds and the execution order into account. As noted by Banerjee, handling the Diophantine equations exactly in the integers is most important; the inequalities can be handled in the reals without losing much accuracy in practise.

The algorithm we present solves Diophantine equations which are linear in the variables, but non-linear in one parameter. This enables us to generalise Banerjee's dependence analysis to computing the data dependences in presence of one non-linear parameter. Some special cases with multiple non-linear paramters, most importantly proving the absence of dependences, can also be handled. The inequalities are handled in the reals using existing techniques [GGL06].

2. Solving Equations

Given $x\,A = b$ with $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^n$, Banerjee's algorithm solves for $x \in \mathbb{Z}^m$ in a two-step procedure. First, $U \in \mathbb{Z}^{m\times m}$ and $S \in \mathbb{Z}^{m\times n}$ are computed such

that U is unimodular, S is an echelon matrix and $UA = S$ ("extended GCD"). Second, $t\,S = b$ is solved for $t \in \mathbb{Z}^m$ ("backward substitution"). The solutions for x are then given by $x = t\,U$. Generalising both steps to handling a non-linear parameter p, i.e., to the case in which the coefficients of the variables can be arbitrary polynomials in p, yields our main result:

Theorem 2.1. *The solutions of $x\,A = b$ with $A \in \mathbb{Z}[p]^{m\times n}$, $b \in \mathbb{Z}[p]^m$ and the parameter $p \in \mathbb{Z}$ can be represented by a finite case distinction on p which can be computed algorithmically.*

To this end, we introduce the ring $\mathcal{AIQ} \supseteq \mathbb{Z}[p]$ and show that, in Theorem 2.1, $\mathbb{Z}[p]$ can be replaced by $\mathcal{AIQ}$. A similar therorem cannot hold for the case with more than one parameter, since the number of steps needed to compute $\gcd(p_1, p_2)$ is unbounded [vdD03]. We proceed by showing the generalisation of the two phases.

2.1. Extended GCD Computation

The GCDs we must compute are mappings $d : p \mapsto \sum_{i=0}^{u} c_i(p)\lfloor \frac{p}{l} \rfloor^i$, which we call $\mathcal{AIQ}$ *polynomials*, where the $c_i : \mathbb{Z} \to \mathbb{Z}$ are periodic functions with some common period l. The set of all $\mathcal{AIQ}$ polynomials forms a ring and is exactly the set of quasi-polynomials f with $f(\mathbb{Z}) \subseteq \mathbb{Z}$.

Theorem 2.2. *Let $f, g \in \mathbb{Z}[p]$. Then the mapping $p \mapsto \gcd\big(f(p), g(p)\big)$ is an $\mathcal{AIQ}$ polynomial (called the $\mathcal{AIQ}$ GCD of f and g).*

In the non-parametric case of $A \in \mathbb{Z}^{2\times 1}$, the matrices S and U are essentially constructed by exploiting the fact that $\gcd(a, b) = \gcd(b, a - kb)$ with of $k = \lfloor \frac{a}{b} \rfloor$. This yields a sequence $A = (a, b)^T = (a_0, b_0)^T \to (a_1, b_1)^T \to \cdots \to (a_n, 0)^T = S$ of column vectors (and unimodular matrices U_i with $(a_{i+1}, b_{i+1})^T = U_i\,(a_i, b_i)^T$ such that $U = U_{n-1} \cdot \ldots \cdot U_0$). Let us call this the GCD sequence of (a, b). To generalise this procedure to integral polynomials $f, g \in \mathbb{Z}[p]$, we have to handle two different cases (assuming w.l.o.g. that $\deg(f) \geq \deg(g)$):

(1) If $\deg(f) = \deg(g)$, we can "lift" the GCD sequence $\big(\mathrm{HC}(f), \mathrm{HC}(g)\big) \to \big(\mathrm{HC}(g), \mathrm{HC}(f) - \lfloor \frac{\mathrm{HC}(f)}{\mathrm{HC}(g)} \rfloor\big) \to \cdots$ of the highest coefficients of f and g to the polynomials themselves: $(f, g) \to \big(g, f - \lfloor \frac{\mathrm{HC}(f)}{\mathrm{HC}(g)} \rfloor g\big) \to \cdots$. As soon as the second component becomes zero within the sequence, the degree of one polynomial in the lifted sequence decreases. Note that $\gcd\big(f(p), g(p)\big) = \gcd\big(g(p), f(p) - kg(p)\big)$ holds for all $p, k \in \mathbb{Z}$, which is necessary to show the correctness of our method.

(2) If $\deg(f) > \deg(g)$, we have in general $\mathrm{HC}(g) \not| \mathrm{HC}(f)$, i.e., we cannot reduce f by g within $\mathbb{Z}[p]$. However, a degree reduction of f by g becomes possible if we substitute the parameter p by $lp' + i$ yielding $f_i'(p') := f(lp' + i)$ and $g_i'(p') := g(lp' + i)$, where p' is a new parameter (i.e., we switch from $\mathbb{Z}[p]$ to $\mathbb{Z}[p']$), $l := |\mathrm{HC}(g)|$ and $i \in \{0, \ldots, l - 1\}$. Each i now represents the case that $p \equiv_l i$. In each case, we find that $\mathrm{HC}(g_i') \mid \mathrm{HC}(f_i')$ and we continue the GCD computation with the polynomials g_i' and $f_i' - p'^{\deg(f)-\deg(g)} \frac{\mathrm{HC}(f_i')}{\mathrm{HC}(g_i')} g_i'$.

```
for (i=0; i<=m; i++)
  for (j=0; j<=m; j++)
    A[n*i+2*j] = i+j;
```

(a) code

$$S = \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \end{pmatrix}, U = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & \frac{n}{2} \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -\frac{n}{2} \end{pmatrix}$$

(b) result for $n \equiv_2 0$

FIGURE 1. Data dependence example

We have shown [Sch07] that, in each case the substitution is compatible with the computation of $\gcd\big(f(p), g(p)\big)$. A full algorithm to compute a description of $\gcd\big(f(p), g(p)\big)$ in dependence of p can be implemented by repeating the appropriate step (depending on the degrees of f and g) until one of the polynomials is zero, and termination can be guaranteed because the sum of the degrees of the polynomials involved decreases with each step. The unimodular matrices can be constructed analogously to the non-parametric case. Similarly, this result can be extended from 2×1-matrices to arbitrary $m \times n$-matrices, just as is described by Banerjee [Ban93] for the non-parametric case.

2.2. Backward Substitution

At the core of the non-parametric backward substitution for solving $t\,S = b$ lies the transition from equations of the form $t_i s_i = b'_i$ with $0 \neq s_i \in \mathbb{Z}$, $b'_i \in \mathbb{Z}$ to the representation $t_i = \frac{b'_i}{s_i}$, provided that $s_i \mid b'_i$. In the parametric setting, the equations are of the form $t_i f(p) = g(p)$ with $f, g \in \mathbb{Z}[p]$ (since the entries in S are $\mathcal{AIQ}$-polynomials in p) and we have to describe those p for which $f(p) \mid g(p)$. The following theorem states that a description with finitely many cases is possible.

Theorem 2.3. *Let $f, g \in \mathbb{Z}[p]$ and let $D(f \mid g) \subseteq \mathbb{Z}$ denote the set of integers p with $f(p) \mid g(p)$. Then there are some l induced by the $\mathcal{AIQ}$ GCD of f and g, some finite set $M \subseteq \mathbb{Z}$ and $k \in \{0, \ldots, l\}$ different integers $0 \leq n_1 < \cdots < n_k < l$ such that*

$$D(f \mid g) = M \cup (l\mathbb{Z} + n_1) \cup \cdots \cup (l\mathbb{Z} + n_k).$$

Moreover, for each n_i, we can compute some $h_i \in \mathcal{AIQ}$ such that $h_i(p) = \frac{g(p)}{f(p)}$ whenever $p \equiv_l n_i$.

The n_i and h_i can be computed algorithmically. This shows that the backward substitution can be performed with a finite case distinction. Together with the echelon matrix computation, this yields a procedure for solving systems of equations, as is claimed in Theorem 2.1.

3. Example

Let us consider the program shown in Figure 1(a). To establish which iterations (i, j) and (i', j') write to the same memory location, we have to consider the equation $(i, j, i', j')\,A = 0$ with $A = (n, 2, -n, -2)^T$. The echelon reduction on A

starts by computing the GCD of $f(n) = n$ and $g(n) = 2$. Since their degrees are different, we have the two ($= |\mathrm{HC}(g)|$) cases $f'_0 = 2n'$, $g'_0 = 2$ if $n \equiv_2 0$ and $f'_1 = 2n' + 1$, $g'_1 = 2$ if $n \equiv_2 1$. This yields a $\gcd\big(f(n), g(n)\big)$ of 2 for even n and of 1 for odd n. For space reasons, we omit the other GCD computations and show only the result for even n in Figure 1(b). Backward substitution in $t\,S = 0$ yields $t_1 = 0$, $t_2, t_3, t_4 \in \mathbb{Z}$ and $(i, j, i', j') = t\,U = (t_2, t_3, t_4, \frac{n}{2}(t_2 - t_4) + t_3)$. Finishing the analysis by taking the loop bounds and $(i', j') \prec (i, j)$ into account, we find (since $\frac{n}{2}(t_2 - t_4) + t_3 \leq m$ and $t_2, t_3, t_4 \geq 0$ must hold) that $n \leq 2m$ is a required condition for dependences to exist.

4. Conclusion

Using certain quasi-polynomials, we are able to generalise backward substitution and the GCD computation involved in echelon reduction to the case with one non-linear parameter. This enables us to extend the set of data dependences which can be computed. For space reasons, we have not reported on how to generalise Theorems 2.2 and 2.3 to $f, g \in \mathcal{AIQ}$ and how our procedure can be applied to special multi-parameter cases. The details can be found in [Sch07].

References

[Ban93] Utpal K. Banerjee. *Loop Transformations for Restructuring Compilers: The Foundations*. Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[GGL06] Armin Größlinger, Martin Griebl, and Christian Lengauer. Quantifier elimination in automatic loop parallelization. *Journal of Symbolic Computation*, 41(11):1206–1221, November 2006. doi:10.1016/j.jsc.2005.09.012.

[PW95] William Pugh and David Wonnacott. Nonlinear array dependence analysis. In B. K. Szymanski and B. Sinharoy, editors, *Languages, Compilers and Run-Time Systems for Scalable Computers*, pages 1–14. Kluwer Academic Publishers, Boston, 1995.

[Sch07] Stefan Schster. On algorithmic and heuristic approaches to integral problems in the polyhedron model with non-linear parameters. Diploma thesis, Universität Passau, June 2007. `http://www.infosun.fim.uni-passau.de/cl/arbeiten/schuster-d.pdf`.

[vdD03] Lou van den Dries. Generating the greatest common divisor, and limitations of primitive recursive algorithms. *Foundations of Computational Mathematics*, 3(3):297–324, 2003.

Stefan Schuster
Mühlstraße 33, D-88480 Achstetten-Stetten, Germany
e-mail: `stefan@ngc-6543.de`

Armin Größlinger
Universität Passau, Fakultät für Informatik und Mathematik, D-94030 Passau, Germany
e-mail: `armin.groesslinger@uni-passau.de`