A Generic Framework for Interprocedural Analysis of Numerical Properties

Markus Müller-Olm[1] and Helmut Seidl[2]

[1] Universität Dortmund, Fachbereich Informatik, LS 5
Baroper Str. 301, 44221 Dortmund, Germany
markus.mueller-olm@cs.uni-dortmund.de
[2] TU München, Institut für Informatik, I2
80333 München, Germany
seidl@in.tum.de

Abstract. In his seminal paper [5], Granger presents an analysis which infers linear congruence relations between integer variables. For affine programs without guards, his analysis is *complete*, i.e., infers *all* such congruences. No upper complexity bound, though, has been found for Granger's algorithm. Here, we present a variation of this analysis which runs in polynomial time. Moreover, we provide an interprocedural extension of this algorithm. These algorithms are obtained by means of multiple instances of a general framework for constructing interprocedural analyses of numerical properties. Finally, we indicate how the analyses can be enhanced to deal with equality guards interprocedurally.

1 Introduction

In recent years, a growing interest in the design of very precise analyses of numerical properties of programs could be observed. On the one hand, this comes from a revived interest in aggressive program optimizations as demanded by low-cost embedded processors. On the other hand when designing and implementing critical applications, we are faced with a need for certifying absence of certain program errors [2, 11] or security vulnerabilities such as buffer-overflows [3, 15].

Here, we concentrate on equality-based numerical properties. Such properties are particularly useful, e.g., for induction variable detection or identification of data alignments [1]. This type of analysis has been pioneered by Karr in [9] where he presents a first intraprocedural analysis of valid affine relations over a field. Karr's analysis maintains for every program point a vector space of valid affine relations. Fifteen years later, his analysis was generalized by Granger [4, 5]. Since Granger uses $\mathbb{Z}$ instead of $\mathbb{Q}$, his intraprocedural analysis not only returns valid affine relations but also valid affine congruence relations — with the draw-back, perhaps, that no upper complexity bound is known. Granger's analysis also differs from Karr's in that Granger first determines a linear (in fact affine) abstraction of the sets of intraprocedurally reachable states from which the set of valid relations then is derived in a second step. A forward accumulation of the abstracted collecting semantics is also used by Müller-Olm and Seidl in [12] where (in absence of equality guards) the run-time of Karr's analysis algorithm is improved and also the sizes of occurring numbers is bounded. The same authors also provide the first precise interprocedural extension of Karr's analysis [13] and show how it can be adapted to work not only over fields but also over modular rings $\mathbb{Z}_m$ where $m = 2^w$ as used by standard programming languages like Java [14]. In [6, 7], Gulwani and Necula re-consider Karr's analysis problem. In order to improve on the complexity of the analysis, they propose randomization. In particular, sizes of occurring numbers are bounded by computing modulo random primes.

In this paper, we present general methods how intraprocedural analyses of numerical properties can be constructed which naturally extend to interprocedural analyses of the same properties. Our framework is parametric in the ring within which the computation of the analysis is performed. For the case of affine relations over fields or modular rings $\mathbb{Z}_m$ (m a power of 2), we subsume versions of the intra- and interprocedural analyses from [12, 13] and [14], respectively. Beyond these known analyses, we succeed in deriving an interprocedural extension of Granger's analysis [5] that determines not only all valid affine relations but also all valid congruence relations. We also indicate how the analyses can be enhanced to deal with equality guards interprocedurally.

The immediate interprocedural extension of Granger's analysis as provided by the general framework shares with Granger's original algorithm the draw-back of performing fixpoint iterations over complete lattices with unbounded (though finite) ascending chains. In order to improve on this, we propose a new algorithm which, in absence of procedures, runs in polynomial time. The new algorithm is based on a careful inspection of Granger's analysis problem which allows us to divide the analysis into one analysis over the field $\mathbb{Q}$ together with several analyses over carefully chosen modular rings.

The paper is organized as follows. In section 2 we introduce affine programs together with their collecting semantics. In section 3 we introduce, for every ring R, the R-*linear abstraction* and show how it can be used to determine valid R-linear relations and also (in case of $\mathsf{R} = \mathbb{Z}$) valid linear congruence relations. In section 4, we then show for every *principal ideal ring* R that the R-linear abstraction of the collecting semantics can be computed precisely and provide complexity bounds for fields and modular rings $\mathbb{Z}_m$. In section 5, we particularly deal with the case $\mathsf{R} = \mathbb{Z}$ and provide an alternative algorithm which (at least in absence of equality guards) determines all intraprocedurally valid linear congruence relations in polynomial time. In the interprocedural case, the new algorithm is polynomial if the length of intermediately occurring numbers is polynomially bounded. In section 6, we finally extend the proposed approach to take equality guards into account. Finally, section 7 summarizes and gives hints on directions of future research.

2 The General Set-up

We use similar conventions as in [13] and [14] which we recall here for reasons of selfcontainedness. Thus, programs are modeled by systems of non-deterministic flow graphs that can recursively call each other as in Figure 1. Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ be the set of (global) variables the program operates on. In order to cover the various computational domains of interest, we assume that the variables take values in some commutative ring R with 1 element. In the programs we analyze, we assume the basic statements either to be *affine assignments* of the form $\mathbf{x}_j := t_0 + \sum_{i=1}^{k} t_i \mathbf{x}_i$ (with

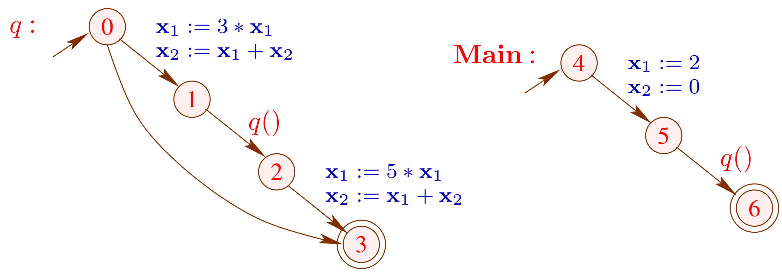

Fig. 1. An interprocedural program.

$t_i \in \mathsf{R}$ for $i = 0, \ldots, k$ and $\mathbf{x}_j \in \mathbf{X}$) or *non-deterministic assignments* of the form $\mathbf{x}_j :=?$ (with $\mathbf{x}_j \in \mathbf{X}$). It is to reduce the number of program points in the example, that we annotated the edges in Figure 1 with sequences of assignments. Also, we use assignments $\mathbf{x}_j := \mathbf{x}_j$ which have no effect onto the program state as skip-statements and omit these in pictures. For the moment, skip-statements are used to abstract guards. Later, we will present methods which treat equality guards more precisely. Non-deterministic assignments $\mathbf{x}_j :=?$ can be used as a safe abstraction of statements in a source program which our analysis cannot handle precisely, for example of assignments $\mathbf{x}_j := t$ with non-affine expressions t or of read statements.

In this setting, an *affine program* comprises a finite set Proc of *procedure names* together with one distinguished procedure $\mathbf{Main}$. Execution starts with a call to $\mathbf{Main}$. Each procedure $q \in \mathsf{Proc}$ is specified by a distinct edge-labeled *control flow graph* with a single start point st_q and a single return point ret_q where each edge is either labeled with an assignment or a call to some procedure.

The basic approach of [13, 12, 14] which we take up here is to construct a precise abstract interpretation of a constraint system characterizing the concrete program semantics. Similar to [5, 12], we find it convenient to start from the *collecting* semantics. For that, we model a *state* attained by program execution when reaching a program point or procedure by a k-dimensional (column) vector[3] $x = [x_1, \ldots, x_k]^{\mathrm{t}} \in \mathsf{R}^k$ of ring elements where x_i is the value assigned to variable $\mathbf{x}_i$. For convenience, we consider *extended* states $[1, x_1, \ldots, x_k]^{\mathrm{t}}$ containing an extra 0-th component 1. Then every assignment $\mathbf{x}_j := t$, $\mathbf{x}_j \in \mathbf{X}$, $t \equiv t_0 + \sum_{i=1}^{k} t_i \mathbf{x}_i$, induces a *linear* transformation $[\![\mathbf{x}_j := t]\!] : \mathsf{R}^{k+1} \to \mathsf{R}^{k+1}$ of the extended state which is described by the matrix:

$$[\![\mathbf{x}_j := t]\!] \;=\; \left[\begin{array}{c|c} I_j & 0 \\ \hline t_0 \ldots t_{j-1} & t_j \ldots t_k \\ \hline 0 & I_{k-j} \end{array}\right]$$

where I_j is the identity matrix in R^{j^2}. This definition is readily extended to sets of extended states. Composition of transformations is captured by matrix multiplication. Since linear mappings are closed under composition, the effect of a single run can be represented by one matrix in $\mathsf{R}^{(k+1)^2}$. Since in general, procedures have multiple runs,

[3] The superscript "t" denotes the *transpose* operation which mirrors a matrix at the main diagonal and changes a row vector into a column vector (and vice versa).

we model their semantics by *sets* of linear transformations. These are characterized by the constraint system $\mathcal{E}_{\mathsf{R}}$:

$$
\begin{array}{lllr}
[\mathcal{E}_{\mathsf{R}}1] & \mathcal{E}_{\mathsf{R}}(q) & \supseteq \mathcal{E}_{\mathsf{R}}(\mathsf{ret}_q) & \\
[\mathcal{E}_{\mathsf{R}}2] & \mathcal{E}_{\mathsf{R}}(\mathsf{st}_q) & \supseteq \{I_{k+1}\} & \\
[\mathcal{E}_{\mathsf{R}}3] & \mathcal{E}_{\mathsf{R}}(v) & \supseteq \mathcal{E}_{\mathsf{R}}(u) \cdot \{[\![\mathbf{x}_j := t]\!]\} & \text{if edge } (u,v) \text{ is labeled } \mathbf{x}_j := t \\
[\mathcal{E}_{\mathsf{R}}4] & \mathcal{E}_{\mathsf{R}}(v) & \supseteq \mathcal{E}_{\mathsf{R}}(u) \cdot \{[\![\mathbf{x}_j := c]\!] \mid c \in \mathsf{R}\} & \text{if edge } (u,v) \text{ is labeled } \mathbf{x}_j :=? \\
[\mathcal{E}_{\mathsf{R}}5] & \mathcal{E}_{\mathsf{R}}(v) & \supseteq \mathcal{E}_{\mathsf{R}}(u) \cdot \mathcal{E}_{\mathsf{R}}(q) & \text{if edge } (u,v) \text{ calls } q
\end{array}
$$

The variable $\mathcal{E}_{\mathsf{R}}(q)$ is meant to capture the set of effects of the procedure q. By the constraints $\mathcal{E}_{\mathsf{R}}1$, this value is obtained as the set of transformations $\mathcal{E}_{\mathsf{R}}(\mathsf{ret}_q)$ for the return point ret_q of q. According to $\mathcal{E}_{\mathsf{R}}2$, this accumulation starts at the start point st_q with the identity transformation. The constraints $\mathcal{E}_{\mathsf{R}}3$ and $\mathcal{E}_{\mathsf{R}}4$ deal with affine and nondeterministic assignments, respectively, while the constraints $\mathcal{E}_{\mathsf{R}}5$ correspond to calls.

Given the effects of procedures, we characterize the sets of extended states reaching program points and procedures by the constraint system $\mathcal{C}_{\mathsf{R}}$:

$$
\begin{array}{lllr}
[\mathcal{C}_{\mathsf{R}}1] & \mathcal{C}_{\mathsf{R}}(\mathbf{Main}) & \supseteq \{1\} \times \mathsf{R}^k & \\
[\mathcal{C}_{\mathsf{R}}2] & \mathcal{C}_{\mathsf{R}}(q) & \supseteq \mathcal{C}_{\mathsf{R}}(u) & \text{if edge } (u, \_) \text{ calls } q \\
[\mathcal{C}_{\mathsf{R}}3] & \mathcal{C}_{\mathsf{R}}(\mathsf{st}_q) & \supseteq \mathcal{C}_{\mathsf{R}}(q) & \\
[\mathcal{C}_{\mathsf{R}}4] & \mathcal{C}_{\mathsf{R}}(v) & \supseteq [\![\mathbf{x}_j := t]\!](\mathcal{C}_{\mathsf{R}}(u)) & \text{if edge } (u,v) \text{ is labeled } \mathbf{x}_j := t \\
[\mathcal{C}_{\mathsf{R}}5] & \mathcal{C}_{\mathsf{R}}(v) & \supseteq \bigcup\{[\![\mathbf{x}_j := c]\!](\mathcal{C}_{\mathsf{R}}(u)) \mid c \in \mathsf{R}\} & \text{if edge } (u,v) \text{ is labeled } \mathbf{x}_j :=? \\
[\mathcal{C}_{\mathsf{R}}6] & \mathcal{C}_{\mathsf{R}}(v) & \supseteq \mathcal{E}_{\mathsf{R}}(q)(\mathcal{C}_{\mathsf{R}}(u)) & \text{if edge } (u,v) \text{ calls } q
\end{array}
$$

The constraint $\mathcal{C}_{\mathsf{R}}1$ indicates that we start before the call of $\mathbf{Main}$ with the full (extended) state space. The constraints $\mathcal{C}_{\mathsf{R}}2$ indicate that the extended states reaching a procedure includes all extended states reaching its calls and the constraints $\mathcal{C}_{\mathsf{R}}3$ state that the extended states reaching a call to a procedure also reach its start point. The constraints $\mathcal{C}_{\mathsf{R}}4$ through $\mathcal{C}_{\mathsf{R}}6$ then are completely analogous to a usual forward propagating definition of the *intra-procedural* collecting semantics only that at a call edge the set of transformations obtained for the called procedure is applied (constraints $\mathcal{C}_{\mathsf{R}}6$).

By the fixpoint theorem of Knaster-Tarski, the constraint systems $\mathcal{E}_{\mathsf{R}}$ and $\mathcal{C}_{\mathsf{R}}$ have least solutions. For convenience, we denote the components of these least solutions by $\mathcal{E}_{\mathsf{R}}(X)$, and $\mathcal{C}_{\mathsf{R}}(X)$, respectively ($X$ a procedure name or program point).

3 The Linear Abstraction

Program analyses of numerical program properties are based on abstractions of subsets of vectors. Here, we consider the abstraction of a set $V \subseteq \mathsf{R}^{k+1}$ of extended states by the R-*linear closure* of V:

$$\alpha_{\mathsf{R}}(V) = \langle V \rangle_{\mathsf{R}} = \{\lambda_1 v_1 + \ldots + \lambda_s v_s \mid s \geq 0, \lambda_i \in \mathsf{R}, v_i \in V\}\,.$$

Due to the extension of states by an extra 0-th component, the abstraction adds all *linear* combinations of vectors in V – with the understanding that only those vectors in the closure are meaningful whose 0-th components equal 1. We remark that $\alpha_{\mathsf{R}}(V)$ is closed under vector addition and multiplication with ring elements $r \in \mathsf{R}$. Such sets are called

R-*modules* where the set $\langle V\rangle_{\mathsf{R}}$ is the R-module *generated* by V. It is well-known that for any r, the R-submodules of R^r are closed under intersection. Ordered by set inclusion (which we denote by $\sqsubseteq$ in the context of submodules) they thus form a complete lattice $\mathbf{Sub}(\mathsf{R}^r)$, like the linear subspaces of $\mathbb{F}^r$ for a field $\mathbb{F}$. The least element of $\mathbf{Sub}(\mathsf{R}^r)$ is $\{\mathbf{0}\}$ consisting of the zero vector only, the greatest element is R^r itself. The least upper bound of two R-submodules M_1, M_2 is

$$M_1 \sqcup M_2 = \langle M_1 \cup M_2\rangle_{\mathsf{R}} = \{m_1 + m_2 \mid m_i \in M_i\}\,.$$

The linear abstraction has been extensively studied for different rings. In [5], it is used with $\mathsf{R} = \mathbb{Z}$ to analyze linear congruence relations. In [12], this abstraction is applied for fields to speed up Karr's analysis [9] of affine relations. Interestingly, the interprocedural analyses of affine relations [13, 14] over fields and modular rings $\mathbb{Z}_m$, $m = 2^w$, do not directly rely on abstractions of the collecting semantics but on linear abstractions of sets of weakest precondition transformers.

In general, we are interested in numerical properties P which invariantly hold for all (extended) states x in the collecting semantics at a given program point. Clearly, the linear abstraction can only be used to detect properties which are invariant under linear combinations of the extended state or, equivalently, affine combinations of the program state. In particular, this is the case for *affine* relations between program variables like, e.g., $2 - 4\mathbf{x}_1 + 3\mathbf{x}_2 = 0$. Since we work with extended states, we can rely on the simpler *linear* relations on extended states here. In general, a linear relation over a ring R is a (row) vector $a = [a_0, \ldots, a_k]$ where $x = [x_0, \ldots, x_k]^t$ satisfies a iff $a \cdot x = \sum_{i=0}^{k} a_i x_i = 0$. The set of affine relations satisfied by a set of states coincides with the set of linear relations satisfied by the corresponding set of extended states. We observe:

Fact 1 *For every ring* R *the following holds:*

1. *For every row vector* a*, the set* $\{x \in \mathsf{R}^{k+1} \mid a \cdot x = 0\}$ *is an* R*-module.*
2. *For every set* $G \subseteq \mathsf{R}^{k+1}$*,*

$$\langle G\rangle_{\mathsf{R}}^{\perp} =_{\mathsf{def}} \{a \mid \forall x \in G : a \cdot x = 0\} = \{a \mid \forall x \in \langle G\rangle_{\mathsf{R}} : a \cdot x = 0\}\,.$$

Moreover, the set $\langle G\rangle_{\mathsf{R}}^{\perp}$ *is an* R*-module.* □

Assume that the R-module $\langle \mathcal{C}_{\mathsf{R}}(X)\rangle_{\mathsf{R}}$ is generated by the finite set $G \subseteq \mathsf{R}^{k+1}$. Then by fact 1, we can determine the set of all valid linear relations at X as the set of all solutions of the homogeneous system of equations:

$$\mathbf{a} \cdot x = 0\,, \quad x \in G$$

where $\mathbf{a} = [\mathbf{a}_0, \ldots, \mathbf{a}_k]$ is a row vector of variables. Here, we are mostly interested in *principal ideal rings* (or PIRs). A principal ring R is a commutative ring with 1 in which every ideal is *principal*. Recall that an *ideal* $I \subseteq \mathsf{R}$ is a subset of R which is closed under addition and multiplication with arbitrary ring elements, i.e., $a+b \in I$ whenever $a, b \in I$ and $r \cdot a \in I$ whenever $a \in I$ and $r \in \mathsf{R}$. An ideal I is principal if it is generated by a single element, i.e., $I = \{r \cdot d \mid r \in \mathsf{R}\}$ for some $d \in \mathsf{R}$. PIRs comprise not only fields but also the integral domain $\mathbb{Z}$ as well as all modular rings $\mathbb{Z}_m$, $m \geq 2$. In [8, 16], efficient methods are developed for computing various normal forms of matrices over PIRs. The most notable property of PIRs is that they allow us to solve linear systems of equations by a generalized Gaussian elimination algorithm. Of particular importance is the integral domain $\mathbb{Z}$. Assume $G \subseteq \mathbb{Z}^{k+1}$ is a set of integer vectors. Then the set of all *linear* relations which are valid for G is (up to multiplication with constants) identical to the set of linear relations which are valid over the $\mathbb{Q}$-module generated by G:

Fact 2 *For every subset $G \subseteq \mathbb{Z}^{k+1}$ of column vectors and every row vector $a \in \mathbb{Z}^{k+1}$, the following statements are equivalent:*

1. $a \cdot x = 0$ *for all* $x \in G$*;*
2. $a \cdot x = 0$ *for all* $x \in \langle G \rangle_{\mathbb{Z}}$*;*
3. $a \cdot x = 0$ *for all* $x \in \langle G \rangle_{\mathbb{Q}}$. □

Assume we want to determine the set of valid $\mathbb{Z}$-linear relations at a program point X. By fact 2, it suffices to determine the linear relations which are valid for $\langle \mathcal{C}_{\mathbb{Z}}(X) \rangle_{\mathbb{Q}}$. Since $\mathbb{Q}$ is a field, these can be computed efficiently with the techniques from [13, 12]. It therefore does not pay off to determine the (complicated) $\mathbb{Z}$-linear closure of the collecting semantics if we are interested in linear relations only.

In [5], however, Granger considers a more general form of properties, namely, *linear congruence relations*. A linear congruence equation is an equation $a \cdot \mathbf{x} \equiv 0\ [m]$ where $a \in \mathbb{Z}$ is a row vector and $m > 0$ is the integer modulus. The column vector $x \in \mathbb{Z}^{k+1}$ satisfies the congruence relation iff $a \cdot x \equiv 0\ [m]$ or, equivalently, $a \cdot x + mz = 0$ for some $z \in \mathbb{Z}$. A linear relation of the extended state can be seen as a particular linear congruence relation if we allow m to equal 0. If $m > 1$, we can assume that all components of a are in the range $\{0, \ldots, m-1\}$. The set of all x satisfying a linear congruence relation is closed under addition and multiplication with elements of $\mathbb{Z}$ and therefore a $\mathbb{Z}$-module. In [5], Granger shows that every $\mathbb{Z}$-module can also be represented as the set of solutions of a finite number of linear congruence relations. For later use, we provide a refinement of his characterization. We introduce the following auxiliary notions. Assume that $G \subseteq \mathbb{Z}^r$ is a set of q linearly independent[4] column vectors. Let $V \subseteq \mathbb{Z}^{r \cdot q}$ denote the matrix formed by the vectors in G. Using generalized Gaussian elimination, some unimodular matrix[5] $T \in \mathbb{Z}^{r^2}$ can be constructed such that $T \cdot V = \left[\frac{D}{0}\right]$ for an upper triangular square matrix D. Then we define $\mathsf{det}(G)$ as the absolute value of the determinant of D. It follows from uniqueness of the Hermite normal form [17, 16] that this definition is independent of the choice of T. We obtain:

Theorem 1. *Assume $G \subseteq \mathbb{Z}^r$ is a set of linearly independent vectors where $\mathsf{det}(G)$ divides $m > 0$. Let E_0 and E_m denote finite sets of generators for $\langle G \rangle_{\mathbb{Z}}^{\perp}$ and $\langle G \rangle_{\mathbb{Z}_m}^{\perp}$, respectively. Then the following holds:*

1. *$\langle G \rangle_{\mathbb{Z}}$ is the set of solutions of the system*
$$a \cdot \mathbf{x} = 0, a \in E_0, \qquad b \cdot \mathbf{x} \equiv 0\ [m], b \in E_m$$
2. *Another linear congruence relation $b' \cdot \mathbf{x} \equiv 0\ [m']$ is satisfied by all vectors in G iff the following holds. If $m' = 0$ then $b' \in \langle E_0 \rangle_{\mathbb{Z}}$. Otherwise, let h denote the least common multiple of m and m' where $m \cdot d = h$ and $m' \cdot d' = h$. Then $d' \cdot b'$ is contained in $\langle E_0 \cup \{d \cdot b \mid b \in E_m\} \rangle_{\mathbb{Z}_h}$.*

[4] Recall that G is linearly independent over $\mathbb{Q}$ iff G is linearly independent over $\mathbb{Z}$.

[5] An integer matrix is *unimodular* iff its determinant equals ± 1.

For a proof of this theorem, see appendix A. By the second statement, the sets E_0 and E_m allow us, for every other modulus m', to determine a finite set E' of generators of all valid linear relations modulo m'. First, we construct the set $E = E_0 \cup \{d \cdot b \mid b \in E_m\}$ where $h = d \cdot m$ is the least common multiple of m and m'. The idea is now to determine E' as a finite set of generators of all $\mathbb{Z}_h$-linear combinations of vectors in E which contain $d' = \frac{h}{m'}$ as a factor. For this, let V denote the matrix whose rows are formed by the vectors in E. Then a vector v is a linear combination of the vectors in E which contains d' as a factor iff $v = y \cdot V$ for some $y \in \mathbb{Z}_h^{|E|}$ such that $m' \cdot (y \cdot V) \equiv \mathbf{0}\ [h]$. Thus, we first compute generators $b_1, \ldots, b_q \in \mathbb{Z}_h^{|E|}$ for the module of solutions of the equation system $\mathbf{y}\,(m' \cdot V) \equiv \mathbf{0}\ [h]$. The vectors $b_i\, V$ can be written as $b_i\, V = d' b_i'$ — giving us the set $E' = \{b_1', \ldots, b_q'\}$ of generators for all valid linear relations modulo m'.

Theorem 1 allows us to compute the linear congruence relations which are valid at X from the $\mathbb{Z}$-linear closure of $\mathcal{C}_{\mathbb{Z}}(X)$, the set of extended states reaching X. Our new observation is that, instead of computing the $\mathbb{Z}$-linear closure of the reachable states, we can decompose the analysis into an analysis returning all valid linear relations plus an analysis returning all valid linear relations modulo a carefully chosen m. If on the other hand, we are interested in the linear closure of the reachable extended states at X, then we can recover these from the linear equations together with the valid linear equations modulo m by solving an appropriate homogeneous system of equations.

4 Constructing Interprocedural Analyses

We have seen that for affine programs, the effects of procedures are given by sets of linear transformations, or matrices. Matrices in turn can be viewed as vectors — only with quadratically many components. We therefore can use the same abstraction α_{R} for effects which we use for sets of extended state vectors. By applying α_{R} to the constraint systems $\mathcal{E}_{\mathsf{R}}$ and $\mathcal{C}_{\mathsf{R}}$, we obtain constraint systems $\mathcal{E}_{\mathsf{R}}^\sharp$ and $\mathcal{C}_{\mathsf{R}}^\sharp$:

$[\mathcal{E}_{\mathsf{R}}^\sharp 1]$	$\mathcal{E}_{\mathsf{R}}^\sharp(q) \sqsupseteq \mathcal{E}_{\mathsf{R}}^\sharp(\mathsf{ret}_q)$	
$[\mathcal{E}_{\mathsf{R}}^\sharp 2]$	$\mathcal{E}_{\mathsf{R}}^\sharp(\mathsf{st}_q) \sqsupseteq \langle\{I_{k+1}\}\rangle_{\mathsf{R}}$	
$[\mathcal{E}_{\mathsf{R}}^\sharp 3]$	$\mathcal{E}_{\mathsf{R}}^\sharp(v) \sqsupseteq \mathcal{E}_{\mathsf{R}}^\sharp(u) \cdot \langle\{[\![\mathbf{x}_j := t]\!]\}\rangle_{\mathsf{R}}$	if edge (u,v) is labeled $\mathbf{x}_j := t$
$[\mathcal{E}_{\mathsf{R}}^\sharp 4]$	$\mathcal{E}_{\mathsf{R}}^\sharp(v) \sqsupseteq \mathcal{E}_{\mathsf{R}}^\sharp(u) \cdot \langle\{[\![\mathbf{x}_j := 0]\!], [\![\mathbf{x}_j := 1]\!]\}\rangle_{\mathsf{R}}$	if edge (u,v) is labeled $\mathbf{x}_j :=?$
$[\mathcal{E}_{\mathsf{R}}^\sharp 5]$	$\mathcal{E}_{\mathsf{R}}^\sharp(v) \sqsupseteq \mathcal{E}_{\mathsf{R}}^\sharp(u) \cdot \mathcal{E}_{\mathsf{R}}^\sharp(q)$	if edge (u,v) calls q

As in [13, 12], the abstract effect of a non-deterministic assignment $\mathbf{x}_j :=?$ can be modeled by the span of the two transformations $[\![\mathbf{x}_j := 0]\!]$ and $[\![\mathbf{x}_j := 1]\!]$.

The constraint system $\mathcal{E}_{\mathsf{R}}^\sharp$ closely resembles the corresponding constraint systems as presented in [13] and [14]. There, however, the accumulated transformations are interpreted as *weakest precondition transformers* and therefore accumulated from the rear. The constraint system now accumulates values in a forward fashion. Accordingly, the second constraint system $\mathcal{C}_{\mathsf{R}}^\sharp$ is in the spirit of the forward *intraprocedural* accumulation as used, e.g., in [12]. Thus, in contrast to [13, 14], the second constraint system *directly* speaks about abstract sets of values and not about abstract sets of transformations:

$$
\begin{array}{lll l}
[\mathcal{C}^\sharp_{\mathsf{R}}1] & \mathcal{C}^\sharp_{\mathsf{R}}(\mathbf{Main}) & \sqsupseteq \mathsf{R}^{k+1} & \\
[\mathcal{C}^\sharp_{\mathsf{R}}2] & \mathcal{C}^\sharp_{\mathsf{R}}(q) & \sqsupseteq \mathcal{C}^\sharp_{\mathsf{R}}(u) & \text{if edge } (u,\_) \text{ calls } q \\
[\mathcal{C}^\sharp_{\mathsf{R}}3] & \mathcal{C}^\sharp_{\mathsf{R}}(\mathsf{st}_q) & \sqsupseteq \mathcal{C}^\sharp_{\mathsf{R}}(q) & \\
[\mathcal{C}^\sharp_{\mathsf{R}}4] & \mathcal{C}^\sharp_{\mathsf{R}}(v) & \sqsupseteq [\![\mathbf{x}_j := t]\!]\,(\mathcal{C}^\sharp_{\mathsf{R}}(u)) & \text{if edge } (u,v) \text{ is labeled } \mathbf{x}_j := t \\
[\mathcal{C}^\sharp_{\mathsf{R}}5] & \mathcal{C}^\sharp_{\mathsf{R}}(v) & \sqsupseteq [\![\mathbf{x}_j := 0]\!]\,(\mathcal{C}^\sharp_{\mathsf{R}}(u)) \sqcup & \\
 & & \quad [\![\mathbf{x}_j := 1]\!]\,(\mathcal{C}^\sharp_{\mathsf{R}}(u)) & \text{if edge } (u,v) \text{ is labeled } \mathbf{x}_j :=? \\
[\mathcal{C}^\sharp_{\mathsf{R}}6] & \mathcal{C}^\sharp_{\mathsf{R}}(v) & \sqsupseteq \mathcal{E}^\sharp_{\mathsf{R}}(q)(\mathcal{C}^\sharp_{\mathsf{R}}(u)) & \text{if edge } (u,v) \text{ calls } q
\end{array}
$$

By the fixpoint theorem of Knaster-Tarski, the constraint systems $\mathcal{E}^\sharp_{\mathsf{R}}$ and $\mathcal{C}^\sharp_{\mathsf{R}}$ have least solutions. Again, we denote the components of these least solutions by $\mathcal{E}^\sharp_{\mathsf{R}}(X)$ and $\mathcal{C}^\sharp_{\mathsf{R}}(X)$, respectively ($X$ a procedure or program point). Abstracting the collecting semantics according to constraint system $\mathcal{C}^\sharp_{\mathsf{R}}$ has the advantage that it relies on matrices only for procedure calls. This means that we can take advantage from any improvements on the abstractions, e.g., for guards $g = 0$ (g an affine combination) or non-affine assignments which have been proposed for the intraprocedural analysis [9, 5].

Furthermore, we verify that the abstraction commutes with the application and with the composition of transformations. By linearity we have:

Proposition 1. *Let* R *denote a commutative ring with 1. Then:*

1. $\langle\{A\,x \mid x \in V, A \in M\}\rangle_{\mathsf{R}} = \langle\{A\,x \mid x \in \langle V\rangle_{\mathsf{R}}, A \in \langle M\rangle_{\mathsf{R}}\}\rangle_{\mathsf{R}}$
2. $\langle\{A_1\,A_2 \mid A_i \in M_i\}\rangle_{\mathsf{R}} = \langle\{A_1\,A_2 \mid A_i \in \langle M_i\rangle_{\mathsf{R}}\}\rangle_{\mathsf{R}}$

for every set of vectors $V \subseteq \mathsf{R}^{k+1}$ *and sets of matrices* $M, M_1, M_2 \subseteq \mathsf{R}^{(k+1)^2}$. □

By the fixpoint transfer lemma, we therefore obtain from proposition 1, for the constraint systems $\mathcal{E}^\sharp_{\mathsf{R}}$ and $\mathcal{C}^\sharp_{\mathsf{R}}$:

Theorem 2. *For a program interpreted over a ring* R*, the following holds:*

1. $\mathcal{E}^\sharp_{\mathsf{R}}(q) = \langle\mathcal{E}_{\mathsf{R}}(q)\rangle_{\mathsf{R}}$ *for every procedure* q*;*
2. $\mathcal{C}^\sharp_{\mathsf{R}}(X) = \langle\mathcal{C}_{\mathsf{R}}(X)\rangle_{\mathsf{R}}$ *for every procedure or program point* X. □

Theorem 2 gives a precise characterization of the linear closure of the collecting semantics through a constraint system. Note that for *principal ideal rings* R, the lattice of R-submodules of R^r satisfies the ascending chain condition. If R is a field, the length of every strictly increasing sequence of R-submodules of R^r is bounded by r for dimensional reasons. If R is a modular ring $\mathbb{Z}_m$, then the length of every strictly increasing sequence of R-submodules of R^r can be shown to be bounded by $r \cdot \log(m)$. If R is the ring of integers, the lengths of strictly increasing sequences of R-submodules, though finite, cannot be bounded.

Secondly, we note that every R-submodule M of R^r can be represented by $M = \langle G\rangle_{\mathsf{R}}$ for a set G of at most r generators. Accordingly, inclusion of R-submodules can be reduced to deciding for a vector $v \in \mathsf{R}^r$ whether or not $v \in \langle G\rangle_{\mathsf{R}}$ for a finite subset $G \subseteq \mathsf{R}^r$. If $G = \{v_1, \ldots, v_s\}$, the latter problem consists in deciding whether there exist $\lambda_1, \ldots, \lambda_s \in \mathsf{R}$ such that

$$\lambda_1 v_1 + \ldots + \lambda_s v_s = v$$

Thus, the problem reduces to solving inhomogeneous systems of linear equations. If R is a field, this can be achieved, e.g., by standard Gaussian elimination. Instead, we may rely on reduction to *echelon form* as discussed in [8, 16]. Therefore, theorem 2 gives rise to an effective analysis over any *effective* PIR R, i.e., every PIR R where 0 and 1, equality as well as the arithmetic operations and the basic principal ideal operations are computable. The ideal operations we need are a *generalized gcd* and effective methods for solving *one variable* equations $a \cdot \mathbf{x}_1 = b$ with $a, b \in \mathsf{R}$ (see again [8, 16] for details). Summarizing, we have:

Theorem 3. *Assume p is an affine program over an effective PIR R. Then the least solutions of the constraint systems $\mathcal{E}^\sharp_\mathsf{R}$ and $\mathcal{C}^\sharp_\mathsf{R}$ are effectively computable.* □

In particular, we obtain interprocedural algorithms for computing the linear closures of the collecting semantics for fields as well as for all modular rings — thus giving us algorithms for computing all valid linear relations. The corresponding run-time complexities for a program of size n with k variables are summarized in figure 2. For simplicity, we have assumed unit cost for every arithmetic operation as well as for the principal ideal operations. The first line reports the results obtained in [12, 13], while the result on $\mathbb{Z}_m$

R	intraprocedural	interprocedural
field	$\mathcal{O}(n \cdot k^3)$	$\mathcal{O}(n \cdot k^8)$
$\mathbb{Z}_m$	$\mathcal{O}(n \cdot k^3 \cdot \log(m))$	$\mathcal{O}(n \cdot k^8 \cdot \log(m))$

Fig. 2. Unit cost complexity of computing the R-linear closure.

is the generalization of [14] to arbitrary modular rings. Theorem 3 also provides us with an interprocedural generalization of Granger's analysis. The complexity, however, remains unclear here, since ascending chains of $\mathbb{Z}$-modules can have arbitrary lengths.

5 Efficient Linear Congruence Analysis

In this section, we refine the general approach for PIRs for the case $\mathsf{R} = \mathbb{Z}$ in order to obtain a *polynomial time* algorithm for computing all intraprocedurally valid linear congruence relations. This algorithm also extends to a fast interprocedural algorithm — provided that mild restrictions on occurring numbers are satisfied.

Theorem 4. *Assume p is an affine program over $\mathbb{Z}$ of size n with k variables.*

1. *For every program point or procedure X, we can compute a (finite) representation of the set of all linear congruence relations valid at X.*
2. *Intraprocedurally, these representations can be computed in polynomial time.*
3. *Interprocedurally, these representations can be computed in exponential time.*

Proof. Assume the program p has k program variables. The algorithm achieving the explicit complexity bounds is based on theorem 1. It proceeds in three phases.

Phase 1: We compute the least solutions of the constraint systems $\mathcal{E}^\sharp_\mathbb{Q}$ and $\mathcal{C}^\sharp_\mathbb{Q}$. More precisely, we compute for every program point or procedure X, linearly independent subsets $G_\mathcal{E}(X) \subseteq \mathcal{E}_\mathbb{Z}(X)$, $G_\mathcal{C}(X) \subseteq \mathcal{C}_\mathbb{Z}(X)$ such that

$$\mathcal{E}^\sharp_\mathbb{Q}(X) = \langle G_\mathcal{E}(X)\rangle_\mathbb{Q} \qquad \mathcal{C}^\sharp_\mathbb{Q}(X) = \langle G_\mathcal{C}(X)\rangle_\mathbb{Q}$$

Then we determine for every X, a set of generators for the set of all $\mathbb{Z}$-linear relations which are valid at X.

Phase 2: For every X, we determine $m(X)$ as the determinant $\mathsf{det}(G_\mathcal{C}(X))$.

Phase 3: For every X, we solve the constraint systems $\mathcal{E}^\sharp_{\mathbb{Z}_m}$ and $\mathcal{C}^\sharp_{\mathbb{Z}_m}$ for $m = m(X)$. This allows us to determine the $\mathbb{Z}_m$-module $\mathcal{C}^\sharp_{\mathbb{Z}_m}(X)$ and compute a set of generators of the $\mathbb{Z}_m$-linear relations which are valid at X.

We successively discuss the three phases of the algorithm. The first phase is readily implemented by a variant of the algorithm proposed in [13] for solving constraint system $\mathcal{E}^\sharp_\mathbb{Q}$ and (an adapted version of) [12] for then solving $\mathcal{C}^\sharp_\mathbb{Q}$. These algorithms are based on semi-naive fixpoint iteration and generate for every program point or procedure X a basis consisting of matrices from $\mathcal{E}_\mathbb{Z}(X)$ and (extended) states from $\mathcal{C}_\mathbb{Z}(X)$, respectively.

Example 1. Consider, e.g., the program from section 2. We find the matrices:

$$Q_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Q_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 15 & 0 \\ 0 & 18 & 1 \end{bmatrix} \quad Q_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 225 & 0 \\ 0 & 282 & 1 \end{bmatrix}$$

which are contained in $\mathcal{E}_\mathbb{Z}(q)$ and together generate the vector space $\mathcal{E}^\sharp_\mathbb{Q}(q)$. Using these matrices, we determine a set of generators for the vector-space $\mathcal{C}^\sharp_\mathbb{Q}(6)$ as:

$$z_0 = [1, 2, 0]^\mathrm{t} \quad z_1 = [1, 30, 36]^\mathrm{t} \quad z_2 = [1, 450, 564]^\mathrm{t}$$

□

Since $\langle \mathcal{C}_\mathbb{Z}(X)\rangle_\mathbb{Q} = \langle \mathcal{C}_\mathbb{Q}(X)\rangle_\mathbb{Q}$, fact 1 implies that the $\mathbb{Z}$-module $\langle G_\mathcal{C}(X)\rangle^\perp_\mathbb{Z}$ already equals the $\mathbb{Z}$-module $\langle \mathcal{C}_\mathbb{Z}(X)\rangle^\perp_\mathbb{Z}$, i.e., the set of valid linear equalities.

Let $\Delta_\mathcal{E}, \Delta_\mathcal{C}$ denote the maximal absolute sizes of the entries of the matrices and vectors, respectively, in the sets of generators used by the fixpoint computation over $\mathbb{Q}$. By inspecting the algorithms in [13, 12], we find:

$$\Delta_\mathcal{E} \;\leq\; 2^{2^{\mathcal{O}(n\cdot k^2)}} \qquad \Delta_\mathcal{C} \;\leq\; \Delta_\mathcal{E}^{\mathcal{O}(n\cdot k)}$$

In general, solving the constraint systems $\mathcal{E}^\sharp_\mathbb{Q}$ and $\mathcal{C}^\sharp_\mathbb{Q}$ over $\mathbb{Q}$ thus can be performed by $\mathcal{O}(n \cdot k^8)$ operations using arithmetic for numbers bounded in length by $\mathcal{O}(n \cdot k^2 \cdot \log(\Delta_\mathcal{E}))$. In case of an intra-procedural analysis, we can completely abandon the constraint system $\mathcal{E}^\sharp_\mathbb{Q}$. Adapting the algorithm from [12], we need just $\mathcal{O}(n \cdot k^3)$ arithmetic operations on numbers of length $\mathcal{O}(n \cdot k^2)$.

We turn to phase 2. Given a linearly independent set $G_\mathcal{C}(X)$ of cardinality q, we compute the determinant $m(X) = \mathsf{det}(G_\mathcal{C}(X))$ with a polynomial number of bit operations, e.g., using the methods of Storjohann [17, 16]. In our application the length of the computed determinant (and thus also of $\log(m(X))$) is bounded by $\mathcal{O}(n \cdot k^2 \cdot \log(\Delta_\mathcal{E}))$. Let G denote a linearly independent set of generators of $\mathcal{C}^\sharp_\mathbb{Z}(X) = \langle \mathcal{C}_\mathbb{Z}(X)\rangle_\mathbb{Z}$. Since $\langle G\rangle_\mathbb{Q} = \langle G_\mathcal{C}(X)\rangle_\mathbb{Q}$, G has cardinality q as well.

Claim: $\mathsf{det}(G)$ divides $\mathsf{det}(G_{\mathcal{C}}(X))$.

This central claim together with theorem 1 implies that the set of all linear congruence relations valid at X can be derived from the set of all linear relations valid at X (as computed in phase 1) together with all linear congruence relations modulo $m(X)$ (as computed in phase 3).

We turn to the proof of the claim. Let $V \in \mathbb{Z}^{(k+1)\cdot q}$ and $V' \in \mathbb{Z}^{(k+1)\cdot q}$ denote the coefficient matrices formed by the vectors of $G_{\mathcal{C}}(X)$ and G, respectively. By definition, there are square unimodular matrices $T, T' \in \mathbb{Z}^{(k+1)^2}$ such that $T \cdot V = \left[\frac{D}{0}\right]$ and $T' \cdot V' = \left[\frac{D'}{0}\right]$ for square upper triangular matrices D, D' where the product of the diagonal elements of D and D' equals $\mathsf{det}(G_{\mathcal{C}}(X))$ and $\mathsf{det}(G)$, respectively. Since $G_{\mathcal{C}}(X) \subseteq \langle G \rangle_{\mathbb{Z}}$, there is also a square matrix $S \in \mathbb{Z}^{q^2}$ such that $V = V' \cdot S$. Therefore, $D = T_1 \cdot D' \cdot S$ where T_1 is the left upper $(q \times q)$-submatrix of $T \cdot (T')^{-1}$ and, thus, $\mathsf{det}(D) = \mathsf{det}(T_1) \cdot \mathsf{det}(D') \cdot \mathsf{det}(S)$. Since T_1 and S are integer matrices the claim follows.

Example 2. Starting from the vectors z_0, z_1, z_2 for program point 6 of example 1, we may apply elementary row transformations (over $\mathbb{Z}$) each with determinant 1 to the coefficient matrix of the z_i. Thus, we obtain the matrix:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 4 & 700 \\ 0 & 0 & -84 \end{bmatrix}$$

Thus, the determinant equals $m(6) = 1 \cdot 4 \cdot 84 = 336$ — serving as the modulus for the third stage. Since the three vectors z_0, z_1, z_2 are linearly independent, they span the complete vector space $\mathbb{Q}^3$. Therefore, no non-trivial linear relation holds for every reachable state at program point 6. □

In phase 3, it remains to determine the set of all linear relations *modulo* $m(X)$ which hold for all vectors in $\mathcal{C}^\sharp_{\mathbb{Z}}(X)$. Since taking integers modulo $m(X)$ is a homomorphism, we conclude that the $\mathbb{Z}_{m(X)}$-module $\langle \mathcal{C}^\sharp_{\mathbb{Z}_{m(X)}}(X) \rangle^{\perp}_{\mathbb{Z}_{m(X)}}$ equals the set of all linear congruence relations which are valid at X modulo $m(X)$. Note further that the third phase of fixpoint iteration for the constraint systems over $\mathbb{Z}_{m(X)}$ need not start from scratch but can use the generators computed in the first phase modulo $m(X)$ as start value.

Example 3. We turn to phase 3 for our example program. Recall that the modulus for program point 6 equals 336. Accordingly, we determine the least solutions of the constraint systems $\mathcal{E}^\sharp_{\mathbb{Z}_{336}}, \mathcal{C}^\sharp_{\mathbb{Z}_{336}}$. We start with the already obtained sets of generators — modulo 336. In order to obtain a subsumption test for $\mathcal{E}^\sharp_{\mathbb{Z}_{336}}$ at variable q, we bring the set of matrices $\{Q_0, Q_1, Q_2\}$ computed in example 1 into echelon form (modulo 336). In our case this results in the matrices:

$$Q'_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Q'_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 14 & 0 \\ 0 & 18 & 0 \end{bmatrix} \quad Q'_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 6 & 0 \end{bmatrix}$$

Propagating, e.g., the matrix Q_2 for the call at the edge $(1, 2)$, we obtain:

$$Q_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 15 & 0 \\ 0 & 192 & 1 \end{bmatrix}$$

Matrix Q_3 is already subsumed by the Q'_i. The same also holds for the propagation of the matrices Q_0 and Q_1. Therefore, the set $\{Q_0, Q_1, Q_2\}$ already represents the fixpoint. Accordingly, the module $\mathcal{C}^\sharp_{\mathbb{Z}_{336}}(6)$ is generated from the vectors:

$$z'_1 = [1, 2, 0]^{\mathrm{t}} \qquad z'_2 = [1, 30, 36]^{\mathrm{t}} \qquad z'_3 = [1, 114, 228]^{\mathrm{t}}$$

Next, we determine the module of valid equalities modulo 336 as the set of solutions of the following homogeneous system of equations over $\mathbb{Z}_{336}$:

$$[\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2] \cdot \begin{bmatrix} 1 & 1 & 1 \\ 2 & 30 & 114 \\ 0 & 36 & 228 \end{bmatrix} = [0, 0, 0]$$

or, equivalently,

$$[\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 2 & 28 & 0 \\ 0 & 36 & 84 \end{bmatrix} = [0, 0, 0]$$

The module of solutions is generated by the two vectors:

$$[312, 12, 0], \quad [0, 0, 28]$$

This corresponds to the congruence equations:

$$312 \cdot \mathbf{x}_0 + 12 \cdot \mathbf{x}_1 \equiv 0\ [336] \qquad 28 \cdot \mathbf{x}_2 \equiv 0\ [336]$$ □

Remark that all calculations on vectors or matrices in the third phase of the algorithm are in fixed modular rings and thus do not incur extra swells of intermediate numbers. In particular, we can use the complexity bounds from figure 2, to estimate the number of arithmetic and generalized gcd computations. For the intraprocedural case, we thus obtain $\mathcal{O}(n \cdot k^3 \cdot \log(m(X)))$ operations. Since the length $\log(m(X))$ of $m(X)$ is polynomially bounded in n and k, we obtain a polynomial algorithm.

In the interprocedural case, the number of operations is bounded by $\mathcal{O}(n \cdot k^8 \cdot \log(m(X))$. The modulus $m(X)$, though, can have exponential length. Therefore, we obtain an exponential complexity bound as stated in assertion 2. □

A subtle point in the algorithm over $\mathbb{Q}$ or $\mathbb{Z}$ is the potential swell of intermediate numbers. Our complexity analysis reveals that the total run-time of the interprocedural algorithm is polynomial in the size n of the program, the number k of variables and $\log(\Delta_\mathcal{E})$. Thus, the algorithm performs well if $\Delta_\mathcal{E}$ is found to be moderate. At the expense of loss of precision, this can always be enforced. Assume we have given us a threshold Δ. Whenever a matrix A with entry $|A_{ij}| > \Delta$ is to be added to some fixpoint variable, we instead add matrices $A^{(0)}, A^{(1)}$ which are obtained from A by replacing the too large entry with 0 and d, respectively, for some divisor d of A_{ij} (e.g., 1).

6 Guards

The draw-back of the interprocedural analyses of section 4 is that conditional branching is abstracted by non-deterministic choice. A natural class of guards to be taken into account are *equality guards* of the form $g = 0$ for $g \equiv g_0 + g_1\mathbf{x}_1 + \ldots + g_k\mathbf{x}_k$. In presence of equality guards, however, already the problem of determining at a given program point whether a variable always equals 0 is undecidable [12]. This holds even in absence of procedures. Accordingly, any effective analysis of programs with guards must be approximate. Intraprocedurally, an approximative treatment of equality guards has been considered both by Karr for fields [9] and by Granger for $\mathbb{Z}$ [5]. In both cases, the effect of such a guard amounts to intersection of affine spaces. This idea also works for R-modules of extended states and any ring R:

$$[\![g = 0]\!]\, M \;=\; \langle M \cap \{[x_0, \ldots, x_k]^{\mathrm{t}} \mid x_0 = 1, \textstyle\sum_{j=0}^{k} g_j x_j = 0\}\rangle_{\mathsf{R}}$$

Computing the intersection can be reduced to solving a pair of linear equations: Assume $M = \langle G\rangle_{\mathsf{R}}$ where G is a finite set of generators. Let V denote a matrix containing the vectors of G as column vectors, let b denote the 0-th row of V. Then we obtain a system of generators for $[\![g = 0]\!]\, M$ by solving the system:

$$(g' V) \cdot \mathbf{y} = \mathbf{0} \qquad\qquad b \cdot \mathbf{y} = 1$$

for the row vectors $g' = [g_0, \ldots, g_k]$ and $b = [b_1, \ldots, b_q]$ and a column vector $\mathbf{y} = [\mathbf{y}_1, \ldots, \mathbf{y}_q]^{\mathrm{t}}$ of variables.

It is not obvious, though, how intersections can be lifted to the transformer level. Therefore, we suggest to *postpone* the decision taken at the guard. Instead of performing the intersection, we *accumulate* the value of the guard expression in an *indicator variable*. More precisely, assume that the edges with guards are numbered $k+1, \ldots, m$. Then we *instrument* the original program by introducing fresh variables $\mathbf{x}_{k+1}, \ldots, \mathbf{x}_m$, one for each guard. Initially, all these variables are assumed to have values 0. At the j-th guard $g = 0$, we place the assignment $\mathbf{x}_j := \mathbf{x}_j + g$. This corresponds to the matrix:

$$\left[\begin{array}{c|c|c|c} I_{k+1} & \multicolumn{3}{c}{0} \\ \hline 0 & I_{j-k-1} & 0 & 0 \\ \hline g_0 \ldots g_k & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & I_{m-j} \end{array}\right]$$

The extra values stored in the indicator variables are then used for an improved treatment of calls in the constraint system $\mathcal{C}^\sharp_{\mathsf{R}}$. As an invariant, we insist in $\mathcal{C}^\sharp_{\mathsf{R}}$ that all indicator variables have values 0, since this is the case for all program runs permitted by the guards. Thus the first constraint now reads:

$$[\mathcal{C}^\sharp_{\mathsf{R}}1] \quad \mathcal{C}^\sharp_{\mathsf{R}}(\mathbf{Main}) \sqsupseteq \mathsf{R}^{k+1} \times \{0^{m-k}\}$$

Accordingly, we modify the constraints for calls to:

$$[\mathcal{C}^\sharp_{\mathsf{R}}6] \quad \mathcal{C}^\sharp_{\mathsf{R}}(v) \supseteq \langle \mathcal{E}^\sharp_{\mathsf{R}}(q)(\mathcal{C}^\sharp_{\mathsf{R}}(u)) \cap (\{1\} \times \mathsf{R}^k \times \{0^{m-k}\})\rangle_{\mathsf{R}} \qquad \text{if edge } (u,v) \text{ calls } q$$

Thus, having applied the transformations from $\mathcal{E}^\sharp_{\mathsf{R}}(q)$, we select just those vectors from the result whose indicator variables all equal 0. These can be determined by solving an appropriate system of linear equations. Altogether, we obtain for every effective PIR R, an enhanced interprocedural analysis which deals with equality guards and conservatively extends the corresponding intraprocedural analysis. In particular, this technique extends the known methods for fields, for modular rings $\mathbb{Z}_m$ and also for $\mathbb{Z}$.

The separation of computing valid affine relations from computing valid modular relations as in section 5 also returns sound information. In presence of guards, however, the latter may result in an extra loss of precision. Consider, e.g., the guard $8 - \mathbf{x}_1 = 0$. Assume that before the guard, we have the extended state $x = [1, 3]^{\mathrm{t}}$. Since $8-3 = 5 \neq 0$, x does not pass the guard both in an analysis over $\mathbb{Q}$ and over $\mathbb{Z}$. Assume, however, that we perform the third stage of the algorithm modulo 5. Since x satisfies the guard modulo 5, x is propagated through the guard — thus incurring an extra loss in precision.

7 Conclusion

We have provided a general framework for analyzing interprocedurally valid affine relations over any principal ideal ring R. In absence of guards, the analyses could be shown to be complete, i.e., to infer all valid relations of the given form. In particular, our framework covers the known cases of fields $\mathbb{Q}$ or $\mathbb{Z}_p$ (p a prime) as well as modular rings $\mathbb{Z}_m$ (m composite) and also provides an interprocedural extension of Granger's analysis of linear congruence relations. In order to obtain a faster analysis, we then decomposed the latter analysis into several instances of our framework. This new algorithm has the advantage that its run-time complexity can be explicitly determined. In particular, its intraprocedural variant runs in polynomial time. Finally, we indicated how the proposed techniques can be enhanced to deal interprocedurally with equality guards.

A key issue in designing efficient algorithms has been to bound potential swell of intermediately occurring numbers. In case of linear congruence analysis, we therefore refrained from computing the $\mathbb{Z}$-affine abstraction of the collecting semantics directly. Instead, we resorted to computations over modular rings. Remark that instead of performing a separate analysis for each program point X of interest we could as well perform one joint analysis using the lcm of the moduli for the X. The disadvantage, however, is that lengths of occurring numbers could then again grow unacceptably.

In order to keep the presentation simple, we have considered parameterless procedures and global variables only. Local variables, call-by-value passing of parameters and return values can be handled along the lines of [13]. At the expense of an increase in the complexity, our methods can also be used to determine valid *polynomial* relations up to a fixed degree d [12, 13]. Further questions remain. It is still open whether it is possible to determine all valid *polynomial* relations — independent of a given degree bound. Also, it is desirable to design interprocedural analyses that deal precisely with further arithmetic operators.

References

1. G. Balakrishnan and T. W. Reps. Analyzing Memory Accesses in x86 Executables. In *Compiler Construction, 13th Int. Conf. (CC)*, 5–23. LNCS 2985. Springer-Verlag, 2004.
2. B. Blanchet, P. Cousot, R. Cousot, J. Feret, C. Mauborgue, D. Mormiaux, and X. Rival. A Static Analyzer for Large Safety-Critical Software. In *Int. ACM Conf. on Programming Language Design and Implementation (PLDI)*, 196–207, 2003.
3. N. Dor, M. Rodeh, and M. Sagiv. Cleanness Checking of String Manipulations in C Programs via Integer Analysis. In *8th Int. Static Analysis Symposium (SAS'01)*, 194–212. LNCS 2126, Springer Verlag, 2001.
4. P. Granger. Static Analysis of Arithmetical Congruences. *Int. J. of Computer Math.*, 165–190, 1989.
5. P. Granger. Static Analysis of Linear Congruence Equalities among Variables of a Program. In *Int. Joint Conf. on Theory and Practice of Software Development (TAPSOFT)*, 169–192. LNCS 493, Springer-Verlag, 1991.
6. S. Gulwani and G. Necula. Discovering Affine Equalities Using Random Interpretation. In *30th ACM Symp. on Principles of Programming Languages (POPL)*, 74–84, 2003.
7. S. Gulwani and G. Necula. Precise Interprocedural Analysis Using Random Interpretation. In *32th Ann. ACM Symp. on Principles of Programming Languages (POPL)*, 324–337, 2005.

8. J. Hafner and K. McCurley. Asymptotically Fast Triangularization of Matrices over Rings. *SIAM J. of Computing*, 20(6):1068–1083, 1991.
9. M. Karr. Affine Relationships Among Variables of a Program. *Acta Informatica*, 6:133–151, 1976.
10. S. Lang. *Algebra, Third Edition*. Pearson Education, Inc., 1993.
11. A. Miné. Relational Abstract Domains for the Detection of Floating-Point Run-Time Errors. In *European Conf. on Programming (ESOP)*, 3–17. LNCS 2986, Springer Verlag, 2004.
12. M. Müller-Olm and H. Seidl. A Note on Karr's Algorithm. In *31st Int. Coll. on Automata, Languages and Programming (ICALP)*, 1016–1028. Springer Verlag, LNCS 3142, 2004.
13. M. Müller-Olm and H. Seidl. Precise Interprocedural Analysis through Linear Algebra. In *31st ACM Symp. on Principles of Programming Languages (POPL)*, 330–341, 2004.
14. M. Müller-Olm and H. Seidl. Analysis of Modular Arithmetic. In *European Symposium on Programming (ESOP)*, 46–60. Springer Verlag, LNCS 3444, 2005.
15. A. Simon and A. King. Analyzing String Buffers in C. In *Algebraic Methodology and Software Technology, 9th Int. Conf. (AMAST)*, 365–379. LNCS 2422, Springer Verlag, 2002.
16. A. Storjohann. *Algorithms for Matrix Canonical Forms*. PhD thesis, ETH Zürich, Diss. ETH No. 13922, 2000.
17. A. Storjohann. *A Fast, Practical, and Deterministic Algorithm for Triangularizing Integer Matrices*. Tech. Rep. 255, ETH Zürich, 1996.
18. O. Zariski and P. Samuel. *Commutative Algebra, Vol. I*. Nostrand, Princeton, NJ, 1958.

A Proof of Theorem 1

The proof of statement (1) is a refinement of Granger's argument for computing a set of congruence relations characterizing $\langle G \rangle_{\mathbb{Z}}$. Let $V \in \mathbb{Z}^{r \cdot q}$ denote the matrix whose column vectors are the vectors from G. Then $x \in \langle G \rangle_{\mathbb{Z}}$ iff $V\, y = x$ for some (column) vector $y = [y_1, \ldots, y_q]^{\mathrm{t}} \in \mathbb{Z}^q$. Since V is linearly independent, we can find a unimodular matrix $T \in \mathbb{Z}^{r^2}$ such that $T \cdot V = \left[\frac{D}{0}\right]$ where D is an upper triangular $(q \times q)$-matrix and the product of the diagonal elements equals $\mathsf{det}(G)$ and thus divides m. In particular, $V\, y = x$ iff $(T \cdot V)\; y = T\; x$. In this matrix equation, the last $r - q$ rows constitute linear equations over $\mathbb{Z}$ whereas the first q rows can equivalently be formulated using linear equations modulo m. In order to see this, let d_i denote the i-th diagonal element of D and t_i the i-th row of T. Then the q-th row of the equation reads $d_q y_q = t_q \cdot x$ which is equivalent to the linear congruence equation $t_q \cdot x \equiv 0\ [d_q]$. By multiplying the remaining rows with d_q and subtracting suitable multiples of the q-th row, we can remove the q-th column of the remaining left-hand side of the equation system which leaves us with a similar problem where q has been decreased by one. Thus, we successively construct linear congruences with moduli $d_i \cdot \ldots \cdot d_q$ for $i = q$ down to $i = 1$. By scaling these equations with the products $p_i = \frac{m}{\mathsf{det}(G)} \cdot d_1 \cdot \ldots \cdot d_{i-1}$, we obtain equivalent congruences modulo m which together with the $m + 1 - q$ linear equations characterize all $x \in \langle G \rangle_{\mathbb{Z}}$.

Example 4. Consider the set $G = \{[2, 16, 34]^{\mathrm{t}}, [-2, -11, -24]^{\mathrm{t}}\}$. Let V denote the corresponding (3×2)-matrix of coefficients. Then there is unimodular matrix T with:

$$T = \begin{bmatrix} -7 & 1 & 0 \\ -8 & 1 & 0 \\ -1 & -2 & 1 \end{bmatrix} \qquad \text{and} \qquad V' = T \cdot V = \begin{bmatrix} 2 & 3 \\ 0 & 5 \\ 0 & 0 \end{bmatrix}$$

From the last row of T we thus can read off the linear equation:

$$-\mathbf{x}_0 - 2\mathbf{x}_1 + \mathbf{x}_2 = 0$$

The first two rows of the matrix equation $V' [y_1, y_2]^{\mathrm{t}} = T [x_0, x_1, x_2]^{\mathrm{t}}$ give us:

$$\begin{aligned} 2y_1 + 3y_2 &= -7x_0 + x_1 \\ 5y_2 &= -8x_0 + x_1 \end{aligned}$$

Subtracting three times the second equation from 5 times the first one gives us:

$$\begin{aligned} 10y_1 \qquad &= -11x_0 + 2x_1 \\ 5y_2 &= \ \ -8x_0 + \ \ x_1 \end{aligned}$$

This provides us with the following two congruence equations which together with the linear relation characterize the $\mathbb{Z}$-module generated by G:

$$\begin{aligned} -11\mathbf{x}_0 + 2\mathbf{x}_1 &\equiv 0\ [10] \\ -8\mathbf{x}_0 + \ \ \mathbf{x}_1 &\equiv 0\ \ [5] \end{aligned}$$

□

It remains to consider statement 2. The case $m' = 0$ is trivial. So let $m' > 0$. As the linear congruence equation $b' \cdot \mathbf{x} \equiv 0\ [m']$ is satisfied for a vector $v \in \mathbb{Z}^r$ iff $(d' \cdot b') \cdot \mathbf{x} \equiv 0\ [h]$ is satisfied for v (recall that $h = m' \cdot d'$), it suffices to show that $\langle E_0 \cup \{d \cdot b \mid b \in E_m\}\rangle_{\mathbb{Z}_h}$ characterizes the linear congruence relations valid for all vectors in G modulo h. Thus we show: $b' \cdot \mathbf{x} \equiv 0\ [h]$ is satisfied by all vectors in G iff $b' \in \langle E_0 \cup \{d \cdot b \mid b \in E_m\}\rangle_{\mathbb{Z}_h}$.

First of all, if $b' \in E_0$, then $b' \cdot x = 0$ and hence also $b' \cdot x = 0\ [h]$ for all $x \in G$. Moreover if $b \in E_m$ then $b \cdot x = 0\ [m]$ and hence $(d \cdot b) \cdot x = 0\ [h]$ for all $x \in G$ because $h = d \cdot m$. Thus, for any $b' \in \langle E_0 \cup \{d \cdot b \mid b \in E_m\}\rangle_{\mathbb{Z}_h}$, $b' \cdot \mathbf{x} \equiv 0\ [h]$ is satisfied by all vectors in G because validity of linear congruence relations is preserved by linear combinations. This shows the "if"-direction.

For the "only if"-direction, let again $V \in \mathbb{Z}^{r \cdot q}$ be the matrix whose columns are formed by the vectors from G. Note that for any $l > 0$ and $b \in \mathbb{Z}_l^r$, the linear congruence relation $b \cdot \mathbf{x} \equiv 0\ [l]$ holds for all $x \in G$ iff b is a solution of the following equation system E over $\mathbb{Z}_l$: $\mathbf{y} \cdot V = \mathbf{0}$. Similarly, for $b \in \mathbb{Z}^r$ the relation $b \cdot \mathbf{x} \equiv 0$ is satisfied by all vectors in G if b is a solution of the equation system E over $\mathbb{Z}$. Let b' be a solution of E over $\mathbb{Z}_h$. We need to show that $b' = b'_0 + d \cdot b'_1$ where b'_0 is a solution of E over $\mathbb{Z}$ and b'_1 is a solution over $\mathbb{Z}_m$. As the columns of V are linearly independent, we can construct a unimodular matrix T such that $V' = T \cdot V = \left[\frac{D}{0}\right]$ where D is upper triangular with diagonal elements $d_1, \ldots, d_q$ and $\mathsf{det}(G) = d_1 \cdot \ldots \cdot d_q$ divides m. Now we consider the homogeneous system E': $\mathbf{y} \cdot V' = \mathbf{0}$. The vector $b'' = b' \cdot T^{-1}$ is a solution of E' over $\mathbb{Z}_h$. We can write b'' in the form $b''_0 + b''_1$ where all components $i = 1, \ldots, q$ of b''_0 and all components $i = q+1, \ldots, r$ of b''_1 are 0. By inspecting E', we see that b''_0 is also a solution of E' over $\mathbb{Z}$ and b''_1 is also a solution over $\mathbb{Z}_h$. By induction for $i = q$ down to $i = 1$, we verify in addition that the i-th entry of b''_1 equals 0 modulo $d \cdot d_{i+1} \cdot \ldots \cdot d_q$. Thus, $b''_1 = d \cdot y$ for some $y \in \mathbb{Z}^r$. Since $d \cdot y \cdot V' = b''_1 \cdot V' \equiv \mathbf{0}\ [d \cdot m]$, we conclude that also $y \cdot V' \equiv \mathbf{0}\ [m]$. Therefore, y is a solution of the system $\mathbf{y} \cdot V' = \mathbf{0}$ over $\mathbb{Z}_m$. Now, we choose $b'_0 = b''_0 \cdot T$ and $b'_1 = y \cdot T$ such that $b' = b'' \cdot T = (b''_0 + d \cdot y) \cdot T = b'_0 + d \cdot b'_1$. Moreover, we have $b'_0 \cdot V = b''_0 \cdot T \cdot V = b''_0 \cdot V'$ such that b'_0 solves equation system E over $\mathbb{Z}$ because b''_0 solves E' over $\mathbb{Z}$. Similarly, $b'_1 \cdot V = b''_1 \cdot T \cdot V = b''_1 \cdot V'$ such that b'_1 solves E over $\mathbb{Z}_m$ because b''_1 solves E' over $\mathbb{Z}_m$. This completes the proof. □