University of Leeds
SCHOOL OF COMPUTING
RESEARCH REPORT SERIES
Report 2005.06

A Linear Domain for Analyzing the Distribution of Numerical Values[1]

by

Roberto Bagnara[2] & Katy Dobson[3] & Patricia M. Hill[3] & Matthew Mundell[3] & Enea Zaffanella[2]

December 2005

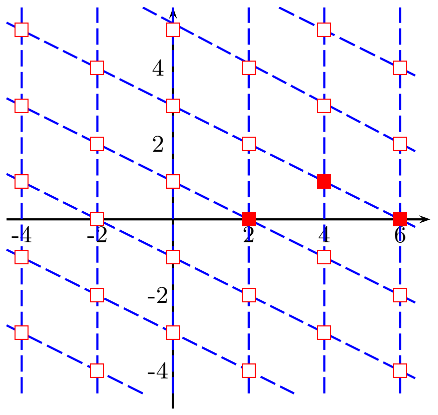
[1] This work has been partly supported by EPSRC project EP/C520726/1 "Numerical Domains for Software Analysis", by MIUR project "AIDA — Abstract Interpretation: Design and Applications," and by a Royal Society (UK) International Joint Project (ESEP) award.
[2] Department of Mathematics, University of Parma, Italy
{bagnara,zaffanella}@cs.unipr.it
[3] School of Computing, University of Leeds, UK
{katyd,hill,mattm}@comp.leeds.ac.uk

Abstract. This paper explores the abstract domain of *grids*, a domain that is able to represent sets of equally spaced points and hyperplanes over an n-dimensional vector space. Such a domain is useful for the static analysis of the patterns of distribution of the values program variables can take. Besides the bare abstract domain, we present a complete set of operations on grids that includes all that is necessary to define the abstract semantics and the widening operators required to compute it in a finite number of steps. The definition of the domain and its operations exploit well-known techniques from linear algebra as well as a dual representation that allows, among other things, for a concise and efficient implementation.

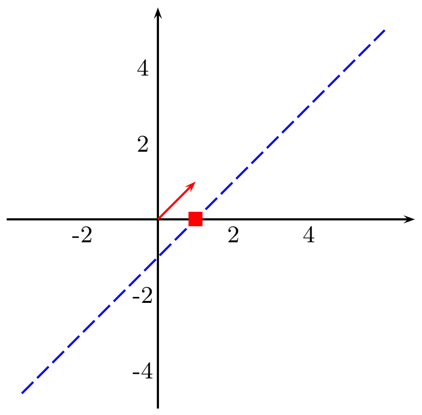
1 Introduction

The static analysis of numerical information about the values program variables can take is a challenging problem that has led to considerable research in both the theory and its practical realization. We distinguish between two kinds of numerical information: outer *limits* (or bounds within which the values must lie) and the pattern of *distribution* of these values. Both kinds of information have important applications. For example, in the field of automatic program verification, limit information is crucial to ensure that array accesses are within bounds, while distribution information is what is required to ensure that the external memory accesses obey the alignment restriction imposed by the host architecture. In the field of program optimization, limit information can be used to compile out various kinds of run-time tests, whereas distribution information enables several transformations for efficient parallel execution as well as optimizations that enhance cache behavior.

Both limit and distribution information often come in a *relational* form; for instance, the outer limits or the pattern of possible values of one variable may depend on the values of one or more other variables. Domains that can capture relational information are generally much more complex than domains that do not have this capability; in exchange they usually offer significantly more precision, often important for the overall performance of the client application. Relational limit information can be captured, among other possibilities, by means of *polyhedral domains*, that is, domains that represent regions of some n-dimensional vector space bounded by a finite set of hyperplanes [11]. While several polyhedral domains have been proposed and that of convex polyhedra, the most popular relational polyhedral domain, has been thoroughly researched and widely used, relational domains for representing the (linear) distribution of numerical values have been less well researched. For instance, previous investigations have hardly touched on the issue of widening operators; ignored the domain difference operation; and have failed to provide any support for a non-relational approximation. In addition, proposed algorithms are unnecessarily complex and standard mathematical procedures have not been utilized to their best advantage. Moreover, as far as we know and at the time of writing, there is no available implementation providing all the basic operations needed



(a) The grid $\mathcal{L}$



(b) The grid $\mathcal{L}'$

Fig. 1. Two grids in $\mathbb{R}^2$

by a relational abstract domain for distribution information. This is in spite of the fact that previous research has shown that a knowledge about the (discrete) distribution of numerical information, especially when combined with that of the limit information, can significantly improve the quality of the analysis results [1]. The purpose of this paper is to fill this gap: we will present a complete relational domain of *grids* that captures numerical distribution information and briefly describe its implementation within the Parma Polyhedra Library (PPL) [5, 6].

1.1 Grids in a Nutshell

The diagrams in Figure 1 illustrate alternative approaches to the description of any grid; either by means of a finite set of congruence relations that all the grid points must satisfy (given by the dashed lines) or by means of a finite set of generating vectors used for constructing the grid points and lines (given by the filled squares and the arrow).

Consider first Figure 1(a). The set of points marked by squares at the intersection of the dashed lines represent the grid $\mathcal{L}$. This denotes the distribution of possible values of integer variables x and y resulting from the execution of the following program fragment (based on an example in [11]) for any value of `m`:

```
x := 2; y := 0;
for i := 1 to m
  if ... then
    x := x + 4
  else
    x := x + 2; y := y + 1
  endif
endfor
```

Note that the vertical dashed lines represent the set of points satisfying the congruence relation $x = 0 \pmod 2$ while the sloping lines represent the set of points satisfying $x + 2y = 2 \pmod 4$. The set

$$\mathcal{C} = \bigl\{x = 0 \pmod 2, \quad x + 2y = 2 \pmod 4\bigr\}$$

is called a *congruence system* and said to *describe* $\mathcal{L}$. The filled squares in Figure 1(a) represent the points

$$\boldsymbol{p}_1 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \ \boldsymbol{p}_2 = \begin{pmatrix} 6 \\ 0 \end{pmatrix} \text{ and } \boldsymbol{p}_3 = \begin{pmatrix} 4 \\ 1 \end{pmatrix}$$

while all the squares (both filled and unfilled) in the diagram mark position vectors $\boldsymbol{v} = \pi_1\boldsymbol{p}_1 + \pi_2\boldsymbol{p}_2 + \pi_3\boldsymbol{p}_3$, where $\pi_1, \pi_2, \pi_3 \in \mathbb{Z}$ and $\pi_1 + \pi_2 + \pi_3 = 1$. The set of points $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3\}$ is said to *generate* $\mathcal{L}$. Some of these generating points can be replaced by *parameters* that give the direction and spacing for the neighbouring points. Specifically, by subtracting the point $\boldsymbol{p}_1$ from each of the other two generating points $\boldsymbol{p}_2, \boldsymbol{p}_3$ we obtain

$$\boldsymbol{q}_2 = \begin{pmatrix} 4 \\ 0 \end{pmatrix} \text{ and } \boldsymbol{q}_3 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}.$$

It follows that each point $\boldsymbol{v} \in \mathcal{L}$ can be written as $\boldsymbol{v} = \boldsymbol{p}_1 + \pi_2\boldsymbol{q}_2 + \pi_3\boldsymbol{q}_3$ for some $\pi_2, \pi_3 \in \mathbb{Z}$; the set $Q = \{\boldsymbol{q}_2, \boldsymbol{q}_3\}$ is called a *parameter set* for $\mathcal{L}$.

Consider next the grid $\mathcal{L}'$ illustrated by the dashed line in Figure 1(b) which is all the points that satisfy the equality $x = y + 1$. This denotes the distribution of values of the real variables x and y after an assignment `x := y + 1`, assuming that nothing is known about the value of y. In this case, as we regard equalities as congruences modulo 0, we say that the congruence system

$$\mathcal{C}' = \{x - y = 1\}$$

describes $\mathcal{L}'$. Observe also that the grid $\mathcal{L}'$ consists of all points that can be obtained as $\lambda\boldsymbol{\ell} + \boldsymbol{p}'$, for any $\lambda \in \mathbb{R}$, where

$$\boldsymbol{\ell} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \text{ and } \boldsymbol{p}' = \begin{pmatrix} 1 \\ 0 \end{pmatrix};$$

the vector $\boldsymbol{\ell}$ defines the direction of the *line* and the vector $\boldsymbol{p}'$ its position (illustrated in Figure 1(b) by the arrow and the filled square, respectively).

From what we have just seen, any grid can be represented both by a congruence system and by a *generator system*. The latter may consist of three components: a set of lines, a set of parameters and a set of points. For instance, the triples $\mathcal{G}_1 = \bigl(\varnothing, \varnothing, P\bigr)$ and $\mathcal{G}_2 = \bigl(\varnothing, Q, \{\boldsymbol{p}_1\}\bigr)$ are both generator systems for $\mathcal{L}$ while the triple $\mathcal{G}' = \bigl(\{\boldsymbol{\ell}\}, \varnothing, \{\boldsymbol{p}'\}\bigr)$ is a generator system for $\mathcal{L}'$.

1.2 Implementing Grids

The congruence and generator systems for representing elements of the domain of rational grids can be seen to have close parallels with the constraint and generator systems for describing convex polyhedra. In this paper, we develop and exploit this analogy. First, as for the systems representing convex polyhedra, by adding an extra dimension, the representations can be made homogeneous [27, 29]. Secondly, as a direct result of [35, Section 4.4], the homogenized congruence and generator systems are dual and, hence, equivalent representations for the domain of rational grids. Thus the congruence and generator systems for representing grids form the two components of a double description method for the grid domain that is very similar to that of the double description method for convex polyhedra [25].

Just as for the domain of convex polyhedra, for a double description method for grids to be viable, we require algorithms that can convert between a rational grid's generator and congruence systems. In fact, for this domain, the standard techniques for matrix inversion can be used as the basis for the conversion algorithms. Moreover, since a congruence or generator system may contain some redundancies, or, even when there are no explicit redundancies, there may exists smaller a set of congruences or generators that generates or describes the same grid, we require algorithms for minimizing the representations. As before, there are standard matrix algorithms (for instance, the Hermite normal form algorithm [28, 35]) that can be used as a basis for minimization algorithms. Note that the minimal representations are needed not just to economize on memory space and subsequent computation time, but also for the conversion algorithms mentioned above.

1.3 Related Work

As far as we know, Granger provided the first account of an analysis for congruence information; in [14] it is shown how a simple integral and non-relational grid domain could be used in a static analyzer. In this domain, an element is characterized as the set of all integers that are congruent to c modulo m where c and m are integers; that is, the set of all integers x where, for some $k \in \mathbb{Z}$, $x = c + km$; which does, in fact, immediately provide an equivalent one-dimensional generator system consisting of a point (c) and parameter (m). One particularly useful aspect of this work is that it indicates some practical applications of the domain that carry over to the more complex domain considered here. For instance, it is shown how an analysis using the grid domain can obtain more precise information than an analysis using a constant propagation domain or a sign domain. It is then indicated how the results may be applied for automatic vectorization. Independently, Larsen et al. [22] have also developed a static analyzer over a non-relational grid domain; this analyzer is specifically designed to detect when the dynamic memory addresses are congruent with respect to a given modulus (the actual modulus being dependent on the application). It is shown how this information may be used to construct a comprehensive set of program transformations that can lead to energy savings on low-power architectures and performance improvements on multimedia processors.

Note that, prior to the current stream of work on congruence relations, Karr [21] had pioneered the consideration of a linear *relational* equality domain for program analysis. More recently, Granger, building on his previous work for non-relational congruences [14], generalized Karr's work to linear congruence relations [16] (see also [15]). This paper focuses on an integral domain, but as indicated in the subsequent paper by the same author [17], this work easily generalizes to the domain of rational grids. Two representations for the domain are described, sets of congruences and sets of generators. In [16], it is shown how standard matrix algorithms [36] can be used to convert from a generator system to a congruence system; a more complex algorithm is also provided for conversion from a congruence system to a generator system. Operators for comparing grids and computing the greatest lower and least upper bounds are also described in [16].

Most previous work (such as that in [16, 17]) as well as the work in this paper assumes fully relational congruence systems for representing grids. In contrast, by applying a generic technique, Miné [24] shows how to construct a weakly relational *zone-congruence* domain (that is, a domain that only allows congruences that have the form $x - y = a \pmod{b}$ where a and b are rationals) from the non-relational congruence domain described in [14].

Müller-Olm and Seidl [27] have recently improved on the work of Granger [16, 17]; one aspect of the improvement is obtained by recognizing, as we do here, that the inhomogeneous representations may be transformed to homogeneous ones by adding an extra dimension. However, we note that [27] is focused, not on the definition of the domain itself, but on the definition of an abstract interpretation of a constraint system that can characterize the concrete program semantics, particularly in the case when there are procedural calls. Thus the main contributions of [27] are largely orthogonal to the contributions here. Note also that as stated in [27], the framework described there subsumes analysis described in previous work by the same authors; in particular, it subsumes the analysis described in [26] designed specifically for analyzing certain integer arithmetic computations that are modulo 2^w where, for example, w is 32 or 64.

Following a completely independent stream of research and publications, Ancourt [1] considered a domain of grids (i.e., lattices) combined with the domain of convex polyhedra; that is the domain of *$\mathcal{Z}$-polyhedra* defined as "the intersections of polyhedra and integral lattices". We are primarily interested here in the "integral lattices" component which may be seen as a subdomain of the domain of rational grids where the grid points are all integral vectors and is full-dimensional. In all the papers concerning the $\mathcal{Z}$-polyhedral domain [1, 29, 31, 32], the representation of these integral lattices is similar to that of the generator representation given in Section 3. All the operations on $\mathcal{Z}$-polyhedra (and therefore the lattices) require canonic representations; hence Quinton et al. [31, 32] defines a canonical form for these lattices with a method for its computation. We note that the algorithm for computing the canonic form has complexity $\mathrm{O}\big(n^4\big)$, where

n is the number of dimensions of the vector space. Other operations provided for lattices are those of intersection, affine image and affine preimage. As there is no congruence representation, the intersection of two lattices is computed directly from the generator representations [1]; a refined version of this method is provided in [31] which we note that, as for computing the canonic form, it has complexity $\mathrm{O}(n^4)$. On the other hand, the operations of grid join '$\oplus$' and grid difference '$\ominus$' (as defined in Section 4) are not considered. Instead the union operator takes two lattices $\mathcal{L}_1$ and $\mathcal{L}_2$ and returns the set $\{\mathcal{L}_1, \mathcal{L}_2\}$ of the two unless one (say $\mathcal{L}_1$) is contained in the other, in which case they will return the larger $\mathcal{L}_2$. Moreover, the difference operation is also exact and, in general, returns a set of lattices.

The domain of integral lattices and operations on them has been implemented in PolyLib[23] following the methodology described in [31, 32]. Thus several of the main operations on lattices, such as union, return *sets* of lattices while others are concerned with manipulating and simplifying these sets. We note in particular that, as explained by Nookala and Risset [29], all the operations are implemented using a homogeneous representation. This is obtained, as we do here, by adding an extra dimension to the inhomogeneous representation that is input to the system via the user interface.

1.4 Contributions

The paper provides a full account of the relational domain of *grids* including its alternative representations, operations required for static analysis, and key algorithms needed for its implementation. Although some of this is already in the literature, it is widely dispersed and has never before been assembled into a single description. There are also several specific contributions.

- Although the Gaussian elimination and Hermite normal form algorithms form the basis of the proposed conversion and minimization algorithms for grids, we have had to extend them so as to apply to any grid and its representations as well as adapt them so as to be suitable for a real, efficient implementation. Therefore we outline the actual algorithms we use in our implementation; some of which are shown to have complexities strictly better than that of previous proposals [16, 27].
- By taking the union of the generator systems (resp., congruence systems) representing two grids we can compute their least upper bound (resp., the greatest lower bound); therefore, since we have reduced the complexity of the algorithms for minimizing and converting between representations, our implementations for these operations also have smaller complexity than that of previous proposals [16].
- We provide an algorithm for computing the smallest grid containing the difference of two grids. Observe that there are no previous proposals for this operator. The only papers that considers the computation of the difference of two grids are those concerning the $\mathcal{Z}$-polyhedral domain [1, 29, 31, 32]; here, as mentioned above, the difference operation returns the exact difference of the grids which has to be represented by a set of grids.

- One major difference between the domains of rational and integral grids is that the rational grid domain $\mathbb{G}_n$ does not satisfy the ascending chain condition. In [17], a widening is proposed for non-relational rational grids and it is briefly stated that it might be generalized to the case of rational grids but without any further details; how this might be done is unclear. Here we provide a practical widening operator for rational grids that, when restricted to the non-relational grid, does correspond to that proposed in [17].
- In order to dynamically manage the complexity/precision tradeoff of program analysis, the analyzer should be able to dynamically switch (in response to timeouts or driven by heuristics) to a simpler analysis domain. We provide a non-relational approximation for an arbitrary grid that can be implemented by means of an interval domain together with algorithms for conversion from a grid to the approximation and vice-versa.

1.5 Plan of the Paper

The rest of the paper is structured as follows. The required notations and preliminary concepts are given in Section 2. Section 3 introduces a grid together with its congruence and generator representations; Section 4 describes the basic operators on grids needed for program analysis; Section 5 provides the main algorithms needed to support the double description; Section 6 introduces the grid widening operation and its implementation; Section 7 gives some examples of grids applications. We conclude the main body of the paper in Section 8. Appendix A contains the proofs of all the stated results.

2 Preliminaries

The *cardinality* of a set S is denoted by $\# S$. The set of integers is denoted by $\mathbb{Z}$, rationals by $\mathbb{Q}$ and reals by $\mathbb{R}$. For any $a, b \in \mathbb{R}$ where $a \neq 0$, we say *a divides b*, denoted by $a|b$, if, for some $m \in \mathbb{Z}$, $am = b$. The complexities we give for the different algorithms assume a unit cost for every arithmetic operation; we take the computation of the greatest common divisor of a pair of numbers $a, b \in \mathbb{R}$ to be a single operation. Given sets X, Y and any relation $R \subseteq X \times Y$, the *image* for R on a subset A of X is $\{\, y \in Y \mid \exists x \in A \,.\, (x, y) \in R \,\}$, and the *preimage* for R on a subset B of Y is $\{\, x \in X \mid \exists y \in B \,.\, (x, y) \in R \,\}$.

2.1 Vectors and Matrices

For each $i \in \{1, \dots, n\}$, v_i denotes the i-th component of the (column) vector $\boldsymbol{v} \in \mathbb{R}^n$. The empty vector (in $\mathbb{R}^0$) is denoted by ϵ; a vector that has all its elements equal to zero is denoted by $\mathbf{0}$; and, for $1 \leq i \leq n$, the notation $\boldsymbol{e}_i$ denotes the vector in $\mathbb{R}^n$ with 1 in the i-th position and zeroes in every other position. For $\boldsymbol{v} \in \mathbb{R}^n$, $\operatorname{piv}_{<}(\boldsymbol{v})$ denotes the maximum index i such that $v_i \neq 0$; if $\boldsymbol{v} = \mathbf{0}$, we define $\operatorname{piv}_{<}(\boldsymbol{v}) := 0$. Similarly, $\operatorname{piv}_{>}(\boldsymbol{v})$ denotes the minimum index i such that $v_i \neq 0$; if $\boldsymbol{v} = \mathbf{0}$, we define $\operatorname{piv}_{>}(\boldsymbol{v}) := n+1$. Any vector $\boldsymbol{v} \in \mathbb{R}^n$ is also a matrix in

$\mathbb{R}^{n\times 1}$ so that it can be manipulated with the usual matrix operations of addition and multiplication, both by a scalar and by another matrix. On the other hand, it is often convenient to consider a matrix $H = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_m) \in \mathbb{R}^{n\times m}$ as a finite set of vectors $\{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_m\} \subseteq \mathbb{R}^n$. For each $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$, the ij-th component of a matrix $H \in \mathbb{R}^{n\times m}$ is denoted by H_{ij} and the i-th row by H_i. The *transposition* of a matrix H is denoted by H^{T}. The *scalar product* of $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^n$, denoted $\langle \boldsymbol{v}, \boldsymbol{w}\rangle$, is the real number $\boldsymbol{v}^{\mathrm{T}}\boldsymbol{w} = \sum_{i=1}^{n} v_i w_i$.

Vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m \in \mathbb{R}^n$ are said to be *affinely independent* if, for all $\boldsymbol{\lambda} \in \mathbb{R}^m$, $\boldsymbol{\lambda} = \boldsymbol{0}$ is the only solution of the set of equations $\bigl\{\sum_{i=1}^{m} \lambda_i \boldsymbol{v}_i = \boldsymbol{0},\ \sum_{i=1}^{m} \lambda_i = 0\bigr\}$.

2.2 Integer Combinations

Let $S = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k\} \subseteq \mathbb{R}^n$ be a set of k vectors. For all scalars $\lambda_1, \ldots, \lambda_k \in \mathbb{R}$, the vector $\boldsymbol{v} = \sum_{j=1}^{k} \lambda_j \boldsymbol{v}_j$ is said to be a *linear* combination of the vectors in S. Such a combination is said to be

- an *affine* combination, if $\sum_{j=1}^{k} \lambda_j = 1$;
- an *integral* combination, if $\lambda_1, \ldots, \lambda_k \in \mathbb{Z}$;
- an *integral affine* combination, if it is both integral and affine.

We denote by affine.hull(S) (resp., int.hull(S), int.affine.hull(S)) the set of all the affine (resp., integer, integer affine) combinations of the vectors in S.

2.3 Congruences and Congruence Relations

For any $a, b, f \in \mathbb{R}$, $a \equiv_f b$ denotes the *congruence* $\exists \mu \in \mathbb{Z} \mathrel{.} a - b = \mu f$. In the case that $f = 0$, the congruence denotes the equality $a = b$. Let $\mathbb{S}$ be either $\mathbb{Q}$ or $\mathbb{R}$. For each vector $\boldsymbol{a} \in \mathbb{S}^n$ and scalars $b, f \in \mathbb{S}$, the notation $\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_f b$ stands for the *linear congruence relation in* $\mathbb{S}^n$ defined by the set of vectors $\bigl\{ \boldsymbol{v} \in \mathbb{R}^n \bigm| \exists \mu \in \mathbb{Z} \mathrel{.} \langle \boldsymbol{a}, \boldsymbol{v}\rangle = b + \mu f \bigr\}$; f is called the *frequency* and b the *base value* of the relation; when $f = 0$, the congruence relation denotes the equality $\langle \boldsymbol{a}, \boldsymbol{x}\rangle = b$; when $f \neq 0$, the congruence relation is said to be *proper*. Thus, provided $\boldsymbol{a} \neq \boldsymbol{0}$, the congruence relation $\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_f b$ defines the set of affine hyperplanes $\bigl\{ \bigl(\langle \boldsymbol{a}, \boldsymbol{x}\rangle = b + \mu f\bigr) \bigm| \mu \in \mathbb{Z} \bigr\}$. The congruence $\langle \boldsymbol{0}, \boldsymbol{x}\rangle \equiv_f b$ defines the universe $\mathbb{R}^n$ if $b \equiv_f 0$, and the emptyset, otherwise. We will assume that in such a congruence (when $\boldsymbol{a} = \boldsymbol{0}$) we have $b \neq 0$. Any vector that satisfies one of the equalities $\langle \boldsymbol{a}, \boldsymbol{x}\rangle = b + \mu f$ for any $\mu \in \mathbb{Z}$ is said to *satisfy* the congruence relation $\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_f b$. We do not distinguish between syntactically different congruences defining the same set of vectors in $\mathbb{S}^n$ so that, e.g., $x \equiv_1 2$ and $2x \equiv_2 4$ are considered to be the same congruence.

We extend the pivot notation to congruences as follows. If $\beta = \bigl(\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_f a_0\bigr)$ and $\mathrm{piv}_<(\boldsymbol{a}) = k$, then $\mathrm{piv}_<(\beta) := k$. We say that congruence β is *pivot equivalent* to $\gamma = \bigl(\langle \boldsymbol{c}, \boldsymbol{x}\rangle \equiv_g c_0\bigr)$, and write $\beta \Uparrow \gamma$, if $\mathrm{piv}_<(\gamma) = \mathrm{piv}_<(\beta) = k$ and $g a_k = f c_k$. Observe that, as $a_k, c_k \neq 0$, this means that β and γ are either both equalities or both proper congruences.

3 Rational Grids

In this section we introduce rational grids and their representation. Note that previously the terminology 'lattice' instead of 'grid' has been used for elements of domains similar to the grids domain described here. However, to avoid any confusion with the use of 'lattice' in its set-theoretic context (which is particularly relevant when working in the framework of abstract interpretation), we prefer the terminology 'grid' for sets of vectors that can represent discrete and linearly repetitive information. Moreover, as $\mathcal{G}$ is used to denote a generator system, we use $\mathcal{L}$ to denote a rational grid.

3.1 The Grid Domain and the Congruence Representation

A *congruence system in* $\mathbb{Q}^n$ is a finite set of congruence relations $\mathcal{C}$ in $\mathbb{Q}^n$. As we do not distinguish between syntactically different congruences defining the same set of vectors, we can assume that all proper congruences in $\mathcal{C}$ have modulus 1.

Definition 1. *The set of vectors $\mathcal{L}$ is a* rational grid *in $\mathbb{R}^n$* described by *a congruence system $\mathcal{C}$ in $\mathbb{Q}^n$ if and only if $\mathcal{L}$ is the set of points in $\mathbb{R}^n$ that satisfy all the congruences in $\mathcal{C}$. We also say that $\mathcal{C}$* is a congruence system for $\mathcal{L}$ *and write $\mathcal{L} = \mathrm{gcon}(\mathcal{C})$.*

If $\mathrm{gcon}(\mathcal{C}) = \varnothing$, then we say that $\mathcal{C}$ is *inconsistent*. For example, the congruence systems $\bigl\{\langle \mathbf{0}, \boldsymbol{x}\rangle \equiv_0 1\bigr\}$ and $\bigl\{\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_2 0, \langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_2 1\bigr\}$, for any $\boldsymbol{a} \in \mathbb{Q}^n$, both describe the empty grid in $\mathbb{R}^n$. In fact, the first congruence system requires that $0 = 1$, while the second one requires that the value of an expression is both even and odd, so that they are both inconsistent.

The *grid domain* $\mathbb{G}_n$ is the set of all rational grids in $\mathbb{R}^n$. When ordering grids by the set inclusion relation, the empty set $\varnothing$ and the vector space $\mathbb{R}^n$ (which is described by the empty set of congruence relations) are, respectively, the smallest and the biggest elements of $\mathbb{G}_n$. The vector space $\mathbb{R}^n$ is also called the *universe* grid. In set theoretical terms, $\mathbb{G}_n$ is a *lattice* under set inclusion.

The *space dimension* of a grid $\mathcal{L} \in \mathbb{G}_n$ is the dimension $n \in \mathbb{N}$ of the corresponding vector space $\mathbb{R}^n$. If the maximum number of affinely independent points in $\mathcal{L}$ is $k+1$, then $\dim(\mathcal{L}) = k$ denotes the *affine dimension* of $\mathcal{L}$. The affine dimension of an empty grid is defined to be 0. Thus we have $0 \leq \dim(\mathcal{L}) \leq n$.

Let $\mathcal{C}$ be a congruence system and $\mathcal{L} = \mathrm{gcon}(\mathcal{C})$. Suppose also that the congruence relation $\beta = \bigl(\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_f b\bigr)$ is such that $\mathcal{L}_\beta = \mathrm{gcon}\bigl(\{\beta\}\bigr)$. We say that

- $\mathcal{L}$ *is disjoint* from β if $\mathcal{L} \cap \mathcal{L}_\beta = \varnothing$; that is, adding β to $\mathcal{C}$ gives us the empty grid.
- $\mathcal{L}$ *strictly intersects* β if $\mathcal{L} \cap \mathcal{L}_\beta \neq \varnothing$ and $\mathcal{L} \cap \mathcal{L}_\beta \subset \mathcal{L}$; that is, adding β to $\mathcal{C}$ gives us a non-empty grid strictly smaller than $\mathcal{L}$.
- $\mathcal{L}$ *is included* in β if $\mathcal{L} \subseteq \mathcal{L}_\beta$; that is, adding β to $\mathcal{C}$ leaves $\mathcal{L}$ unchanged.

Many algorithms used in the implementation require the congruence systems to have a minimal number of elements.

Definition 2. *Suppose $\mathcal{C}$ is a congruence system in $\mathbb{Q}^n$. Then we say that $\mathcal{C}$ is in* minimal form *if either $\mathcal{C} = \{\langle \mathbf{0}, \boldsymbol{x}\rangle \equiv_0 1\}$ or $\mathcal{C}$ is consistent and, for each congruence $\beta = \bigl(\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_f b\bigr) \in \mathcal{C}$, the following hold:*

1. *if $\operatorname{piv}_<(\beta) = k$, then $k > 0$ and $a_k > 0$;*
2. *for all $\beta' \in \mathcal{C} \setminus \{\beta\}$, $\operatorname{piv}_<(\boldsymbol{\beta'}) \neq \operatorname{piv}_<(\boldsymbol{\beta})$.*

Proposition 1. *Let $\mathcal{C}$ be a congruence system in $\mathbb{Q}^n$. Then there exists an algorithm for finding a congruence system $\mathcal{C}'$ in minimal form such that $\operatorname{gcon}(\mathcal{C}) = \operatorname{gcon}(\mathcal{C}')$. Letting $m = \#\mathcal{C}$, the complexity of the algorithm is $\mathrm{O}\bigl(n^2 m\bigr)$.*

The pivot equivalence relation between congruences can be extended to congruence systems. That is, we say that congruence systems $\mathcal{C}_1$ and $\mathcal{C}_2$ are *pivot equivalent* if, for each $\beta \in \mathcal{C}_1$, there exists $\gamma \in \mathcal{C}_2$ such that $\beta \Uparrow \gamma$ and, for each $\gamma \in \mathcal{C}_2$, there exists $\beta \in \mathcal{C}_1$ such that $\gamma \Uparrow \beta$. This equivalence relation provides an easy syntactic check to establish if two grids are equal, given that it is known that one is a subset of the other.

Proposition 2. *Let $\mathcal{L}_1 = \operatorname{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 = \operatorname{gcon}(\mathcal{C}_2)$ be non-empty grids in $\mathbb{G}_n$ such that $\mathcal{L}_1 \subseteq \mathcal{L}_2$. If $\mathcal{C}_1$ and $\mathcal{C}_2$ are pivot equivalent congruence systems in minimal form, then $\mathcal{L}_1 = \mathcal{L}_2$.*

3.2 The Generator Representation

Let $\mathcal{L}$ be a grid in $\mathbb{G}_n$. Then

- a vector $\boldsymbol{p} \in \mathcal{L}$ is called a *point* of $\mathcal{L}$;
- a vector $\boldsymbol{q} \in \mathbb{R}^n$, where $\boldsymbol{q} \neq \mathbf{0}$, is called a *parameter* of $\mathcal{L}$ if $\mathcal{L} \neq \varnothing$ and $\boldsymbol{p} + \mu\boldsymbol{q} \in \mathcal{L}$, for all points $\boldsymbol{p} \in \mathcal{L}$ and all $\mu \in \mathbb{Z}$;
- a vector $\boldsymbol{l} \in \mathbb{R}^n$, where $\boldsymbol{l} \neq \mathbf{0}$, is called a *line* of $\mathcal{L}$ if $\mathcal{L} \neq \varnothing$ and $\boldsymbol{p} + \lambda\boldsymbol{l} \in \mathcal{L}$, for all points $\boldsymbol{p} \in \mathcal{L}$ and all $\lambda \in \mathbb{R}$.

We can generate any rational grid in $\mathbb{G}_n$ from a finite subset of its points, parameters and lines; each point in a grid is obtained by adding a linear combination of its generating lines to an integral combination of its parameters and an integral affine combination of its generating points.

If L, Q and P are each finite subsets of $\mathbb{Q}^n$ and

$$\mathcal{L} := \operatorname{linear.hull}(L) + \operatorname{int.hull}(Q) + \operatorname{int.affine.hull}(P) \tag{1}$$

where the symbol '+' denotes the Minkowski's sum,[2] then $\mathcal{L} \in \mathbb{G}_n$ is a rational grid [15, Theorem 48]. The 3-tuple (L, Q, P) is said to be a *generator system* for $\mathcal{L}$ and we write $\mathcal{L} = \operatorname{ggen}(L, Q, P)$. $\mathcal{L}$ is said to *subsume* a generator g if adding g to any generator system representing $\mathcal{L}$ does not change $\mathcal{L}$.

Note that the grid $\mathcal{L} = \operatorname{ggen}(L, Q, P) = \varnothing$ if and only if the set of points $P = \varnothing$. If $P \neq \varnothing$, then $\mathcal{L} = \operatorname{ggen}(L, \varnothing, Q_{\boldsymbol{p}} \cup P)$ where, for some $\boldsymbol{p} \in P$, $Q_{\boldsymbol{p}} = \{\, \boldsymbol{p} + \boldsymbol{q} \in \mathbb{R}^n \mid \boldsymbol{q} \in Q \,\}$.

As for congruence systems, for many procedures in the implementation, it is useful if the generator systems have a minimal number of elements.

[2] This is defined, for each $S, T \subseteq \mathbb{R}^n$, by $S + T := \{\, \boldsymbol{s} + \boldsymbol{t} \in \mathbb{R}^n \mid \boldsymbol{s} \in S, \boldsymbol{t} \in T \,\}$.

Definition 3. *Suppose $\mathcal{G} = (L, Q, P)$ is a generator system in $\mathbb{Q}^n$. Then we say that $\mathcal{G}$ is in* minimal form *if either $L = Q = P = \varnothing$ or $\# P = 1$ and, for each generator $\boldsymbol{v} \in L \cup Q$, the following hold:*

1. *if $\operatorname{piv}_>(\boldsymbol{v}) = k$, then $v_k > 0$;*
2. *for all $\boldsymbol{v}' \in (L \cup Q) \setminus \{\boldsymbol{v}\}$, $\operatorname{piv}_>(\boldsymbol{v}') \neq \operatorname{piv}_>(\boldsymbol{v})$.*

Proposition 3. *Let $\mathcal{G} = (L, Q, P)$ be a generator system in $\mathbb{Q}^n$. Then there exists an algorithm for finding a generator system $\mathcal{G}'$ in minimal form such that $\operatorname{ggen}(\mathcal{G}') = \operatorname{ggen}(\mathcal{G})$. Letting $m = \# L + \# Q$, the complexity of the algorithm is $\mathrm{O}\bigl(n^2 m\bigr)$.*

3.3 Double Description

We have shown that any grid $\mathcal{L}$ can be described by using a congruence system $\mathcal{C}$ and also generated by a generator system $\mathcal{G}$. For the same reasons as for the polyhedral domain, it is useful to represent the grid $\mathcal{L}$ by the *double description* $(\mathcal{C}, \mathcal{G})$. In order to maintain and exploit such a view of a grid, we will use a *double description method* for grids which is a collection of results very similar to those for convex polyhedra showing that, given one kind of representation, there are algorithms for computing a representation of the other kind and for minimizing both representations. As for convex polyhedra, having easy access to both the representations is assumed in the implementation of many operators such as, for example, intersection and grid join described in Section 4.

Suppose we have a double description $\bigl(\mathcal{C}, \mathcal{G}\bigr)$ of a grid $\mathcal{L} \in \mathbb{G}_n$, where both $\mathcal{C}$ and $\mathcal{G}$ are in minimal form. Then, it follows from the definition of minimal form that $\# \mathcal{C} \leq n + 1$ and $\# L + \# Q \leq n$. In fact, we have a stronger result.

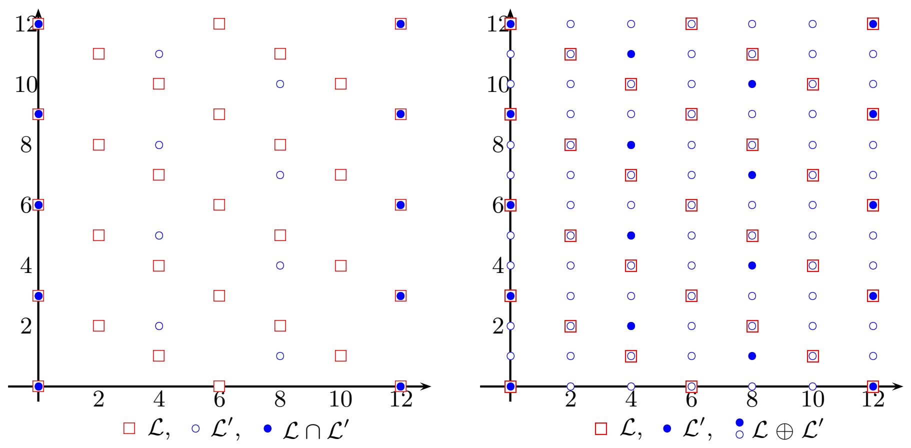
Proposition 4. *Let $(\mathcal{C}, \mathcal{G})$ be a double description where both $\mathcal{C}$ and $\mathcal{G}$ are in minimal form. Letting $\mathcal{C} = \mathcal{E} \cup \mathcal{F}$ where $\mathcal{E}$ and $\mathcal{F}$ are sets of equalities and proper congruences, respectively, and $\mathcal{G} = (L, Q, P)$, then*

$$\# \mathcal{F} = \# Q = n - \# L - \# \mathcal{E}.$$

Example 1. Consider the grids $\mathcal{L}$ and $\mathcal{L}'$ in Figure 1. The congruence systems $\mathcal{C}$ and $\mathcal{C}'$ are in minimal form and the generator systems $\mathcal{G}_2$ and $\mathcal{G}'$ are also in minimal form; however, $\mathcal{G}_1$ is not in minimal form as it contains more than one point. Furthermore, for $i = 1$, 2, the pairs $(\mathcal{C}, \mathcal{G}_i)$ are double descriptions for $\mathcal{L}$ while $(\mathcal{C}', \mathcal{G}')$ is a double description for $\mathcal{L}'$.

4 Operations on Rational Grids

In this section we describe some operations on rational grids that are analogous to ones that are already provided for the domain of convex polyhedra and used in program analyzers.



(a) Grid $\mathcal{L}_1 \cap \mathcal{L}_2$ (b) Grid $\mathcal{L}_1 \oplus \mathcal{L}_2$

Fig. 2. Grid intersection and join

4.1 Grid Intersection and Grid Join

For any pair of grids $\mathcal{L}_1, \mathcal{L}_2 \in \mathbb{G}_n$, the *intersection* of $\mathcal{L}_1$ and $\mathcal{L}_2$, defined as the set intersection $\mathcal{L}_1 \cap \mathcal{L}_2$, is the largest grid included in both $\mathcal{L}_1$ and $\mathcal{L}_2$; similarly, the *grid join* of $\mathcal{L}_1$ and $\mathcal{L}_2$, denoted by $\mathcal{L}_1 \oplus \mathcal{L}_2$, is the smallest grid that includes both $\mathcal{L}_1$ and $\mathcal{L}_2$.

In theoretical terms, the intersection and grid join operators defined above are the binary *meet* and the binary *join* operators on the lattice $\mathbb{G}_n$.

Example 2. Consider the $\mathbb{G}_2$ grids $\mathcal{L} = \operatorname{gcon}(\mathcal{C})$ and $\mathcal{L}' = \operatorname{gcon}(\mathcal{C}')$ where

$$\mathcal{C} := \{x \equiv_2 0,\ -x + y \equiv_3 0\} \text{ and } \mathcal{C}' := \{x \equiv_4 0,\ -x + 2y \equiv_6 0\}.$$

The grids $\mathcal{L}$ and $\mathcal{L}'$ are illustrated by the squares and circles in Figure 2(a), respectively. The grid-intersection $\mathcal{L} \cap \mathcal{L}' = \operatorname{gcon}(\mathcal{C} \cup \mathcal{C}')$ is illustrated by the filled circles in the same diagram. Observe that, if

$$\mathcal{C}'' := \{x \equiv_{12} 0,\ y \equiv_3 0\}$$

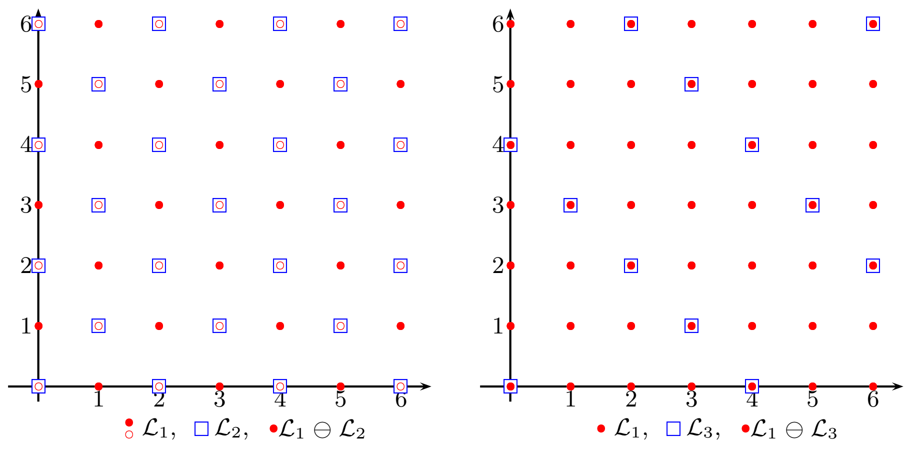
then $\mathcal{C}''$ is a reduced form of $\mathcal{C} \cup \mathcal{C}'$ so that we also have $\mathcal{L} \cap \mathcal{L}' = \operatorname{gcon}(\mathcal{C}'')$.

Example 3. Consider the grids $\mathcal{L} = \operatorname{ggen}\bigl((\varnothing, \varnothing, P)\bigr)$ and $\mathcal{L}' = \operatorname{ggen}\bigl((\varnothing, \varnothing, P')\bigr)$ in $\mathbb{G}_2$ such that

$$P := \begin{pmatrix} 2 & 0 & 0 \\ 2 & 3 & 0 \end{pmatrix} \text{ and } P' := \begin{pmatrix} 4 & 0 & 0 \\ 2 & 3 & 0 \end{pmatrix}.$$

The grids $\mathcal{L}$ and $\mathcal{L}'$ are illustrated by the squares and filled circles in Figure 2(b), respectively. The grid-join $\mathcal{L} \oplus \mathcal{L}' = \operatorname{ggen}\bigl((\varnothing, \varnothing, P \cup P')\bigr)$ is illustrated by all (open and filled) circles in the same diagram. Observe that, if

$$Q'' := \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } P'' := \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$



(a) Grid $\mathcal{L}_1 \ominus \mathcal{L}_2$ (b) Grid $\mathcal{L}_1 \ominus \mathcal{L}_3$

Fig. 3. Grid difference

then $(\varnothing, Q'', P'')$ is minimal form of $(\varnothing, \varnothing, P \cup P')$ so that we also have $\mathcal{L} \oplus \mathcal{L}' = \mathrm{ggen}\big((\varnothing, Q'', P'')\big)$. Note that, in this example, $\mathcal{L} \oplus \mathcal{L}' \neq \mathcal{L} \cup \mathcal{L}'$.

4.2 Grid Difference

For any pair of grids $\mathcal{L}_1, \mathcal{L}_2 \in \mathbb{G}_n$, the *grid difference* of $\mathcal{L}_1$ and $\mathcal{L}_2$, denoted by $\mathcal{L}_1 \ominus \mathcal{L}_2$, is defined as the smallest grid containing the set-theoretic difference of $\mathcal{L}_1$ and $\mathcal{L}_2$.

Example 4. Consider grids

$$\begin{aligned}
\mathcal{L}_1 &:= \mathrm{gcon}\big(\{x \equiv_1 0, y \equiv_1 0\}\big) \\
\mathcal{L}_2 &:= \mathrm{gcon}\big(\{x \equiv_1 0, x + y \equiv_2 0\}\big) \\
\mathcal{L}_3 &:= \mathrm{gcon}\big(\{x \equiv_1 0, x + y \equiv_4 0\}\big).
\end{aligned}$$

The grids $\mathcal{L}_1$ and $\mathcal{L}_2$ are illustrated by all the circles (open and solid) and squares, respectively, in Figure 3(a); the grid difference

$$\mathcal{L}_1 \ominus \mathcal{L}_2 = \mathrm{gcon}\big(\{x \equiv_1 0, x + y \equiv_2 1\}\big).$$

is illustrated by the filled circles. The grids $\mathcal{L}_1$ and $\mathcal{L}_3$ are illustrated by the circles and squares, respectively, in Figure 3(b); in this case, the grid difference is $\mathcal{L}_1 \ominus \mathcal{L}_3 = \mathcal{L}_1$.

```
Algorithm 1: The grid difference algorithm
Input: Two grids $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_1)$ in $\mathbb{G}_n$.
Output: A grid in $\mathbb{G}_n$.
(1)     if $\mathcal{L}_1 = \varnothing \vee \mathcal{L}_2 = \varnothing$
(2)         return $\mathcal{L}_1$
(3)     if $\mathcal{L}_1 \subseteq \mathcal{L}_2$
(4)         return $\varnothing$
(5)     $\mathcal{L}' :=$ the empty grid in $\mathbb{G}_n$
(6)     while $\exists \beta = (e \equiv_f 0) \in \mathcal{C}_2$
(7)         $\mathcal{C}_2 := \mathcal{C}_2 \setminus \{\beta\}$
(8)         if $\mathcal{L}_1 \nsubseteq \mathrm{gcon}\bigl(\{\beta\}\bigr)$
(9)             if $\mathcal{L}_1 \subseteq \mathrm{gcon}\bigl(\{2e \equiv_f 0\}\bigr)$
(10)                $\mathcal{L}_\beta := \mathrm{gcon}\bigl(\mathcal{C}_1 \cup \{2e - f \equiv_{2f} 0\}\bigr)$
(11)                $\mathcal{L}' := \mathcal{L}' \oplus \mathcal{L}_\beta$
(12)            else
(13)                return $\mathcal{L}_1$
(14)    return $\mathcal{L}'$
```

Algorithm 1 provides an implementation for grid difference.

Proposition 5. *Let $\mathcal{L}_1, \mathcal{L}_2 \in \mathbb{G}_n$ and suppose that $\mathcal{L}$ is the grid returned by* Algorithm 1. *Then $\mathcal{L} = \mathcal{L}_1 \ominus \mathcal{L}_2$.*

4.3 Affine Images and Preimages

Given a grid $\mathcal{L} \in \mathbb{G}_n$, a variable x_k and linear expression $e = \langle \boldsymbol{a}, \boldsymbol{x} \rangle + b$ with coefficients in $\mathbb{Q}$, the *affine image operator* $\phi(\mathcal{L}, x_k, e)$ maps the grid $\mathcal{L}$ to

$$\Bigl\{ \bigl(p_1, \ldots, p_{k-1}, \langle \boldsymbol{a}, \boldsymbol{p} \rangle + b, p_{k+1}, \ldots, p_n\bigr)^{\mathrm{T}} \in \mathbb{R}^n \Bigm| \boldsymbol{p} \in \mathcal{L} \Bigr\}.$$

Conversely, the *affine preimage operator* $\phi^{-1}(\mathcal{L}, x_k, e)$ maps the grid $\mathcal{L}$ to

$$\Bigl\{ \boldsymbol{p} \in \mathbb{R}^n \Bigm| \bigl(p_1, \ldots, p_{k-1}, \langle \boldsymbol{a}, \boldsymbol{p} \rangle + b, p_{k+1}, \ldots, p_n\bigr)^{\mathrm{T}} \in \mathcal{L} \Bigr\}.$$

Observe that the affine image $\phi(\mathcal{L}, x_k, e)$ and preimage $\phi^{-1}(\mathcal{L}, x_k, e)$ are invertible if and only if the coefficient a_k in the vector $\boldsymbol{a}$ is non-zero.

Example 5. Suppose $\mathcal{L} = \mathrm{ggen}\bigl((\varnothing, \varnothing, P)\bigr) \in \mathbb{G}_2$, where $P = \bigl(\begin{smallmatrix} 0 & 0 & 3 \\ 0 & 3 & 0 \end{smallmatrix}\bigr)$. Then

$$\mathcal{L}_1 = \phi(\mathcal{L}, x, 3x + 2y + 1) = \mathrm{ggen}\bigl((\varnothing, \varnothing, P_1)\bigr) = \mathrm{gcon}(\mathcal{C}_1)$$

where $P_1 = \bigl(\begin{smallmatrix} 1 & 7 & 10 \\ 0 & 3 & 0 \end{smallmatrix}\bigr)$ and $\mathcal{C}_1 = \bigl\{(x \equiv_3 1), (x + y \equiv_9 1)\bigr\}$. On the other hand,

$$\mathcal{L}_2 = \phi(\mathcal{L}, x, y) = \mathrm{ggen}\bigl((\varnothing, \varnothing, P_2)\bigr) = \mathrm{gcon}(\mathcal{C}_2)$$

where $P_2 = \bigl(\begin{smallmatrix} 0 & 3 \\ 0 & 3 \end{smallmatrix}\bigr)$ and $\mathcal{C}_2 = \bigl\{(x \equiv_3 0), (y \equiv_3 0), (x - y = 0)\bigr\}$; that is, the grid containing all the points whose coordinates are integral multiples of 3 and lie on line $x = y$.

Note that the affine preimage of $\mathcal{L}_1$ using the same variable x and linear expression $3x + 2y + 1$ is the original grid $\mathcal{L}$. Moreover, the affine preimage of $\mathcal{L}_2$ using variable x and linear expression y is the grid $\operatorname{ggen}\bigl((L_3, Q_3, P_3)\bigr)$ where

$$L_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \; Q_3 = \begin{pmatrix} 0 \\ 3 \end{pmatrix} \text{ and } P_3 = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

that is, all the points in $\mathbb{R}^2$ where the y coordinate is an integral multiple of 3.

The *generalized affine image* and *generalized affine preimage* are extensions of the affine image and affine preimage operators defined above. Given a grid $\mathcal{L} \in \mathbb{G}_n$, linear expressions $e' = \langle \boldsymbol{a}', \boldsymbol{x} \rangle + b'$, $e = \langle \boldsymbol{a}, \boldsymbol{x} \rangle + b$ with coefficients in $\mathbb{Q}$ and $f \in \mathbb{Q}$, the generalized affine image operator $\psi(\mathcal{L}, e', f, e)$ will transform the grid $\mathcal{L}$ to the grid

$$\left\{ \boldsymbol{w} \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \boldsymbol{v} \in \mathcal{L} \\ \bigl(i \in \{1, \ldots, n\} \land a'_i = 0\bigr) \implies w_i = v_i \\ \langle \boldsymbol{a}', \boldsymbol{w} \rangle + b' \equiv_f \langle \boldsymbol{a}, \boldsymbol{v} \rangle + b \end{array} \right\}.$$

Note that, when $e' = x_k$ and $f = 0$, then the transformation is equivalent to the standard affine transformation on $\mathcal{L}$ with respect to the variable x_k and the affine expression e; that is

$$\psi(\mathcal{L}, x_k, 0, e) = \phi(\mathcal{L}, x_k, e).$$

Example 6. Let $\mathcal{L}$ and $\mathcal{L}_1$ be as defined in Example 5. Then

$$\psi(\mathcal{L}, x, 0, 3x + 2y + 1) = \phi(\mathcal{L}, x, 3x + 2y + 1) = \mathcal{L}_1.$$

On the other hand,

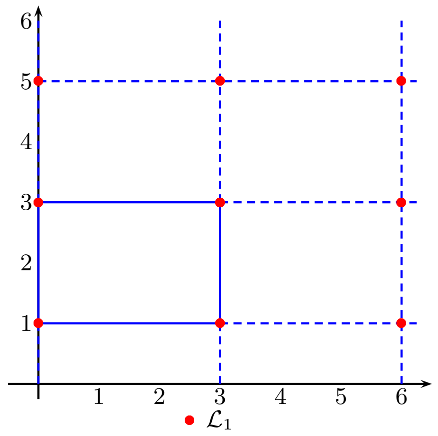
$$\psi(\mathcal{L}, x, 3, 3x + 2y + 1) = \operatorname{ggen}\bigl((\varnothing, \varnothing, P'_1)\bigr) = \operatorname{gcon}(\mathcal{C}'_1)$$

where $P'_1 = \left(\begin{smallmatrix} 1 & 4 & 1 \\ 0 & 0 & 3 \end{smallmatrix}\right)$ and $\mathcal{C}'_1 = \bigl\{(x \equiv_3 1), (y \equiv_3 0)\bigr\}$.

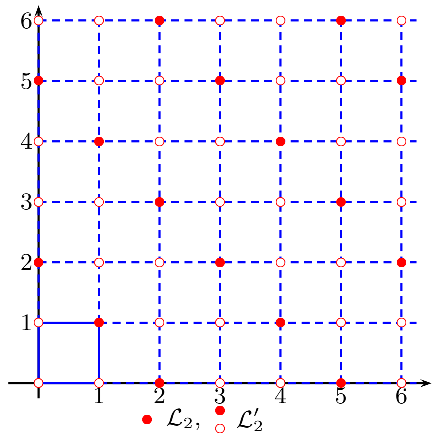
4.4 Rectilinear Grids and Covering Boxes

We show how we can reuse the standard interval domain [7] to represent a *rectilinear grid*, which is a grid that can be defined by a set of non-relational congruences (i.e., congruences where at most one coefficient is non-zero). We also show how such grids can provide safe approximations for any rational grid.

A non-empty n-dimensional *rational box* $\mathcal{B}$ is a sequence $(I_1, \ldots . I_n)$ of rational intervals. A *rational interval* is a pair $(\mu_i, \nu_i) \in \bigl(\mathbb{Q} \cup \{\infty\}\bigr)^2$ of *bounds*, called *lower* and *upper*, respectively. The lower bound is always less than or equal to the upper bound, where $q < \infty$ for each $q \in \mathbb{Q}$. If both the bounds are in $\mathbb{Q}$, the interval is said to be *bounded*.



(a) Grid $\mathcal{L}_1$



(b) Grids $\mathcal{L}_2$ and $\mathcal{L}_2'$

Fig. 4. Covering boxes for a grid

The essential idea we use here exploits the fact that the shape and size of an n-dimensional box $\mathcal{B}$ defines a rectangular "tile" that may be used to cover an n-dimensional vector space. It therefore follows that the box $\mathcal{B}$ determines a *covering* (i.e., tiling) of the n-dimensional vector space, where the given box $\mathcal{B}$ provides the position for one of the tiles. By defining a grid to be the vertices of the tiles in such a tiling, we obtain a rational rectilinear grid $\mathcal{L}$ and call $\mathcal{B}$ a *covering box for* $\mathcal{L}$.

Let $\mathcal{B} = (I_1, \dots . I_n)$ be a non-empty box. For each $i = 1, \dots, n$, let $I_i = (\mu_i, \nu_i)$; then, if $\mu_i \neq \nu_i$, let

$$\beta_i := \begin{cases} \bigl(\langle \boldsymbol{e}_i, \boldsymbol{x} \rangle \equiv_{\nu_i - \mu_i} \mu_i\bigr), & \text{if } I_i \text{ is bounded;} \\ \bigl(\langle \boldsymbol{e}_i, \boldsymbol{x} \rangle = \mu_i\bigr), & \text{if } I_i \text{ is not bounded.} \end{cases}$$

Then we say that the box $\mathcal{B}$ *represents* the grid $\mathcal{L} := \mathrm{gcon}(\mathcal{C})$, where

$$\mathcal{C} := \{\, \beta_i \bigm| 1 \leq i \leq n, \mu_i \neq \nu_i \,\}. \tag{2}$$

Note that the congruence system $\mathcal{C}$ is in minimal form. Observe also that, when $\mu_i = \nu_i$ for some $1 \leq i \leq n$, there is no corresponding congruence in $\mathcal{C}$ for (μ_i, ν_i); this is because, in this case, the tiling will cover every value in this dimension and hence there will be a line $\boldsymbol{e}_i$ in the generator representation of $\mathcal{L}$.

Consider now any non-empty rational grid $\mathcal{L}$. A *covering box* for $\mathcal{L}$ is a rational box representing the smallest rectilinear grid that contains $\mathcal{L}$.

Example 7. Consider grid $\mathcal{L}_1 = \mathrm{gcon}\bigl(\{x \equiv_3 0, y \equiv_2 1\}\bigr)$ illustrated by all the circles in Figure 4(a); then $\mathcal{L}_1$ is rectilinear and and box $\mathcal{B}_1 = \bigl\{(0,3), (1,3)\bigr\}$ is a covering box representing $\mathcal{L}_1$. Consider the grid $\mathcal{L}_2 = \mathrm{gcon}\bigl(\{x \equiv_1 0, x+y \equiv_3 2\}\bigr)$

illustrated by all the filled circles in Figure 4(b); then $\mathcal{L}_2$ is not rectilinear and box $\mathcal{B}_2 = \bigl\{(1,2),(1,2)\bigr\}$, is a covering box for $\mathcal{L}_2$. Thus $\mathcal{B}_2$ represents the grid $\mathcal{L}_2' = \mathrm{gcon}\bigl(\{x \equiv_1 1, y \equiv_1 1\}\bigr)$ which is illustrated by all the circles in Figure 4(b).

We now provide a procedure for computing the covering box of a grid.

Proposition 6. *Let $\mathcal{L} = \mathrm{ggen}(\mathcal{G})$ where $\mathcal{G} = \bigl(L, Q, \{\boldsymbol{p}\}\bigr)$. Let $q = \# Q$ and $Q = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_q\}$. Let $\mathcal{B} = (I_1, \ldots, I_n)$ such that, for each $i \in \{1, \ldots, n\}$:*

1. *if, for some $\boldsymbol{\ell} \in L$, $\ell_i \neq 0$, then let $I_i := (0,0)$;*
2. *if, for all $\boldsymbol{\ell} \in L$, $\ell_i = 0$, and $q_{1i} = \cdots = q_{qi} = 0$, let $I_i := (p_i, \infty)$;*
3. *otherwise, let $I_i := \bigl(p_i, p_i + |g|\bigr)$ where $g = \gcd\bigl(\{q_{1i}, \ldots, q_{qi}\}\bigr)$.*

Then $\mathcal{B}$ is a covering box for $\mathcal{L}$.

The complexity of computing the covering box for a grid in $\mathbb{G}_n$ using the procedure given in Proposition 6, is $\mathrm{O}\bigl(n^2\bigr)$.

It should be stressed that a grid $\mathcal{L} \in \mathbb{G}_n$ does not have, in general, a unique covering box. For instance, if $\mathcal{B} = (I_1, \ldots, I_i, \ldots, I_n)$ is a covering box for $\mathcal{L}$, and the interval $I_i = (\mu_i, \nu_i)$ is bounded, then the box $\mathcal{B}' = (I_1, \ldots, I_i', \ldots, I_n)$ is also a covering box for $\mathcal{L}$ if, for some $m \in \mathbb{Z}$, $I_i' = (\mu_i + m(\nu_i - \mu_i), \nu_i + m(\nu_i - \mu_i))$.

4.5 Other Grid Operators

There are many operations that a practical domain of grids must provide for applications in program analysis and verification. For instance, the *concatenation* of two grids $\mathcal{L}_1 \in \mathbb{G}_n$ and $\mathcal{L}_2 \in \mathbb{G}_m$ (taken in this order) is the grid in $\mathbb{G}_{n+m}$ defined as

$$\left\{ (x_1, \ldots, x_n, y_1, \ldots, y_m)^{\mathrm{T}} \in \mathbb{R}^{n+m} \;\middle|\; \begin{array}{l} (x_1, \ldots, x_n)^{\mathrm{T}} \in \mathcal{L}_1 \\ (y_1, \ldots, y_m)^{\mathrm{T}} \in \mathcal{L}_2 \end{array} \right\}.$$

Other operators are required that add, remove, rename and map the space dimensions, or expand and fold them along the lines of [13]. Even though all these operations have been specified and implemented, space considerations make their discussion here inadvisable: the interested reader is referred to the web site of the Parma Polyhedra Library, `http://www.cs.unipr.it/ppl/`, where all the code and documentation is publicly available.

5 Implementation

In this section, we first describe convenient internal representations of the congruence and generator systems in terms of arrays (i.e., matrices). Then we show how matrix inversion provides a basis for converting between the (possibly singular) matrix representations of the congruence and generator systems in minimal form.

5.1 Homogeneous Representations

A congruence system $\mathcal{C}$ is *homogeneous* if, for all $\bigl(\langle \boldsymbol{a}, \boldsymbol{x} \rangle \equiv_f b\bigr) \in \mathcal{C}$, we have $b = 0$. Similarly, a generator system (L, Q, P) is *homogeneous* if $\mathbf{0} \in P$. For the implementation, it is convenient to work with a homogeneous system. Thus we first convert any congruence or generator systems in $\mathbb{Q}^n$ to a homogeneous representation in $\mathbb{Q}^{n+1}$. The extra dimension is denoted with a 0 subscript and $\hat{\boldsymbol{x}}$ denotes the vector $(x_0, \dots, x_n)^{\mathrm{T}}$; we also let $\boldsymbol{e}_0$ denote the vector $(1, \mathbf{0}^{\mathrm{T}})^{\mathrm{T}}$.

Consider the congruence system $\mathcal{C} = \mathcal{F} \cup \mathcal{E}$ in $\mathbb{Q}^n$ where $\mathcal{E}$ is a set of equalities and $\mathcal{F}$ is a set of proper congruences. Then the *homogeneous form* for $\mathcal{C}$ is the congruence system $\hat{\mathcal{C}} = \hat{\mathcal{F}} \cup \hat{\mathcal{E}}$ in $\mathbb{Q}^{n+1}$ defined by:

$$\hat{\mathcal{F}} := \Bigl\{ \bigl\langle f^{-1}(-b, \boldsymbol{a}^{\mathrm{T}})^{\mathrm{T}}, \hat{\boldsymbol{x}} \bigr\rangle \equiv_1 0 \Bigm| \bigl(\langle \boldsymbol{a}, \boldsymbol{x} \rangle \equiv_f b\bigr) \in \mathcal{F} \Bigr\} \cup \bigl\{ \langle \boldsymbol{e}_0, \hat{\boldsymbol{x}} \rangle \equiv_1 0 \bigr\}, \tag{3}$$

$$\hat{\mathcal{E}} := \Bigl\{ \bigl\langle (-b, \boldsymbol{a}^{\mathrm{T}})^{\mathrm{T}}, \hat{\boldsymbol{x}} \bigr\rangle = 0 \Bigm| \bigl(\langle \boldsymbol{a}, \boldsymbol{x} \rangle = b\bigr) \in \mathcal{E} \Bigr\}. \tag{4}$$

The congruence $\langle \boldsymbol{e}_0, \hat{\boldsymbol{x}} \rangle \equiv_1 0$, expressing the fact that $1 \equiv_1 0$, is called the *integrality congruence*. By writing $\hat{\mathcal{F}} = (F^{\mathrm{T}} \boldsymbol{x} \equiv_1 \mathbf{0})$ and $\hat{\mathcal{E}} = (E^{\mathrm{T}} \boldsymbol{x} = \mathbf{0})$ where $F, E \subseteq \mathbb{Q}^{n+1}$, it can be seen that the pair (F, E), called the *matrix form* of the homogeneous system $\hat{\mathcal{C}}$, is sufficient to determine $\mathcal{C}$. Observe also that we can recover the original grid $\mathrm{gcon}(\mathcal{C})$ from $\mathrm{gcon}(\hat{\mathcal{C}})$ since:

$$\mathrm{gcon}(\mathcal{C}) = \bigl\{ \boldsymbol{v} \in \mathbb{R}^n \bigm| (1, \boldsymbol{v}^{\mathrm{T}})^{\mathrm{T}} \in \mathrm{gcon}(\hat{\mathcal{C}}) \bigr\}.$$

Consider next a generator system $\mathcal{G} = (L, Q, P)$ in $\mathbb{Q}^n$. Then the *homogeneous form* for $\mathcal{G}$ is the generator system $\hat{\mathcal{G}} := \bigl(\hat{L}, \hat{Q} \cup \hat{P}, \{\mathbf{0}\}\bigr) \in \mathbb{Q}^{n+1}$ where

$$\hat{L} := \bigl\{ (0, \boldsymbol{\ell}^{\mathrm{T}})^{\mathrm{T}} \bigm| \boldsymbol{\ell} \in L \bigr\},\ \hat{Q} := \bigl\{ (0, \boldsymbol{q}^{\mathrm{T}})^{\mathrm{T}} \bigm| \boldsymbol{q} \in Q \bigr\},\ \hat{P} := \bigl\{ (1, \boldsymbol{p}^{\mathrm{T}})^{\mathrm{T}} \bigm| \boldsymbol{p} \in P \bigr\}. \tag{5}$$

We can recover the original grid $\mathrm{ggen}(\mathcal{G})$ from $\mathrm{ggen}(\hat{\mathcal{G}})$ since:

$$\mathrm{ggen}(\mathcal{G}) = \bigl\{ \boldsymbol{v} \in \mathbb{R}^n \bigm| (1, \boldsymbol{v}^{\mathrm{T}})^{\mathrm{T}} \in \mathrm{ggen}(\hat{\mathcal{G}}) \bigr\}.$$

Suppose that $\mathcal{C}$ is a congruence (resp., $\mathcal{G}$ is a generator) system in $\mathbb{Q}^{n+1}$; then we say that $\mathcal{C}$ (resp., $\mathcal{G}$) *is in homogeneous form* if there exists a congruence system $\check{\mathcal{C}}$ (resp., generator system $\check{\mathcal{G}}$) in $\mathbb{Q}^n$ such that $\mathcal{C}$ (resp., $\mathcal{G}$) is the homogeneous form of $\check{\mathcal{C}}$ (resp., $\check{\mathcal{G}}$). Note that, if $(\mathcal{C}, \mathcal{G})$ is a double description for a grid and $\hat{\mathcal{C}}$ and $\hat{\mathcal{G}}$ are homogeneous forms for $\mathcal{C}$ and $\mathcal{G}$, then $(\hat{\mathcal{C}}, \hat{\mathcal{G}})$ is also a double description.

5.2 Converting Representations

In this subsection, we outline the *conversion algorithms* (that is, algorithms for converting from the congruence to the generator minimized representations of a nonempty grid and vice versa).

By considering the matrix forms of the (homogeneous forms of the) representations, we can build the conversion algorithms on top of standard techniques for matrix inversion. For an informal explanation why this is appropriate, suppose that the generator system $\mathcal{G} = \bigl(\varnothing, Q, \{\mathbf{0}\}\bigr)$ in $\mathbb{Q}^n$ is in minimal form and that $\# Q = n$. Then Q is a non-singular square matrix. Letting $\mathcal{L} = \operatorname{ggen}(\mathcal{G}) = \{\, Q\boldsymbol{\pi} \mid \boldsymbol{\pi} \in \mathbb{Z}^n \,\}$, then we also have $\mathcal{L} = \{\, \boldsymbol{v} \in \mathbb{R}^n \mid Q^{-1}\boldsymbol{v} \equiv_1 0 \,\}$, so that $(Q^{-1}, \varnothing)$ is the matrix form of a congruence system for the same grid $\mathcal{L}$. Similarly we can use matrix inversion to convert the matrix form of a homogeneous congruence system in minimal form consisting of n proper congruences for a grid $\mathcal{L}$ to a generator system for $\mathcal{L}$.

Proposition 7. *Let $\mathcal{C}$ be a congruence system in $\mathbb{Q}^n$ in minimal form. Let (F, E) be the matrix form of the homogeneous form for $\mathcal{C}$. Let N be a matrix in $\mathbb{Z}^{n+1}$ whose vectors are of the form $\boldsymbol{e}_i$, $i \in \{0, \dots, n\}$, and such that $(N, \hat{F}, \hat{E})$ is square and nonsingular. Let*

$$(\hat{L}, \hat{Q}, M) := \bigl((N, \hat{F}, \hat{E})^{-1}\bigr)^{\mathrm{T}}$$

where $\# \hat{L} = \# N$, $\# \hat{Q} = \# \hat{F}$ and $\# M = \# \hat{E}$. Then $\hat{\mathcal{G}} = \bigl(\hat{L}, \hat{Q}, \{\mathbf{0}\}\bigr)$ is the homogeneous form for a generator system $\mathcal{G}$ in minimal form and $\operatorname{ggen}(\mathcal{G}) = \operatorname{gcon}(\mathcal{C})$.

As matrix inversion and transposition are both computable and invertible, the conversion from generator to congruence system is similar.

Proposition 8. *Let $\mathcal{G}$ be a generator system in $\mathbb{Q}^n$ in minimal form and $\hat{\mathcal{G}} = \bigl(\hat{L}, \hat{Q}, \{\mathbf{0}\}\bigr)$ be the homogeneous form for $\mathcal{G}$. Let M be a matrix in $\mathbb{Z}^{n+1}$ whose vectors are of the form $\boldsymbol{e}_i$, $i \in \{0, \dots, n\}$, and such that $(\hat{L}, \hat{Q}, M)$ is square and nonsingular. Let*

$$(N, \hat{F}, \hat{E}) := \bigl((\hat{L}, \hat{Q}, M)^{-1}\bigr)^{\mathrm{T}}$$

where $\# N = \# \hat{L}$, $\# \hat{F} = \# \hat{Q}$ and $\# \hat{E} = \# M$. Then $(\hat{F}, \hat{E})$ is the matrix form of the homogeneous form for a congruence system $\mathcal{C}$ in minimal form and $\operatorname{gcon}(\mathcal{C}) = \operatorname{ggen}(\mathcal{G})$.

Both conversion algorithms just perform matrix inversion so that the complexity is that of matrix inversion, namely $\mathrm{O}\bigl(n^3\bigr)$.

6 Grid Widening

In this section we describe a widening for the domain of grids, and compare our proposal with an operator based on the standard widening for the domain of convex polyhedra.

A simple and general characterization of a widening for enforcing and accelerating convergence of an upward iteration sequence is given in [7–10]. We assume here a minor variation of this classical definition (see footnote 6 in [10, p. 275]).

Definition 4. (Widening.) *Let $\langle D, \vdash, \mathbf{0}, \oplus \rangle$ be a join-semilattice. The partial operator $\nabla \colon D \times D \rightarrowtail D$ is a* widening *if*

1. *for each $d_1, d_2 \in D$, $d_1 \vdash d_2$ implies that $d_1 \nabla d_2$ is defined and $d_2 \vdash d_1 \nabla d_2$;*
2. *for each increasing chain $d_0 \vdash d_1 \vdash \cdots$, the increasing chain defined by $d_0' := d_0$ and $d_{i+1}' := d_i' \nabla (d_i' \oplus d_{i+1})$, for $i \in \mathbb{N}$, is not strictly increasing.*

In addition to the formal requirements in Definition 4, it is also important to have a widening that has an efficient implementation, preferably, one that depends on a simple syntactic mapping of the representations. At the same time, so that the widening is well-defined, the result of this operation should be independent of the actual representation used. For this reason, the widening we propose requires the concept of *strong minimal form* for the congruence systems.

Definition 5. *A congruence system $\mathcal{C}$ in $\mathbb{Q}^n$ is in* strong minimal form *if $\mathcal{C}$ is in minimal form and, for each pair of distinct proper congruences*

$$\beta = \bigl(\langle \boldsymbol{a}, \boldsymbol{x} \rangle \equiv_1 b\bigr), \; \gamma = \bigl(\langle \boldsymbol{c}, \boldsymbol{x} \rangle \equiv_1 d\bigr) \in \mathcal{C},$$

if $\operatorname{piv}_<(\gamma) = k > 0$, then $-c_k < 2a_k \leq c_k$.

A congruence system in minimal form can always be reduced to a congruence system in strong minimal form that describes the same grid.

Proposition 9. *Let $\mathcal{C}$ be a congruence system in $\mathbb{Q}^n$ in minimal form. Then there exists an algorithm with complexity $\mathrm{O}\bigl(n^3\bigr)$ for converting $\mathcal{C}$ to a congruence system $\mathcal{C}'$ in strong minimal form such that $\mathcal{C}$ is pivot equivalent to $\mathcal{C}'$ and $\operatorname{gcon}(\mathcal{C}) = \operatorname{gcon}(\mathcal{C}')$.*

We also have that, relative to the affine hull of a grid, the strong minimal form of any congruence system that describes that grid is canonical.

Proposition 10. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be congruence systems in strong minimal form in $\mathbb{Q}^n$ such that $\mathcal{L} = \operatorname{gcon}(\mathcal{C}_1) = \operatorname{gcon}(\mathcal{C}_2)$. Then, for all $\beta \in \mathcal{C}_1$, there exists $\gamma \in \mathcal{C}_2$ such that*

$$\operatorname{gcon}\bigl(\{\beta\}\bigr) \cap \operatorname{affine.hull}(\mathcal{L}) = \operatorname{gcon}\bigl(\{\gamma\}\bigr) \cap \operatorname{affine.hull}(\mathcal{L}).$$

We now define a widening for grids.

Definition 6. *Let $\mathcal{L}_1 = \operatorname{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 = \operatorname{gcon}(\mathcal{C}_2)$ be two grids in $\mathbb{G}_n$ such that $\mathcal{L}_1 \subseteq \mathcal{L}_2$, $\mathcal{C}_1$ is in minimal form and $\mathcal{C}_2$ is in strong minimal form. Then the* grid widening *$\mathcal{L}_1 \nabla \mathcal{L}_2$ is defined by*

$$\mathcal{L}_1 \nabla \mathcal{L}_2 := \begin{cases} \mathcal{L}_2, & \text{if } \mathcal{L}_1 = \varnothing \text{ or } \dim(\mathcal{L}_1) < \dim(\mathcal{L}_2); \\ \operatorname{gcon}(\mathcal{C}_S), & \text{otherwise,} \end{cases}$$

where

$$\mathcal{C}_S := \{\, \gamma \in \mathcal{C}_2 \mid \exists \beta \in \mathcal{C}_1 \,.\, \beta \Uparrow \gamma \,\}. \tag{6}$$

Proposition 11. *The operator '∇' is a widening on $\mathbb{G}_n$.*

Assuming that the two congruence systems are already available and in strong minimal form, the complexity of this widening is $\mathrm{O}\bigl(n^2\bigr)$.

By Proposition 10, for a fixed variable ordering, the operator '∇' is well-defined. The next example illustrates that '∇' depends on the variable ordering.

Example 8. Consider the grids $\mathcal{L}_1 := \mathrm{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 := \mathrm{gcon}(\mathcal{C}_2)$ in $\mathbb{G}_2$, where

$$\mathcal{C}_1 := \{5x + y \equiv_1 0,\ 22x \equiv_1 0\}, \qquad \mathcal{C}_2 := \{5x + y \equiv_1 0,\ 44x \equiv_1 0\}.$$

Assume that variables are ordered so that x precedes y, as in the vector $(x, y)^{\mathrm{T}}$; then, the congruence systems $\mathcal{C}_1$ and $\mathcal{C}_2$ are in strong normal form and, according to Definition 6, we obtain $\mathcal{L}_1 \nabla \mathcal{L}_2 = \mathrm{gcon}\bigl(\{5x + y \equiv_1 0\}\bigr)$. On the other hand, the same grids as before are also described by the congruence systems

$$\mathcal{C}'_1 := \{9y + x \equiv_1 0,\ 22y \equiv_1 0\}, \qquad \mathcal{C}'_2 := \{9y + x \equiv_1 0,\ 44y \equiv_1 0\},$$

respectively, which are in strong normal form when considering the variable order where y precedes x. In this case, by Definition 6, $\mathcal{L}_1 \nabla \mathcal{L}_2 = \mathrm{gcon}\bigl(\{9y+x \equiv_1 0\}\bigr)$.

In Definition 6, we require that the second congruence systems $\mathcal{C}_2$ is in strong minimal form. The following example shows that this is necessary for the operator '∇' to be well-defined,

Example 9. Consider the congruence systems

$$\begin{aligned}\mathcal{C}_1 &:= \{x \equiv_2 0,\ y \equiv_2 0\},\\ \mathcal{C}_2 &:= \{x \equiv_1 0,\ x + y \equiv_2 0\},\\ \mathcal{C}_3 &:= \{x \equiv_1 0,\ 3x + y \equiv_2 0\},\end{aligned}$$

and let $\mathcal{L}_1 := \mathrm{gcon}(\mathcal{C}_1)$, $\mathcal{L}_2 := \mathrm{gcon}(\mathcal{C}_2)$ and $\mathcal{L}_3 := \mathrm{gcon}(\mathcal{C}_3)$; then $\mathcal{L}_2 = \mathcal{L}_3$. Note that only $\mathcal{C}_1$ and $\mathcal{C}_2$ are in strong minimal form. Therefore, assuming $\mathcal{C}_S$ (resp., $\mathcal{C}_S{}'$) is defined as in (6) using $\mathcal{C}_1$ and $\mathcal{C}_2$ (resp., $\mathcal{C}_1$ and $\mathcal{C}_3$), we have

$$\mathcal{C}_S = \{x + y \equiv_2 0\}, \qquad \mathcal{C}_S{}' = \{3x + y \equiv_2 0\}.$$

Thus $\mathcal{L}_1 \nabla \mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_S) \neq \mathrm{gcon}(\mathcal{C}_S{}')$.

The grid widening '∇' defined above is a fully functional widening operator and, as such, can be used for the development of more refined widenings (or extrapolation operators) by following well-known techniques and, in particular, the frameworks proposed in [2, 4].

For instance, when instantiating the finite powerset construction [4] with the domain of grids, a widening for the powerset of grids can be obtained by designing a *finite convergence certificate* for the grid widening. In formal terms, this certificate is a triple $(\mathcal{O}, \succ, \mu)$ where $(\mathcal{O}, \succ)$ is a well-founded ordered set and $\mu \colon \mathbb{G}_n \to \mathcal{O}$, which is called *level mapping*, is such that, for all $\mathcal{L}_1 \subset \mathcal{L}_2 \in \mathbb{G}_n$,

$\mu(\mathcal{L}_1) \succ \mu(\mathcal{L}_1 \nabla \mathcal{L}_2)$. One such certificate can be easily defined by taking $\mathcal{O}$ equal to $\{0, \ldots, n\} \times \{0, \ldots, n\}$, $\succ$ the lexicographic ordering on $\mathcal{O}$ and, for all $\mathcal{L} \in \mathbb{G}_n$, letting $\mu(\mathcal{L}) := (\# \mathcal{E}, \# \mathcal{C})$ where $\mathcal{L} = \mathrm{gcon}(\mathcal{C})$, $\mathcal{C}$ is in minimal form, and $\mathcal{E} \subseteq \mathcal{C}$ is the set of equalities in $\mathcal{C}$. By Definition 6 and Proposition 2, if follows that $\mathcal{L}_1 \neq \mathcal{L}_2$ implies $\mu(\mathcal{L}_1) \succ \mu(\mathcal{L}_1 \nabla \mathcal{L}_2)$; hence we have a finite convergence certificate for '∇'.

The grid widening '∇' can be enhanced with additional delay constructs such as the *with token* option previously described for widenings on convex polyhedra [3]. Often in analysis or verification, the convergence guarantee that comes with a widening operator is not essential and in such cases, all that is required are *extrapolation* operators. These differ from widenings in that their use along an upper iteration sequence does not ensure convergence in a finite number of steps. In particular, following the *widening "up to"* technique as described in [20] for convex polyhedra, we define a *limited* extrapolation operator to be an operator that takes a congruence system as an additional parameter and uses it to improve the approximation yielded by the grid widening. We can also exploit the covering boxes, defined in Subsection 4.4, to provide a *covered* extrapolation operator that improves the approximation yielded by the widening operator by ensuring that the result cannot be worse than the covering box for the grid in the second argument of the widening.

6.1 Discussion

To motivate further our choice of '∇' for the grid widening, consider the following grid counterpart of the standard widening for convex polyhedra as specified in the PhD thesis of N. Halbwachs [18], also described in [20].

Let $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2) \in \mathbb{G}_n$ where $\mathcal{L}_1 \subseteq \mathcal{L}_2$ and $\mathcal{C}_1$ and $\mathcal{C}_2$ are congruence systems in $\mathbb{R}^n$ in strong minimal form. Then $h(\mathcal{L}_1, \mathcal{L}_2) \in \mathbb{G}_n$ is defined to be $\mathcal{L}_2$, if $\mathcal{L}_1 = \varnothing$, and $\mathrm{gcon}(\mathcal{C}_1' \cup \mathcal{C}_2')$, if $\mathcal{L}_1 \neq \varnothing$, where

$$\mathcal{C}_1' := \left\{ \beta \in \mathcal{C}_1 \;\middle|\; \mathcal{L}_2 \subseteq \mathrm{gcon}\bigl(\{\beta\}\bigr) \right\}, \tag{7}$$

$$\mathcal{C}_2' := \left\{ \gamma \in \mathcal{C}_2 \;\middle|\; \exists \beta \in \mathcal{C}_1 \mathrel{.} \mathcal{L}_1 = \mathrm{gcon}\bigl(\mathcal{C}_1[\gamma/\beta]\bigr) \right\}. \tag{8}$$

One difference here with the standard widening for convex polyhedra is in the handling of equalities. This is because, whereas an equality is equivalent to two inequalities, there is no finite set of proper congruences that is semantically equivalent to an equality, so the above definition has ignored any distinction between equalities and proper congruences.

A second difference is that to ensure that the operator 'h' is well-defined, we require that *both* the congruence systems $\mathcal{C}_1$ and $\mathcal{C}_2$ are in strong minimal form whereas the standard widening just requires that the constraint system describing the smaller polyhedron is minimized.

Example 10. Consider again the grids and congruence systems in Example 9. Then, assuming definition (7) for $\mathcal{C}_1'$ and (8) for $\mathcal{C}_2'$ and $\mathcal{C}_3'$, we have $\mathcal{C}_1' = \varnothing$,

$\mathcal{C}_2' = \{x + y \equiv_2 0\}$ and $\mathcal{C}_3' = \{3x + y \equiv_2 0\}$ so that $h(\mathcal{L}_1, \mathcal{L}_2) = \mathcal{C}_1' \cup \mathcal{C}_2' = \mathrm{gcon}\bigl(\{x + y \equiv_2 0\}\bigr)$ although $\mathcal{C}_1' \cup \mathcal{C}_3' = \{3x + y \equiv_2 0\}$.

Observe that the computation of the congruence system $\mathcal{C}_2'$ should be carefully tuned, since any naive implementation is going to be rather expensive. On the other hand, Example 10 illustrates that, simply ignoring the $\mathcal{C}_2'$ component (as done in some implementations of the standard widening for convex polyhedra) can lose precision, even when the affine hull of both the grids is the universe.

7 Applications

In this section we discuss applications for the domain of rational grids in program analysis and verification.

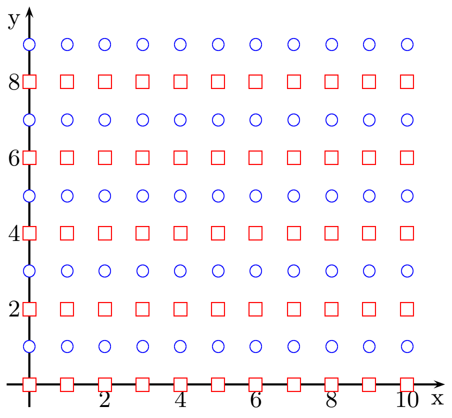
Many program properties are quantitative or depend on quantitative information. While such information may depend directly on the values of numerical data objects, it could instead reflect some numerical measures of the structure of the program and its data. We first discuss applications where the values of numeric variables are abstracted. In this case, any collection of numerical data objects (such as the numerical variables of some procedure) is a candidate for a non-trivial approximation via the grid domain.

Example 11. Consider again the example program given in Section 1 but annotated with program points Pj, for $j = 1, \ldots, 5$:

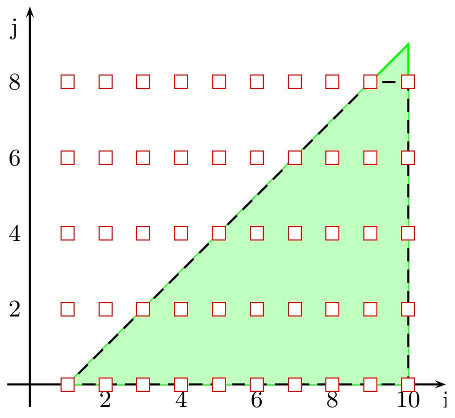
```
x := 2; y := 0;              (P1)
for i := 1 to m              (P2)
  if ... then
    x := x + 4               (P3)
  else
    x := x + 2; y := y + 1   (P4)
  endif                      (P5)
endfor
```

We show here that with the grid domain, we can derive the invariant set of congruences $\mathcal{C} = \{x + 2y \equiv_4 2,\ x \equiv_2 0\}$ that hold at point P2 and hence at end of the program, when the loop terminates. Let $G_j^i \in \mathbb{G}_2$ denote the grid computed at the i-th iteration executed by the point Pj. Initially, $G_j^0 = \varnothing = \mathrm{gcon}\bigl(\{1 = 0\}\bigr)$, for $j = 1, \ldots, 5$. After the first iteration of the analysis, we have:

$$\begin{aligned}
G_1^1 &= \mathrm{gcon}\bigl(\{x = 2,\ y = 0\}\bigr),\\
G_2^1 &= \mathrm{gcon}\bigl(\{x = 2,\ y = 0\}\bigr),\\
G_3^1 &= \mathrm{gcon}\bigl(\{x = 6,\ y = 0\}\bigr),\\
G_4^1 &= \mathrm{gcon}\bigl(\{x = 4,\ y = 1\}\bigr),\\
G_5^1 &= \mathrm{gcon}\bigl(\{x = 4,\ y = 1\}\bigr) \oplus \mathrm{gcon}\bigl(\{x = 6,\ y = 0\}\bigr)\\
&= \mathrm{gcon}\bigl(\{x + 2y = 6,\ x \equiv_2 0\}\bigr).
\end{aligned}$$



(a) Example 12: $\mathcal{L}_r$ and $\mathcal{L}_w$



(b) Example 13: $\mathcal{L}$, $\mathcal{P}$ and $\mathcal{P}'$

Fig. 5. Examples 12 and 13

Then we have

$$\begin{aligned} G_2^2 &= \operatorname{gcon}\bigl(\{x = 2, y = 0\}\bigr) \oplus \operatorname{gcon}\bigl(\{x + 2y = 6, x \equiv_2 0\}\bigr) \\ &= \operatorname{gcon}\bigl(\{x + 2y \equiv_4 2, x \equiv_2 0\}\bigr) \end{aligned}$$

and subsequent steps in the computation show that we have already computed an invariant for `P2` since we have $G_3^2 = G_3^1$, $G_4^2 = G_4^1$, $G_5^2 = G_5^1$ so that $G_3^2 = G_2^2$.

Example 11 shows that, even for a very simple piece of code, using the grids domain, we can find non-trivial *relational* congruence properties that cannot be found using the polyhedra domain [11], constraint-based analysis [34] or polynomial invariants [33].

Data dependence analysis for arrays —deciding if two elements of an array can refer to the same element and, if so, under what conditions— is required for advanced optimizing compilers [30].

Example 12. Consider the following program (adapted from a simple example given in [30]):

```
for i := 0 to 100
  for j := 2i to 100
    A[i, 2j + 1] := A[i, 2j]
  endfor
endfor
```

Then, the program reads from array elements $(0, 0)$, $(0, 2)$, $(1, 4)$, ... and writes to array elements $(0, 1)$, $(0, 3)$, $(1, 5)$, ... The two sets of points generate, respectively, the two grids $\mathcal{L}_r$ and $\mathcal{L}_w$ in $\mathbb{R}^2$: $\mathcal{L}_r = \operatorname{ggen}\bigl\{(0, 0), (0, 2), (1, 4)\bigr\}$ includes all the array elements that are read from, while $\mathcal{L}_w = \operatorname{ggen}\bigl\{(0,1),(0,3),(1,5)\bigr\}$ includes all the array elements that are written to. Notice that, by computing the intersection $\mathcal{L}_r \cap \mathcal{L}_w$ we can deduce that no location is both read and written.

Figure 5(a) illustrates the grids $\mathcal{L}_r$ and $\mathcal{L}_w$ where squares denote the points of the grid $\mathcal{L}_r$ and circles denote the points of the grid $\mathcal{L}_w$.

As mentioned in Section 1, most of the previous work on domains for numerical information concerns the representation of bounds on the values of the data objects. When one of these bounds is strict, an implementation normally has to use a different and, usually, less efficient representation compared with that for domain elements that are known to be topologically closed (and therefore defined entirely by non-strict constraints). However, if the data objects can only take discrete values and we have information about the frequency of their distribution, we can convert all the inequalities into non-strict ones, allowing for a more efficient implementation as well as possibly improving the precision.

Example 13. Consider the following program fragment:

```
for i := 1 to 10
  j := 0
  while j < i
    A[j] := A[j + 2];
    j := j + 2
  endwhile
endfor
```

Then the linear constraints representing the values of i and j at the start of each iteration of the while loop will lead to a convex polyhedron bounded by constraints $\{1 \leq i \leq 10,\ 0 \leq j < i\}$. If we add the information that i, j must be integers, we obtain a polyhedron $\mathcal{P}$ bounded by constraints $\{1 \leq i \leq 10,\ 0 \leq j \leq i - 1\}$; so that it may be deduced that at most 9 elements of the vector A are changed. However, using a grid domain we can represent the distribution of the possible values of i and j by a grid $\mathcal{L}$ described by the congruences $\{i \equiv_1 0,\ j \equiv_2 0\}$; so that, by combining this information with that represented by $\mathcal{P}$, we can see that the values of i and j must lie in the polyhedron $\mathcal{P}'$ bounded by constraints $\{1 \leq i \leq 10,\ 0 \leq j \leq i - 1,\ j \leq 8\}$ and deduce that at most 8 elements of the vector A are changed since the value of $A[9]$ is unaltered.

Figure 5(b) illustrates $\mathcal{L}$, $\mathcal{P}$, and $\mathcal{P}'$ indicated, respectively, by the squares, the shaded area, and the region enclosed by the dashed lines.

Example 14. This example is taken from [19] and concerns a water tank monitor. Let t be the time in seconds and w the water level in the tank in meters. Let p denote the pump switch position (on or off). When the pump is running the water fills the tank at 1 meter a second. When the pump is off, the water empties from the tank at 2 meters a second. At the start when $t = 0$, the water level $w = 1$ and the pump position $p = \text{on}$. By changing the position of the pump switch, the monitor ensures that the water level never reaches 12 meters and

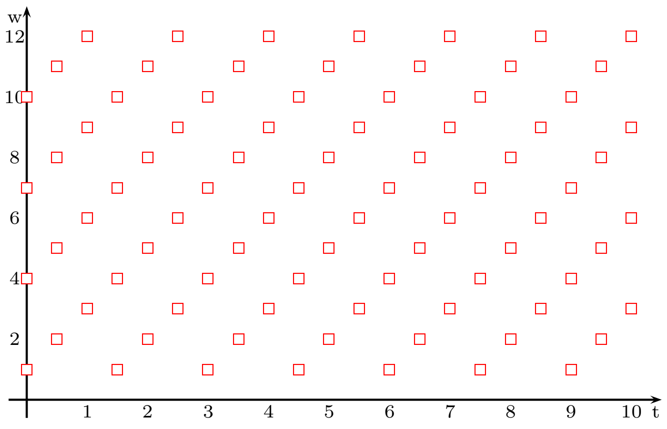


Fig. 6. Water tank monitor example

that there is always 2 meters of water in the tank. It checks the water level every half second. The procedure is described by the following program where T is a constant denoting the maximum time:

```
t = 0; w = 1; p = on;
while t < T
  t = t + 0.5;
  if p = on then
    w := w + 0.5;
    if w >= 11.5
      p := off
  else
    w := w - 1;
    if w <= 2
      p := on
    endif
  endif
endwhile
```

This will result in a grid of points relating the time t elapsed and the level of the water w. As the monitor checks every half second, we just have the discrete points at half second intervals. The grid is therefore generated by the set of points $\big\{(0,1),(1,2),(10.5,11.5),(11,10.5)\big\}$. This is also described by the set of congruences $\{2t \equiv_1 0, 2t - 2w \equiv_3 1\}$; the grid $\mathrm{gcon}\big(\{2t \equiv_1 0, 2t - 2w \equiv_3 1\}\big)$ is illustrated in Figure 6.

8 Conclusion

We have presented the abstract domain of grids, a domain that allows to represent and manipulate sets of regular spaced points and hyperplanes over $\mathbb{R}^n$. Two alternative grid representations in $\mathbb{Q}^n$ are described: one, a set of congruences, where the congruences can be either equalities or proper congruences; and the other, a set of generators, which can be lines, parameters or points.

We have provided algorithms for minimizing and converting the grid representations. Assuming the systems in $\mathbb{Q}^n$ consist of m congruences or generators, the minimization algorithms have complexity $\mathrm{O}\bigl(n^2 m\bigr)$. Note that other, previously proposed algorithms for minimization, which only consider the addition of one generator at a time, have complexity, at best, $\mathrm{O}\bigl(n^3\bigr)$ [27]. This is comparable with ours since, in this case, $m = \mathrm{O}\bigl(n\bigr)$. We have shown that existing algorithms for matrix inversion can be used for converting between generators and congruence systems so that these conversion procedures have complexity $\mathrm{O}\bigl(n^3\bigr)$ (for congruence to generator conversion this is an improvement over the previous proposals, which have complexity no better than $\mathrm{O}\bigl(n^4\bigr)$ [17]).

We have observed that the grid meet (resp., join) operations can be easily implemented if we have the congruence (resp. generator) representations for the grids as we then just need to take the union of the systems; hence both the operators have complexity $\mathrm{O}\bigl(n^3\bigr)$. In contrast, in [16] the grid join operation has complexity $\mathrm{O}\bigl(n^4 \log_2 n\bigr)$ since the generators of one of the grids are added, one at a time, to the generators of the other grid, reducing the new set at each stage to a linearly independent set. Moreover, because of the complexity of the conversion algorithm which is needed to convert the representation to a congruence system, the complexity of the grid meet operation in [16] is also $\mathrm{O}\bigl(n^4\bigr)$.

An algorithm for computing the grid difference, the best approximation by a grid for the difference of two grids, has been proposed and proved to be correct (see Section 4). We have proposed a grid widening operator for rational grids that is straightforward to implement. To ensure it is well-defined, the congruence system representing the grid has to be in a strong minimal form; that is, one in minimal form for which the non-diagonal elements of the matrix of coefficients of the proper congruences have to have a minimal absolute value. We also discuss how the proposed widening compares with the grid counterpart of the standard widening for convex polyhedra [20]. We have based the widening on the congruence representations of the grids. However, we anticipate that a dual form of the widening could also be defined that uses the generator representations instead; further investigation of this is future work.

Apart from a widening, we also show how the polyhedral domain of intervals may be used to represent a non-relational grid. We have also provided algorithms for finding the best approximation of an arbitrary relational grid as a non-relational grid. We observe that this idea of using a polyhedral domain to represent the "holes" in a grid could easily be generalized from that of representing a rectangular grid using covering boxes to the representation of any relational grid using n-dimensional parallelograms.

The grid domain is being implemented in the PPL [5, 6] following the approach and algorithms described in this paper. The current code is already publicly available in the `grids` branch of the PPL's CVS repository (see the PPL web site at `http://www.cs.unipr.it/ppl/` for more information). As the grid domain is implemented in a library providing several polyhedral domains, it can be combined with any of these domains to give not only the $\mathcal{Z}$-polyhedra domain but also many variations such as grid-polyhedra, grid-octagons, grid-bounded-differences and grid-intervals. Moreover, as the PPL provides full support for lifting any domain to the powerset of that domain, a user of the PPL will be able to experiment with powersets of grids (and of grid-polyhedra combinations) and the extra precision this provides.

References

1. C. Ancourt. *Génération Automatique de Codes de Transfert pour Multiprocesseurs à Mémoires Locales.* PhD thesis, Université de Paris VI, March 1991.
2. R. Bagnara, P. M. Hill, E. Ricci, and E. Zaffanella. Precise widening operators for convex polyhedra. In R. Cousot, editor, *Static Analysis: Proceedings of the 10th International Symposium*, volume 2694 of *Lecture Notes in Computer Science*, pages 337–354, San Diego, California, USA, 2003. Springer-Verlag, Berlin.
3. R. Bagnara, P. M. Hill, E. Ricci, and E. Zaffanella. Precise widening operators for convex polyhedra. *Science of Computer Programming*, 58(1–2):28–56, 2005.
4. R. Bagnara, P. M. Hill, and E. Zaffanella. Widening operators for powerset domains. In B. Steffen and G. Levi, editors, *Verification, Model Checking and Abstract Interpretation: Proceedings of the 5th International Conference (VMCAI 2004)*, volume 2937 of *Lecture Notes in Computer Science*, pages 135–148, Venice, Italy, 2003. Springer-Verlag, Berlin.
5. R. Bagnara, P. M. Hill, and E. Zaffanella. *The Parma Polyhedra Library User's Manual.* Department of Mathematics, University of Parma, Parma, Italy, release 0.8 edition, January 2005. Available at `http://www.cs.unipr.it/ppl/`.
6. R. Bagnara, E. Ricci, E. Zaffanella, and P. M. Hill. Possibly not closed convex polyhedra and the Parma Polyhedra Library. In M. V. Hermenegildo and G. Puebla, editors, *Static Analysis: Proceedings of the 9th International Symposium*, volume 2477 of *Lecture Notes in Computer Science*, pages 213–229, Madrid, Spain, 2002. Springer-Verlag, Berlin.
7. P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In B. Robinet, editor, *Proceedings of the Second International Symposium on Programming*, pages 106–130, Paris, France, 1976. Dunod, Paris, France.
8. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the Fourth Annual ACM Symposium on Principles of Programming Languages*, pages 238–252, New York, 1977. ACM Press.
9. P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, 1992.
10. P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation. In M. Bruynooghe and M. Wirsing, editors, *Proceedings of the 4th International Symposium on Programming Language Implementation and Logic Programming*, volume 631 of *Lecture Notes in Computer Science*, pages 269–295, Leuven, Belgium, 1992. Springer-Verlag, Berlin.
11. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Conference Record of the Fifth Annual ACM Symposium*

on Principles of Programming Languages, pages 84–96, Tucson, Arizona, 1978. ACM Press.
12. R. Giacobazzi, editor. *Static Analysis: Proceedings of the 11th International Symposium*, volume 3148 of *Lecture Notes in Computer Science*, Verona, Italy, 2004. Springer-Verlag, Berlin.
13. D. Gopan, F. DiMaio, N. Dor, T. Reps, and M. Sagiv. Numeric domains with summarized dimensions. In K. Jensen and A. Podelski, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 10th International Conference, TACAS 2004*, volume 2988 of *Lecture Notes in Computer Science*, pages 512–529. Springer-Verlag, Berlin, 2004.
14. P. Granger. Static analysis of arithmetical congruences. *International Journal of Computer Mathematics*, 30:165–190, 1989.
15. P. Granger. *Analyses Sémantiques de Congruence*. PhD thesis, École Polytechnique, 921128 Palaiseau, France, July 1991.
16. P. Granger. Static analysis of linear congruence equalities among variables of a program. In Samson Abramsky and T. S. E. Maibaum, editors, *TAPSOFT'91: Proceedings of the International Joint Conference on Theory and Practice of Software Development, Volume 1: Colloquium on Trees in Algebra and Programming (CAAP'91)*, volume 493 of *Lecture Notes in Computer Science*, pages 169–192, Brighton, UK, 1991. Springer-Verlag, Berlin.
17. P. Granger. Static analyses of congruence properties on rational numbers (extended abstract). In P. Van Hentenryck, editor, *Static Analysis: Proceedings of the 4th International Symposium*, volume 1302 of *Lecture Notes in Computer Science*, pages 278–292, Paris, France, 1997. Springer-Verlag, Berlin.
18. N. Halbwachs. *Détermination Automatique de Relations Linéaires Vérifiées par les Variables d'un Programme*. Thèse de 3$^{\text{ème}}$ cycle d'informatique, Université scientifique et médicale de Grenoble, Grenoble, France, March 1979.
19. N. Halbwachs, Y.-E. Proy, and P. Raymond. Verification of linear hybrid systems by means of convex approximations. In B. Le Charlier, editor, *Static Analysis: Proceedings of the 1st International Symposium*, volume 864 of *Lecture Notes in Computer Science*, pages 223–237, Namur, Belgium, 1994. Springer-Verlag, Berlin.
20. N. Halbwachs, Y.-E. Proy, and P. Roumanoff. Verification of real-time systems using linear relation analysis. *Formal Methods in System Design*, 11(2):157–185, 1997.
21. M. Karr. Affine relationships among variables of a program. *Acta Informatica*, 6:133–151, 1976.
22. S. Larsen, E. Witchel, and S. P. Amarasinghe. Increasing and detecting memory address congruence. In *Proceedings of the 2002 International Conference on Parallel Architectures and Compilation Techniques (PACT'02)*, pages 18–29, Charlottesville, VA, USA, 2002. IEEE Computer Society Press.
23. V. Loechner. *PolyLib*: A library for manipulating parameterized polyhedra. Available at `http://icps.u-strasbg.fr/~loechner/polylib/`, March 1999. Declares itself to be a continuation of [37].
24. A. Miné. A few graph-based relational numerical abstract domains. In M. V. Hermenegildo and G. Puebla, editors, *Static Analysis: Proceedings of the 9th International Symposium*, volume 2477 of *Lecture Notes in Computer Science*, pages 117–132, Madrid, Spain, 2002. Springer-Verlag, Berlin.
25. T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall. The double description method. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games – Volume II*, number 28 in Annals of Mathematics Studies, pages 51–73. Princeton University Press, Princeton, New Jersey, 1953.

26. M. Müller-Olm and H. Seidl. Analysis of modular arithmetic. In Mooly Sagiv, editor, *Programming Languages and Systems (Proceedings of the 14th Symposium, ESOP 2005)*, number 3444 in Lecture Notes in Computer Science, pages 46–60, 2005.
27. M. Müller-Olm and H. Seidl. A generic framework for interprocedural analysis of numerical properties. In C. Hankin, editor, *Proceedings of SAS 2005 (Static Analysis Symposium)*, Lecture Notes in Computer Science, 2005. to appear.
28. G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1988.
29. S. P. K. Nookala and T. Risset. A library for Z-polyhedral operations. *Publication interne* 1330, IRISA, Campus de Beaulieu, Rennes, France, 2000.
30. W. Pugh. A practical algorithm for exact array dependence analysis. *Communications of the ACM*, 35(8):102–114, 1992.
31. P. Quinton, S. Rajopadhye, and T. Risset. On manipulating Z-polyhedra. Technical Report 1016, IRISA, Campus Universitaire de Bealieu, Rennes, France, July 1996.
32. P. Quinton, S. Rajopadhye, and T. Risset. On manipulating Z-polyhedra using a canonic representation. *Parallel Processing Letters*, 7(2):181–194, 1997.
33. E. Rodríguez-Carbonell and D. Kapur. An abstract interpretation approach for automatic generation of polynomial invariants. In Giacobazzi [12], pages 280–295.
34. S. Sankaranarayanan, H. Sipma, and Z. Manna. Constraint-based linear-relations analysis. In Giacobazzi [12], pages 53–68.
35. A. Schrijver. *Theory of Linear and Integer Programming.* Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1999.
36. G. Stewart. *Introduction to matrix Computations.* Academic Press, 1973.
37. D. K. Wilde. A library for doing polyhedral operations. Master's thesis, Oregon State University, Corvallis, Oregon, December 1993. Also published as IRISA *Publication interne* 785, Rennes, France, 1993.

Appendix

A Proofs

Here we prove those results stated but not proved in the main part of the paper.

In the following two proofs, we require an extended version of the Euclidean algorithm. Suppose $v, v' \in \mathbb{Z}$. Let $r \in \mathbb{Z}$ be the greatest common divisor of v, v' and $s, s' \in \mathbb{Z}$ be relatively prime such that $sv + s'v' = r$. Then we write

$$\gcd(v, v') := r, \qquad \operatorname{gcdext}(v, v') := \bigl(r, (s, s')\bigr).$$

Proof (of Proposition 1). To prove the result, we first define the key transformation step in the algorithm and show that the resulting congruence system describes the same grid. Suppose there exist distinct congruences

$$\beta_1 = \bigl(\langle \boldsymbol{a}_1, \boldsymbol{x} \rangle \equiv_{f_1} b_1\bigr), \qquad \beta_2 = \bigl(\langle \boldsymbol{a}_2, \boldsymbol{x} \rangle \equiv_{f_2} b_2\bigr) \tag{9}$$

in $\mathcal{C}$ such that $\operatorname{piv}_<(\beta_1) = \operatorname{piv}_<(\beta_2) = i > 0$. We will define a congruence system $\mathcal{C}''$ such that either $\mathcal{C}'' = \mathcal{C} \setminus \{\beta_1, \beta_2\} \cup \{\beta_1'', \beta_2''\}$ or $\mathcal{C}'' = \mathcal{C} \setminus \{\beta_1, \beta_2\} \cup \{\beta_1''\}$ and show that $\operatorname{gcon}(\mathcal{C}) = \operatorname{gcon}(\mathcal{C}'')$. Letting

$$\beta_1'' = \bigl(\langle \boldsymbol{a}_1'', \boldsymbol{x} \rangle \equiv_{f_1} b_1''\bigr), \qquad \beta_2'' = \bigl(\langle \boldsymbol{a}_2'', \boldsymbol{x} \rangle \equiv_{f_2} b_2''\bigr),$$

we show that $\operatorname{piv}_<(\boldsymbol{a}_1'') = i$ and, if β_2'' is defined, then $\operatorname{piv}_<(\boldsymbol{a}_2'') < i$. There are two cases.

1. Either β_1 or β_2 is an equality; without loss of generality, we assume that β_1 is an equality so that $f_1 = 0$. Then, we let $\beta_1'' = \beta_1$ and, using Gaussian elimination,

$$\boldsymbol{a}_2'' = \boldsymbol{a}_2 - (a_{2i}/a_{1i})\boldsymbol{a}_1, \qquad b_2'' = b_2 - (a_{2i}/a_{1i})b_1.$$

2. Both β_1, β_2 are proper congruences; so that we can assume that $f_1 = f_2 = 1$. Let $\operatorname{gcdext}(a_{1i}, a_{2i}) = \bigl(r, (s, t)\bigr)$ Let

$$\begin{aligned} \boldsymbol{a}_1'' &= s\boldsymbol{a}_1 + t\boldsymbol{a}_2, & b_1'' &= sb_1 + tb_2 \\ \boldsymbol{a}_2'' &= (-a_{2i}/r)\boldsymbol{a}_1 + (a_{1i}/r)\boldsymbol{a}_2, & b_2'' &= (-a_{2i}/r)b_1 + (a_{1i}/r)b_2. \end{aligned}$$

In both cases, let $\mathcal{C}'' = \mathcal{C} \setminus \{\beta_1, \beta_2\} \cup \{\beta_1'', \beta_2''\}$ if $\boldsymbol{a}_2'' \neq \boldsymbol{0}$ or $b_2'' \neq 0$, and let $\mathcal{C}'' = \mathcal{C} \setminus \{\beta_1, \beta_2\} \cup \{\beta_1''\}$, otherwise. Then $\operatorname{gcon}(\mathcal{C}) = \operatorname{gcon}(\mathcal{C}'')$. Note that these transformations require a computation for each coefficient of the considered congruences so that their complexity is $\mathrm{O}\bigl(n\bigr)$.

The proof of the result is by induction on i; where $0 \leq i \leq n$ is the maximum value for which there exist distinct congruences $\beta_1, \beta_2 \in \mathcal{C}$ defined as in (9) such that $\operatorname{piv}_<(\beta_1) = \operatorname{piv}_<(\beta_2) = i$.

The base case is when $i = 0$ so that $\boldsymbol{a}_1 = \boldsymbol{a}_2 = \boldsymbol{0}$. In this case, if there exists $\bigl(\langle \boldsymbol{0}, \boldsymbol{x} \rangle \equiv_f b\bigr) \in \mathcal{C}$ and $b \equiv_f 0$ is inconsistent, let $\mathcal{C}' = \bigl\{\langle \boldsymbol{0}, \boldsymbol{x} \rangle \equiv_0 1\bigr\}$; otherwise, let $\mathcal{C}' = \bigl\{\, \beta \in \mathcal{C} \bigm| \operatorname{piv}_<(\beta) \neq 0 \,\bigr\}$.

For the step case, we keep applying the transformation (1) if either $f_1 = 0$ or $f_2 = 0$, and (2) otherwise until no more transformations are applicable for this index; that is when we obtain a congruence system $\mathcal{C}_j$ for which $j < i$ is the maximum index such that there exist distinct congruences β_1, $\beta_2 \in \mathcal{C}_j$ where $\operatorname{piv}_<(\beta_1) = \operatorname{piv}_<(\beta_2) = j$. We note that we will have to perform these transformations at most m times for each step, where $\#\mathcal{C} = m$, so that the complexity of each step is $\mathrm{O}\bigl(nm\bigr)$. By the inductive hypothesis, we can compute $\mathcal{C}'$ in minimal form such that $\operatorname{gcon}(\mathcal{C}') = \operatorname{gcon}(\mathcal{C}_j)$. Therefore $\operatorname{gcon}(\mathcal{C}') = \operatorname{gcon}(\mathcal{C})$. As we iterate at most n times over the step case, it can be seen that the algorithm has complexity $\mathrm{O}\bigl(n^2 m\bigr)$. $\square$

Proof (of Proposition 3). If $P = \varnothing$, then $\operatorname{ggen}(\mathcal{G}) = \varnothing$; in this case, let $\mathcal{G}' = (\varnothing, \varnothing, \varnothing)$. Suppose now that there exists a point $\boldsymbol{p} \in P$. Let $Q_{\boldsymbol{p}} = \bigl\{\boldsymbol{p}'' - \boldsymbol{p} \bigm| \boldsymbol{p}'' \in P \setminus \{\boldsymbol{p}\}\bigr\} \cup Q$ and let $\mathcal{G}_{\boldsymbol{p}} = \bigl(L, Q_{\boldsymbol{p}}, \{\boldsymbol{p}\}\bigr)$. Then, by equation (1), $\operatorname{ggen}(\mathcal{G}_{\boldsymbol{p}}) = \operatorname{ggen}(\mathcal{G})$.

To prove the result, we first define the key transformation step in the algorithm and show that the resulting generator system describes the same grid. Suppose there exist distinct generators $\boldsymbol{v}_1, \boldsymbol{v}_2 \in L \cup Q_{\boldsymbol{p}}$ such that $\operatorname{piv}_>(\boldsymbol{v}_1) = \operatorname{piv}_>(\boldsymbol{v}_2) = i \leq n$. We will define a generator system $\mathcal{G}'' = \bigl(L'', Q''_{\boldsymbol{p}}, \{\boldsymbol{p}\}\bigr)$ in $\mathbb{Q}^n$ where $L'' \cup Q''_{\boldsymbol{p}} = L \cup Q_{\boldsymbol{p}} \setminus \{\boldsymbol{v}_1, \boldsymbol{v}_2\} \cup \bigl(\{\boldsymbol{v}''_1, \boldsymbol{v}''_2\} \setminus \{\boldsymbol{0}\}\bigr)$, $\operatorname{ggen}(\mathcal{G}'') = \operatorname{ggen}(\mathcal{G}_{\boldsymbol{p}})$, $\operatorname{piv}_>(\boldsymbol{v}''_1) = i$ and, if $\boldsymbol{v}''_2 \neq \boldsymbol{0}$, $\operatorname{piv}_>(\boldsymbol{v}''_2) > i$. There are two cases.

1. Either $\boldsymbol{v}_1$ or $\boldsymbol{v}_2$ is in L; without loss of generality, we assume that $\boldsymbol{v}_1 \in L$. Then, using Gaussian elimination, let $\boldsymbol{v}''_1 = \boldsymbol{v}_1$ and
$$\boldsymbol{v}''_2 = \boldsymbol{v}_2 - (v_{2i}/v_{1i})\boldsymbol{v}_1.$$
Let $L'' = L \setminus \{\boldsymbol{v}_2\} \cup \bigl(\{\boldsymbol{v}''_2\} \setminus \{\boldsymbol{0}\}\bigr)$, if $\boldsymbol{v}_2 \in L$, and $Q''_{\boldsymbol{p}} = Q_{\boldsymbol{p}} \setminus \{\boldsymbol{v}_2\} \cup \bigl(\{\boldsymbol{v}''_2\} \setminus \{\boldsymbol{0}\}\bigr)$, if $\boldsymbol{v}_2 \in Q_{\boldsymbol{p}}$.
2. Both $\boldsymbol{v}_1, \boldsymbol{v}_2$ are in $Q_{\boldsymbol{p}}$. Let $\operatorname{gcdext}(v_{1i}, v_{2i}) = \bigl(r, (s, t)\bigr)$,
$$\begin{aligned} \boldsymbol{v}''_1 &= s\boldsymbol{v}_1 + t\boldsymbol{v}_2, & \boldsymbol{v}''_2 &= (-v_{2i}/r)\boldsymbol{v}_1 + (v_{1i}/r)\boldsymbol{v}_2, \\ L'' &= L, & Q''_{\boldsymbol{p}} &= Q_{\boldsymbol{p}} \setminus \{\boldsymbol{v}_1, \boldsymbol{v}_2\} \cup \bigl(\{\boldsymbol{v}''_1, \boldsymbol{v}''_2\} \setminus \{\boldsymbol{0}\}\bigr). \end{aligned}$$

In both cases, $\operatorname{ggen}(\mathcal{G}'') = \operatorname{ggen}(\mathcal{G}_{\boldsymbol{p}})$. Note that these transformations require a computation for each coefficient of the considered generators so that their complexity is $\mathrm{O}\bigl(n\bigr)$.

The proof of the result is by induction on $n + 1 - i$ where i is the least value such that, if there exists a pair of distinct generators $\boldsymbol{v}_1, \boldsymbol{v}_2 \in L \cup Q_{\boldsymbol{p}}$, then $i = \operatorname{piv}_>(\boldsymbol{v}_1) = \operatorname{piv}_>(\boldsymbol{v}_2)$ and $i = n + 1$, otherwise.

The base case is when $i = n + 1$, in which case $\mathcal{G}_{\boldsymbol{p}}$ is already in minimal form so let $\mathcal{G}' = \mathcal{G}_{\boldsymbol{p}}$.

For the step case, we apply the transformations (1) and (2) until no more transformations are applicable with index i; that is when we obtain a generator system $\mathcal{G}_j = (L_j, Q_j, \{\boldsymbol{p}\})$ for which $j > i$ is the least value such that, if there exists a pair of distinct generators $\boldsymbol{v}_1$, $\boldsymbol{v}_2 \in L_j \cup Q_j$, then $j = \operatorname{piv}_>(\boldsymbol{v}_1) = \operatorname{piv}_>(\boldsymbol{v}_2)$

and $j = n + 1$, otherwise. We note that we will have to perform these transformations at most m times for each step, where $\# L + \# Q_{\boldsymbol{p}} = m$, so that the complexity of each step is $\mathrm{O}\bigl(nm\bigr)$. By the inductive hypothesis, we can compute $\mathcal{G}'$ in minimal form such that $\mathrm{ggen}(\mathcal{G}') = \mathrm{ggen}(\mathcal{G}_j)$. Therefore $\mathrm{ggen}(\mathcal{G}') = \mathrm{ggen}(\mathcal{G})$. As we iterate at most n times over the step case, the algorithm has complexity $\mathrm{O}\bigl(n^2 m\bigr)$. □

Proof (of Proposition 5). If $\mathcal{L}_1 = \varnothing$ or $\mathcal{L}_2 = \varnothing$, then $\mathcal{L}_1 \ominus \mathcal{L}_2 = \mathcal{L}_1 \setminus \mathcal{L}_2 = \mathcal{L}_1$ and, by lines (1-2), the algorithm returns $\mathcal{L} = \mathcal{L}_1$. Similarly, if $\mathcal{L}_1 \subseteq \mathcal{L}_2$, then $\mathcal{L}_1 \ominus \mathcal{L}_2 = \mathcal{L}_1 \setminus \mathcal{L}_2 = \varnothing$ and, by lines (3-4), the algorithm returns $\mathcal{L} = \varnothing$. Suppose now that $\mathcal{L}_1 \neq \varnothing, \mathcal{L}_2 \neq \varnothing$ and $\mathcal{L}_1 \nsubseteq \mathcal{L}_2$. Let $\mathcal{L}'$ be the empty grid in $\mathbb{G}_n$ defined on line (5). Then the algorithm executes lines (6-14). Notice that there are two lines in this range that return a value for $\mathcal{L}$; line (13) when $\mathcal{L} = \mathcal{L}_1$ and line (14) when $\mathcal{L} = \mathcal{L}'$.

Consider first the case when line (13) is executed so that $\mathcal{L} = \mathcal{L}_1$. By definition of grid difference, $\mathcal{L} \supseteq \mathcal{L}_1 \ominus \mathcal{L}_2$; Hence it remains to show that $\mathcal{L}_1 \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. If $\boldsymbol{p} \in \mathcal{L}_1 \setminus \mathcal{L}_2$, then, by the definition of grid difference, $\boldsymbol{p} \in \mathcal{L}_1 \ominus \mathcal{L}_2$. Suppose now that $\boldsymbol{p} \in \mathcal{L}_1 \cap \mathcal{L}_2$. As line (13) is only executed by following the else branches of the conditionals on lines (8) and (9), for some congruence $\beta = (e \equiv_f 0) \in \mathcal{C}_2$, there exists a point $\boldsymbol{q} \in \mathcal{L}_1$ that does not satisfy $(2e \equiv_f 0)$ so that $\boldsymbol{q}$ does not satisfy β and hence $\boldsymbol{q} \notin \mathcal{L}_2$. Consider the point $\boldsymbol{r} = \boldsymbol{p} + 2(\boldsymbol{q} - \boldsymbol{p})$. Then, as $\boldsymbol{r}$ is an integral affine combination of points in $\mathcal{L}_1$, $\boldsymbol{r} \in \mathcal{L}_1$. Let $e = \bigl(\langle \boldsymbol{a}, \boldsymbol{x} \rangle - b\bigr)$. Then, as $\boldsymbol{p} \in \mathcal{L}_2$ satisfies β, $\langle \boldsymbol{a}, \boldsymbol{p} \rangle - b \equiv_f 0$. If $\boldsymbol{r}$ also satisfies β, then $\langle \boldsymbol{a}, \boldsymbol{r} \rangle - b \equiv_f 0$ and hence $\langle \boldsymbol{a}, 2\boldsymbol{q} \rangle - 2b \equiv_f 0$ so that $\boldsymbol{q}$ would satisfy $(2e \equiv_f 0)$; a contradiction. Thus $\boldsymbol{r} \notin \mathcal{L}_2$. Therefore $\boldsymbol{p} = 2\boldsymbol{q} - \boldsymbol{r}$ is an integral affine combination of points in $\mathcal{L}_1 \setminus \mathcal{L}_2$ and hence $p \in \mathcal{L}_1 \ominus \mathcal{L}_2$. As $\boldsymbol{p} \in \mathcal{L}_1 = \mathcal{L}$ was arbitrary, $\mathcal{L} \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$.

Suppose now that line (13) is not executed. Then the loop iterates once for each congruence in $\mathcal{C}_2$ before executing line (14). Suppose $\# \mathcal{C}_2 = c$ and $\beta_i = (e_i \equiv_f 0) \in \mathcal{C}_2$ is the congruence selected at line (6) in the i-th iteration of the loop, for $0 < i \leq c$. Let $\mathcal{L}'_0 = \varnothing$ and $\mathcal{L}'_i$ denote the grid $\mathcal{L}'$ after the i-th iteration. Then we need to show that $\mathcal{L}'_c = \mathcal{L}_1 \ominus \mathcal{L}_2$. We prove that $\mathcal{L}'_c \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$ and $\mathcal{L}'_c \supseteq \mathcal{L}_1 \ominus \mathcal{L}_2$ separately.

We first show that $\mathcal{L}'_c \supseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. Since $\mathcal{L}_1 \ominus \mathcal{L}_2$ is the smallest grid containing $\mathcal{L}_1 \setminus \mathcal{L}_2$, we just need to show that $\mathcal{L}'_c \supseteq \mathcal{L}_1 \setminus \mathcal{L}_2$. To do this, let $\boldsymbol{p} \in \mathcal{L}_1 \setminus \mathcal{L}_2$; then we prove that $\boldsymbol{p} \in \mathcal{L}'_c$. As $\boldsymbol{p} \notin \mathcal{L}_2$, for some $j = 1, \ldots, c$, $\boldsymbol{p} \notin \mathrm{gcon}\bigl(\{\beta_j\}\bigr)$. Consider the j-th iteration of the loop. Then the test on line (8) will succeed and the execution continues with the test on line (9). Moreover, as we know that line (13) will not be executed, this test must succeed so that $\boldsymbol{p} \in \mathrm{gcon}\bigl(\{2e_j \equiv_f 0\}\bigr)$ and lines (10-11) will be executed with $\beta = \beta_j$. As $\mathrm{gcon}\bigl(\{\beta_j\}\bigr)$ and $\mathrm{gcon}\bigl(\{2e_j \equiv_f f\}\bigr)$ are disjoint and their set union is the grid $\mathrm{gcon}\bigl(\{2e_j \equiv_f 0\}\bigr)$, $\boldsymbol{p}$ must satisfy the congruence $\bigl(2e_j - f \equiv_{2f} 0\bigr)$. Let $\mathcal{L}_{\beta_j} = \mathrm{gcon}\bigl(\mathcal{C}_1 \cup \{2e_j - f \equiv_{2f} 0\}\bigr)$ as on line (10). Then, as $\boldsymbol{p} \in \mathcal{L}_1$, we have $\boldsymbol{p} \in \mathcal{L}_{\beta_j}$; hence, after line (11), $\boldsymbol{p} \in \mathcal{L}'_j$. For each $i = j + 1, \ldots, c$, either $\mathcal{L}'_i = \mathcal{L}'_{i-1}$ or line (10) is executed, in which case $\mathcal{L}'_i \supseteq \mathcal{L}'_{i-1}$; hence $\boldsymbol{p} \in \mathcal{L}'_i$. In particular, $\boldsymbol{p} \in \mathcal{L}'_c$. As this holds for all $\boldsymbol{p} \in \mathcal{L}_1 \setminus \mathcal{L}_2$, $\mathcal{L}'_c \supseteq \mathcal{L}_1 \setminus \mathcal{L}_2$.

Finally we prove, by induction on i, that, for each $i = 0, \dots, c$, $\mathcal{L}'_i \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. Initially $\mathcal{L}'_0 = \varnothing$ and the result holds. Suppose now that $i > 0$ and that $\mathcal{L}'_{i-1} \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. If $\mathcal{L}_1 \subseteq \mathrm{gcon}\bigl(\{\beta_i\}\bigr)$, then $\mathcal{L}'_i = \mathcal{L}'_{i-1}$ is unchanged by the iteration. On the other hand, if $\mathcal{L}_1 \not\subseteq \mathrm{gcon}\bigl(\{\beta_i\}\bigr)$, the test on line (8) will succeed and the execution continues with the test on line (9). Moreover, as we know that line (13) will not be executed, this test must succeed so that $\mathcal{L}_1 \subseteq \mathrm{gcon}\bigl(\{2e_i \equiv_f 0\}\bigr)$. Let $\mathcal{L}_{\beta_i} = \mathrm{gcon}\bigl(\mathcal{C}_1 \cup \{2e_i - f \equiv_{2f} 0\}\bigr)$ as defined on line (10); then $\mathcal{L}_{\beta_i} \cap \mathcal{L}_2 = \varnothing$ so that $\mathcal{L}_{\beta_i} \subseteq \mathcal{L}_1 \setminus \mathcal{L}_2 \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. Since, on line (11), $\mathcal{L}'_i$ is assigned $\mathcal{L}'_{i-1} \oplus \mathcal{L}_{\beta_i}$, by definition of grid join and grid difference, $\mathcal{L}'_i \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. Therefore, letting $i = c$, we have $\mathcal{L}'_c \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. □

Proof (of Proposition 6). Let $\mathcal{C}$ be a congruence system for $\mathcal{L}$ and let $\mathcal{L}_R$ be the smallest rectilinear grid that contains $\mathcal{L}$. Suppose also that $\mathcal{B}$ defines the rectilinear grid $\mathcal{L}_B$. Then we need to show that $\mathcal{L}_B = \mathcal{L}_R$. Let $\mathcal{C}_B$ be the congruence system for $\mathcal{L}_B$ defined by equation (2) and $\mathcal{C}_R$ be congruence system in minimal form for $\mathcal{L}_R$. Observe that $\mathcal{L}_R = \mathrm{ggen}(\mathcal{G}_R)$ where $\mathcal{G}_R$ is a minimal form for the generator system

$$\bigl(\{\ell_i \boldsymbol{e}_i | \boldsymbol{\ell} \in L, \ell_i \neq 0\}, \{q_i \boldsymbol{e}_i | \boldsymbol{q} \in Q, q_i \neq 0\}, \{\boldsymbol{p}\}\bigr).$$

To prove the result, we consider each dimension i where $1 \leq i \leq n$ separately and show that there exists $\gamma_i = \bigl(\langle \boldsymbol{e}_i, \boldsymbol{x} \rangle \equiv_f b\bigr) \in \mathcal{C}_R$ for some $f, b \in \mathbb{Q}$ if and only if β_i exists and $\mathrm{gcon}(\gamma_i) = \mathrm{gcon}(\beta_i)$. We consider each of the three cases (1), (2) and (3) separately.

Suppose, for some $\boldsymbol{\ell} \in L$, $\ell_i \neq 0$. Then case (1) applies and $I_i = (0, 0)$. Moreover, there is a line $\boldsymbol{e}_i \ell_i$ in $\mathcal{G}_R$. Thus, there are no congruences $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \equiv_f b$ in $\mathcal{C}_B$ or $\mathcal{C}_R$ where $\mathrm{piv}_<(\boldsymbol{a}) = i$.

Suppose, for all $\boldsymbol{\ell} \in L$, $\ell_i = 0$, and $q_{1i} = \cdots = q_{qi} = 0$. Then case (2) applies and $I_i = (p_i, \infty)$. Therefore there exists $\beta_i = \bigl(\langle \boldsymbol{e}_i, \boldsymbol{x} \rangle = p_i\bigr) \in \mathcal{C}_B$. Moreover, there are no lines or parameters in $\mathcal{G}_R$ with a non-zero i-th coordinate. Thus there exists $\gamma_i \in \mathcal{C}_R$ where $f = 0$ and $b = p_i$ so that $\mathrm{gcon}(\gamma_i) = \mathrm{gcon}(\beta_i)$.

Suppose, for all $\boldsymbol{\ell} \in L$, $\ell_i = 0$, and, for some $j = 1, \dots, q$, $q_{ji} \neq 0$. Then case (3) applies and $I_i = (p_i, p_i + |g|)$ where $g = \gcd\bigl(\{q_{1i}, \dots, q_{qi}\}\bigr) \neq 0$. Therefore there exists $\beta_i = \bigl(\langle \boldsymbol{e}_i, \boldsymbol{x} \rangle \equiv_g p_i\bigr) \in \mathcal{C}_B$. Moreover, there are no lines in $\mathcal{G}_R$ with a non-zero i-th coordinate. Thus there exists $\gamma_i \in \mathcal{C}_R$ where $f = g$ and $b = p_i + mg$ for some $m \in \mathbb{Z}$. Thus $\mathrm{gcon}(\gamma_i) = \mathrm{gcon}(\beta_i)$. □

In the following lemma, letting $H \in \mathbb{R}^{n \times n}$, we say that H is *upper triangular form* if, for all $i = 1, \dots, n$, $H_{ii} \neq 0$ and, for all j where $1 \leq j < i \leq n$, $H_{ij} = 0$. Similarly, a matrix H is in *lower triangular form* if, for all $i = 1, \dots, n$, $H_{ii} \neq 0$ and, for all j where $1 \leq i < j \leq n$, $H_{ij} = 0$. Thus the transpose of an upper triangular matrix is lower triangular and vice-versa.

Lemma 1. *There exists a computable, invertible function that converts a generator system $\mathcal{G} = \bigl(L, Q, \{\boldsymbol{p}\}\bigr)$ in $\mathbb{Q}^n$ in minimal form to a congruence system $\mathcal{C} = \mathcal{E} \cup \mathcal{F}$ in $\mathbb{Q}^n$ in minimal form where $\mathcal{E}$ are equalities and $\mathcal{F}$ are proper congruences and such that*

1. *$\# Q = \# \mathcal{F} = n - \# L - \# \mathcal{E}$;*
2. *there exists $\boldsymbol{q} \in Q$ if and only if there exists $\beta = \bigl(\langle \boldsymbol{a}, \boldsymbol{x} \rangle \equiv_1 0\bigr) \in \mathcal{F}$, such that, for some $k \in \{1, \dots, n\}$, $\operatorname{piv}_>(\boldsymbol{q}) = \operatorname{piv}_<(\boldsymbol{a}) = k$ and $q_k a_k = 1$;*
3. *for all $\boldsymbol{\ell} \in L$ and $\beta = \bigl(\langle \boldsymbol{a}, \boldsymbol{x} \rangle = 0\bigr) \in \mathcal{E}$, $\operatorname{piv}_>(\boldsymbol{\ell}) \neq \operatorname{piv}_<(\boldsymbol{a})$;*
4. *$\operatorname{gcon}(\mathcal{C}) = \operatorname{ggen}(\mathcal{G})$.*

Proof. Let $\hat{\mathcal{G}}$ be the homogeneous form for $\mathcal{G}$ and $\hat{\mathcal{C}} = \hat{\mathcal{F}} \cup \hat{\mathcal{E}}$ be the homogeneous form for $\mathcal{C}$ where $\mathcal{G}$ is as defined in equation (5) and $\hat{\mathcal{E}}$ and $\hat{\mathcal{F}}$ are as defined in equations (3) and (4) in Section 5. Let $\hat{\mathcal{E}} = \hat{E}^{\mathrm{T}} \hat{\boldsymbol{x}} = \mathbf{0}$ and $\hat{\mathcal{F}} = \hat{F}^{\mathrm{T}} \hat{\boldsymbol{x}} \equiv_1 \mathbf{0}$ where $\hat{\boldsymbol{x}} = (x_0, \dots, x_n)^{\mathrm{T}}$. Then, as $\mathcal{G}$ and $\mathcal{C}$ are in minimal form, by the definition of homogeneous form, items (1), (2), (3), and (4) hold if and only if the following hold:

5. $\# \hat{Q} = \# \hat{\mathcal{F}} = n + 1 - \# \hat{L} - \# \hat{\mathcal{E}}$;
6. there exists $\boldsymbol{q} \in \hat{Q}$ if and only if there exists $\beta = \bigl(\langle \boldsymbol{a}, \hat{\boldsymbol{x}} \rangle \equiv_1 0\bigr) \in \hat{\mathcal{F}}$, such that, for some $k \in \{0, \dots, n\}$, $\operatorname{piv}_>(\boldsymbol{q}) = \operatorname{piv}_<(\boldsymbol{a}) = k$ and $q_k a_k = 1$;
7. for all $\boldsymbol{\ell} \in \hat{L}$ and $\beta = \bigl(\langle \boldsymbol{a}, \hat{\boldsymbol{x}} \rangle = 0\bigr) \in \hat{\mathcal{E}}$, $\operatorname{piv}_>(\boldsymbol{\ell}) \neq \operatorname{piv}_<(\boldsymbol{a})$;
8. $\operatorname{gcon}(\hat{\mathcal{C}}) = \operatorname{ggen}(\hat{\mathcal{G}})$

It therefore remains to prove that (5), (6), (7), and (8) hold.

Let M, N be matrices in $\mathbb{Z}^{n+1}$, whose vectors are of the form $\boldsymbol{e}_i$, $1 \leq i \leq n$, and such that $(\hat{L}, \hat{Q}, M)$ and $(N, \hat{F}, \hat{E})$ are square and nonsingular. Then we let

$$(N, \hat{F}, \hat{E}) := \bigl((\hat{L}, \hat{Q}, M)^{-1}\bigr)^{\mathrm{T}}$$

where $\# N = \# \hat{L}$, $\# \hat{Q} = \# \hat{F}$ and $\# \hat{E} = \# N$; ensuring (5) holds. Note that matrix inversion and transposition are both computable and invertible.

Let $\ell = \# \hat{L}$, $q = \# \hat{Q} - 1$ and $m = \# M$. Then

$$\begin{aligned}
\hat{\boldsymbol{x}} \in \hat{\mathcal{G}} &\iff \hat{\boldsymbol{x}} = \hat{Q}\boldsymbol{\pi} + \hat{L}\boldsymbol{\lambda} + M\mathbf{0}, && \text{for } \boldsymbol{\lambda} \in \mathbb{R}^{\ell}, \boldsymbol{\pi} \in \mathbb{Z}^{q+1}, \mathbf{0} \in \mathbb{Z}^{m} \\
&\iff \hat{\boldsymbol{x}} = (\hat{Q}, \hat{L}, M)(\boldsymbol{\pi}^{\mathrm{T}}, \boldsymbol{\lambda}^{\mathrm{T}}, \mathbf{0}^{\mathrm{T}})^{\mathrm{T}}, && \text{for } \boldsymbol{\lambda} \in \mathbb{R}^{\ell}, \boldsymbol{\pi} \in \mathbb{Z}^{q+1}, \mathbf{0} \in \mathbb{Z}^{m} \\
&\iff (\hat{Q}, \hat{L}, M)^{-1}\hat{\boldsymbol{x}} = (\boldsymbol{\pi}^{\mathrm{T}}, \boldsymbol{\lambda}^{\mathrm{T}}, \mathbf{0}^{\mathrm{T}})^{\mathrm{T}}, && \text{for } \boldsymbol{\lambda} \in \mathbb{R}^{\ell}, \boldsymbol{\pi} \in \mathbb{Z}^{q+1}, \mathbf{0} \in \mathbb{Z}^{m}.
\end{aligned}$$

As $\hat{\mathcal{G}}$ is in minimal form (so that the set of vectors in $(\hat{L}, \hat{Q})$ is a subset of the set of vectors in a triangular matrix) and M adds the missing columns with only the diagonal elements being non-zero, $(\hat{Q}, \hat{L}, M)$ is a permutation of a lower triangular matrix. Thus the transposed inverse $(N, \hat{F}, \hat{E})$ is the same permutation of an upper triangular matrix so that $\hat{\mathcal{C}}$ is also a congruence system in minimal form. The generating system $\hat{\mathcal{G}}$ is the homogeneous form of a system that contains a point if and only if there is an element $\boldsymbol{q} \in \hat{Q}$ such that $q_0 = 1$ and $\operatorname{piv}_>(\boldsymbol{q}) = 1$; that is, if and only if there is also an element $\boldsymbol{a} \in \hat{F}$ such that $a_0 = 1$ and $\operatorname{piv}_<(\boldsymbol{a}) = 1$. Thus $\hat{\mathcal{C}}$ is a congruence system in minimal homogeneous form if and only if $\hat{\mathcal{G}}$ is a generator system in minimal homogeneous form.

More generally, for vectors $\boldsymbol{v}_i$ and $\boldsymbol{w}_i$ in $(N, \hat{F}, \hat{E})$ and $(\hat{L}, \hat{Q}, M)$, respectively, $\operatorname{piv}_<(\boldsymbol{v}_i) = \operatorname{piv}_>(\boldsymbol{w}_i) = k$ and $v_{ik} w_{ik} = 1$. Since $\# N = \# \hat{L}$, $\# \hat{Q} = \# \hat{F}$ and $\# \hat{E} = \# M$ we have proved items (6) and (7).

Since $(N, \hat{F}, \hat{E}) = \big((\hat{Q}, \hat{L}, M)^{-1}\big)^{\mathrm{T}}$ we have that $\hat{\boldsymbol{x}} \in \operatorname{ggen}(\hat{\mathcal{G}})$ if and only if $N^{\mathrm{T}}\hat{\boldsymbol{x}} = \boldsymbol{\lambda}$, $\hat{F}^{\mathrm{T}}\hat{\boldsymbol{x}} = \boldsymbol{\pi}$, $\hat{E}^{\mathrm{T}}\hat{\boldsymbol{x}} = \mathbf{0}$ for some $\boldsymbol{\lambda} \in \mathbb{R}^{\ell}, \boldsymbol{\pi} \in \mathbb{Z}^{q+1}$. Since $\boldsymbol{\lambda} \in \mathbb{R}^{\ell}$, the condition $N^{\mathrm{T}}\hat{\boldsymbol{x}} = \boldsymbol{\lambda}$ is vacuous and, as $\boldsymbol{\pi} \in \mathbb{Z}^{q+1}$, the condition $\hat{F}^{\mathrm{T}}\hat{\boldsymbol{x}} = \boldsymbol{\pi}$ holds if and only if $\hat{F}^{\mathrm{T}}\hat{\boldsymbol{x}} \equiv_1 \mathbf{0}$. Thus we have that $\hat{\boldsymbol{x}} \in \operatorname{ggen}(\hat{\mathcal{G}})$ if and only if $\hat{F}^{\mathrm{T}}\hat{\boldsymbol{x}} \equiv_1 \mathbf{0}$, $\hat{E}^{\mathrm{T}}\hat{\boldsymbol{x}} = \mathbf{0}$. Therefore $\operatorname{gcon}(\hat{\mathcal{C}}) = \operatorname{ggen}(\hat{\mathcal{G}})$ and (8) holds. □

Proof (of Propositions 7 and 8). These are direct consequences of Lemma 1. □

Proof (of Proposition 9). Suppose that $\mathcal{C}$ is not in strong minimal form. Then, by Definition 5, there exists a proper congruence $\beta = \big(\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_1 b\big) \in \mathcal{C}$, such that the following holds:

1. there exists $i > 0$ and a proper congruence $\gamma = \big(\langle \boldsymbol{c}, \boldsymbol{x}\rangle \equiv_1 d\big) \in \mathcal{C} \setminus \{\beta\}$ where $\operatorname{piv}_<(\gamma) = i$ and either $2a_i \leq -c_i$ or $2a_i > c_i$.

Suppose that $0 \leq k \leq n$ is the maximum value for the index i such that condition (1) holds.

We show, by induction on k, that there exists a sequence of at most n transformations, each of which having complexity $\mathrm{O}\big(n\big)$, from β to the congruence $\beta' = \big(\langle \boldsymbol{a}', \boldsymbol{x}\rangle \equiv_1 b'\big)$, such that, if $\mathcal{C}' := \big(\mathcal{C} \setminus \{\beta\}\big) \cup \{\beta'\}$, then $\operatorname{gcon}(\mathcal{C}') = \operatorname{gcon}(\mathcal{C})$ and condition (1) (when β is replaced by β') does not hold.

If $k = 0$, then condition (1) does not hold for β. Therefore let $\beta' = \beta$.

Suppose now that $k > 0$ so that condition (1) holds for $i = k$. As $\mathcal{C}$ is in minimal form, $k < \operatorname{piv}_<(\boldsymbol{a})$. Let

$$\boldsymbol{a}'' = \begin{cases} \boldsymbol{a} - \left\lceil \frac{a_k}{c_k} \right\rceil \boldsymbol{c} \quad \text{and} \quad b'' = b - \left\lceil \frac{a_k}{c_k} \right\rceil d, & \text{if } a_k \bmod c_k > \frac{a_k}{2}; \\ \boldsymbol{a} - \left\lfloor \frac{a_k}{c_k} \right\rfloor \boldsymbol{c} \quad \text{and} \quad b'' = b - \left\lfloor \frac{a_k}{c_k} \right\rfloor d, & \text{if } a_k \bmod c_k \leq \frac{a_k}{2}. \end{cases}$$

Then $-c_k < 2a''_k \leq c_k$. Also, for $k+1 \leq j \leq n$, we have $c_j = 0$ so that $a_j = a''_j$ and $\operatorname{piv}_<(\boldsymbol{a}'') = \operatorname{piv}_<(\boldsymbol{a})$. Letting $\beta'' := \big(\langle \boldsymbol{a}'', \boldsymbol{x}\rangle \equiv_1 b''\big)$ and $\mathcal{C}'' := \big(\mathcal{C} \setminus \{\beta\}\big) \cup \{\beta''\}$, we have $\operatorname{gcon}(\mathcal{C}'') = \operatorname{gcon}(\mathcal{C})$. Note that this transformation has a complexity $\mathrm{O}\big(n\big)$. As k'', the maximum index such that condition (1) holds for β'', is strictly less than k we can apply the inductive hypothesis to $\mathcal{C}''$ and β''. Thus there is a sequence of at most $n-1$ transformations from β'' to β' such that, $\operatorname{gcon}\Big(\big(\mathcal{C}'' \setminus \{\beta''\}\big) \cup \{\beta'\}\Big) = \operatorname{gcon}(\mathcal{C}'')$ and condition (1) (when β is replaced by β') does not hold. Thus there is a sequence of at most n transformations from β to β' such that $\operatorname{gcon}\Big(\big(\mathcal{C} \setminus \{\beta\}\big) \cup \{\beta'\}\Big) = \operatorname{gcon}(\mathcal{C})$. As each of the individual steps has complexity $\mathrm{O}\big(n\big)$, the sequence of transformations has complexity $\mathrm{O}\big(n^2\big)$.

We repeat this sequence of transformations for each proper congruence in $\mathcal{C}$ to obtain a congruence system $\mathcal{C}'$ such that, for each proper congruence $\beta' \in \mathcal{C}'$, condition (1) does not hold. Thus, by Definition 5, $\mathcal{C}'$ is in strong normal form. Thus, as there are at most n proper congruences in $\mathcal{C}$ since, by hypothesis, $\mathcal{C}$ is in minimal form, the complexity of computing the strong minimal form is $\mathrm{O}\big(n^3\big)$. □

The proofs of Propositions 10 and 11 depend on the following lemma. This shows that if one grid is a subset of another then the pivot elements of the proper congruences of the larger grid must be divisible by the corresponding pivot elements of the smaller grid.

Lemma 2. *Let $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1)$, $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2)$ be non-empty grids in $\mathbb{G}_n$ such that $\mathcal{L}_1 \subseteq \mathcal{L}_2$ and the congruence systems $\mathcal{C}_1$ and $\mathcal{C}_2$ are in minimal form. Then, for each $\gamma = \bigl(\langle \boldsymbol{c}, \boldsymbol{x}\rangle \equiv_g d\bigr) \in \mathcal{C}_2$, there exists $\beta = \bigl(\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_f b\bigr) \in \mathcal{C}_1$ such that $\mathrm{piv}_<(\boldsymbol{a}) = \mathrm{piv}_<(\boldsymbol{c}) = k$ and either $f = g = 0$ or $g \neq 0$ and $ga_k \mid fc_k$.*

Proof. Suppose $\gamma = \bigl(\langle \boldsymbol{c}, \boldsymbol{x}\rangle \equiv_g d\bigr) \in \mathcal{C}_2$ and $k = \mathrm{piv}_<(\boldsymbol{c})$. Then, as $\mathcal{L}_1 \subseteq \mathcal{L}_2$, $\mathcal{L}_1 \subseteq \mathrm{gcon}\bigl(\{\gamma\}\bigr)$. Let $\mathcal{G}_1 = \bigl(L_1, Q_1, \{\boldsymbol{p}\}\bigr)$ be a generator system for $\mathcal{L}_1$ in minimal form constructed as in Lemma 1 from $\mathcal{C}_1$.

We first prove that there exists $\beta = \bigl(\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_f b\bigr) \in \mathcal{C}_1$ such that $\mathrm{piv}_<(\boldsymbol{a}) = k$. To see this, suppose instead that, for all $\beta = \bigl(\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_f b\bigr) \in \mathcal{C}_1$, $\mathrm{piv}_<(\boldsymbol{a}) \neq k$. Then, by Lemma 1, there must exist a line $\boldsymbol{\ell} \in L_1$ such that $\mathrm{piv}_>(\boldsymbol{\ell}) = k$; hence $\langle \boldsymbol{c}, \boldsymbol{\ell}\rangle = c_k\ell_k \neq 0$. Since $\mathcal{L}_1 \subseteq \mathrm{gcon}\bigl(\{\gamma\}\bigr)$, this implies that

$$\bigl\langle \boldsymbol{c}, (\boldsymbol{p} + r\boldsymbol{\ell})\bigr\rangle = \langle \boldsymbol{c}, \boldsymbol{p}\rangle + rc_k\ell_k \equiv_g d,$$

for all $r \in \mathbb{R}$; which is a contradiction.

We next show that if $g = 0$ then $f = 0$. To see this, suppose instead that $g = 0$ but $f \neq 0$. Then, by Lemma 1, there exists $\boldsymbol{q} \in Q_1$ such that $\mathrm{piv}_>(\boldsymbol{q}) = k$; hence $\langle \boldsymbol{c}, \boldsymbol{q}\rangle = c_kq_k \neq 0$. Since $\mathcal{L}_1 \subseteq \mathrm{gcon}\bigl(\{\gamma\}\bigr)$, this implies that

$$\bigl\langle \boldsymbol{c}, (\boldsymbol{p} + r\boldsymbol{q})\bigr\rangle = \langle \boldsymbol{c}, \boldsymbol{p}\rangle + mc_kq_k \equiv_g d,$$

for all $m \in \mathbb{Z}$; which is a contradiction.

We now assume that $g \neq 0$ and show that $ga_k \mid fc_k$. This is trivial if $f = 0$; therefore, suppose $f \neq 0$. By Lemma 1, there exists a parameter $\boldsymbol{q}$ in Q_1 such that $\mathrm{piv}_>(\boldsymbol{q}) = k$ (so that $q_kc_k \neq 0$) and $q_k = fa_k^{-1}$. Thus, as $\mathcal{L}_1 \subseteq \mathrm{gcon}\bigl(\{\gamma\}\bigr)$, $\langle \boldsymbol{q}, \boldsymbol{c}\rangle = q_kc_k = mg$, for some $m \in \mathbb{Z} \setminus \{0\}$. Therefore we must have $ga_k \mid fc_k$. □

Proof (of Proposition 4). Let $\mathcal{C}'$ be the congruence system obtained, as in Lemma 1, from $\mathcal{G}$. Let $\mathcal{G} = (L, Q, P)$ and let $\mathcal{C} = (\mathcal{F}, \mathcal{E})$ and $\mathcal{C}' = (\mathcal{F}', \mathcal{E}')$ where $\mathcal{E}, \mathcal{E}'$ are sets of equalities and $\mathcal{F}, \mathcal{F}'$ are sets of proper congruences. Then, by Lemma 1,

$$\# Q = \# \mathcal{F}' = n - \# L - \# \mathcal{E}'.$$

By applying Lemma 2 twice where $\mathcal{L}_1 = \mathcal{L}_2 = \mathcal{L}$, we obtain $\# \mathcal{E} = \# \mathcal{E}'$ and $\# \mathcal{F} = \# \mathcal{F}'$. Therefore

$$\# Q = \# \mathcal{F} = n - \# L - \# \mathcal{E}.$$

□

The next lemma, which is needed in the proofs of Propositions 2, 10 and 11 shows that if, two grids, one a subset of the other are described by two congruence systems in strong minimal form that are pivot equivalent, then, relative to the affine hull of the grids, pivot equivalent congruences in these systems are the same.

Lemma 3. *Let $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1)$, $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2)$ be non-empty grids in $\mathbb{G}_n$ where $\mathcal{L}_1 \subseteq \mathcal{L}_2$ and the congruence systems $\mathcal{C}_1$ and $\mathcal{C}_2$ are in strong minimal form. Suppose that $\mathcal{C}_1$ is pivot equivalent to $\mathcal{C}_2$. Then, for each $\beta \in \mathcal{C}_1$ and $\gamma \in \mathcal{C}_2$ such that $\beta \Uparrow \gamma$,*

$$\mathrm{gcon}\bigl(\{\beta\}\bigr) \cap \mathrm{affine.hull}(\mathcal{L}_1) = \mathrm{gcon}\bigl(\{\gamma\}\bigr) \cap \mathrm{affine.hull}(\mathcal{L}_1). \qquad (10)$$

Proof. Let $\beta = \bigl(\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_f b\bigr) \in \mathcal{C}_1$. By the definition of pivot equivalence for congruence systems in Section 3, as $\mathcal{C}_1 \Uparrow \mathcal{C}_2$ there exists $\gamma = \bigl(\langle \boldsymbol{c}, \boldsymbol{x}\rangle \equiv_g d\bigr) \in \mathcal{C}_2$ such that $\beta \Uparrow \gamma$. We show that equation (10) holds. By the definition of pivot equivalence for congruences in Section 2, $\mathrm{piv}_<(\boldsymbol{a}) = \mathrm{piv}_<(\boldsymbol{c}) = k$ and $ga_k = fc_k$. Thus as $a_k, c_k \neq 0$, either $f = g = 0$ and β, γ are both equalities, or we have $f, g \neq 0$ so that β, γ are both proper congruences and we can assume that $f = g = 1$.

Let $\mathcal{E}_1$ be the set of equalities in $\mathcal{C}_1$. By Gaussian elimination, the set $\mathcal{E}_1$ can be transformed to the set of equalities $\mathcal{E}_1'$ such that $\mathrm{gcon}(\mathcal{E}_1') = \mathrm{gcon}(\mathcal{E}_1) = \mathrm{affine.hull}(\mathcal{L}_1)$ and has the following property: let $\mathcal{E}_1' = \{\beta_1, \ldots, \beta_m\}$ such that, for each $i \in \{1, \ldots, m\}$, $\mathrm{piv}_<(\beta_i) = k_i$ and $\beta_i = \bigl(\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle = b_i\bigr)$; then, for each $i, j \in \{1, \ldots, m\}$ where $i \neq j$, we have $a_{ik_j} = 0$. Let

$$\boldsymbol{a}'' = \boldsymbol{a} - \boldsymbol{c} - \sum_{i=1}^{m} \frac{(a_{k_i} - c_{k_i})}{a_{k_i}} \boldsymbol{a}_i, \qquad b'' = b - d - \sum_{i=1}^{m} \frac{(a_{k_i} - c_{k_i})}{a_{k_i}} b_i. \qquad (11)$$

Let $\beta'' := \bigl(\langle \boldsymbol{a}'', \boldsymbol{x}\rangle \equiv_1 b''\bigr)$ and $\ell := \mathrm{piv}_<(\boldsymbol{a}'')$. Then $\mathcal{L}_1 \subseteq \mathrm{gcon}\bigl(\{\beta''\}\bigr) \subseteq \mathcal{L}_2$. Moreover, for any equality $\beta_i \in \mathcal{E}_1'$, $\mathrm{piv}_<(\boldsymbol{a}_i) \neq \ell$. Thus, if β, γ are equalities, $\boldsymbol{a''} = \boldsymbol{0}$ and, as $\mathcal{C}_1$ is consistent, $b'' = 0$. Therefore $\mathrm{gcon}(\mathcal{E}_1') \subseteq \mathrm{gcon}\bigl(\{\beta\}\bigr)$ and $\mathrm{gcon}(\mathcal{E}_1') \subseteq \mathrm{gcon}\bigl(\{\gamma\}\bigr)$. Hence equation (10) holds.

Consider now the case when β, γ are proper congruences. We first show that $\boldsymbol{a''} = \boldsymbol{0}$ and $b'' \in \mathbb{Z}$. Without loss of generality we can assume that $f = g = 1$. Note that, as $a_k = c_k$ we have $\ell < k$. We show $\ell = 0$; suppose, to the contrary that $\ell > 0$. Since $\mathcal{L}_1 \subseteq \mathcal{L}_2$, and $\gamma \in \mathcal{C}_2$, we have $\mathcal{L}_1 \subseteq \mathrm{gcon}\bigl(\{\gamma\}\bigr)$; so we can apply Lemma 2 to the grids $\mathcal{L}_1$ and $\mathrm{gcon}\bigl(\{\gamma\}\bigr)$. Thus there exists a proper congruence $\beta' = \bigl(\langle \boldsymbol{a}', \boldsymbol{x}\rangle \equiv_1 b'\bigr) \in \mathcal{C}_1$ where $\mathrm{piv}_<(\boldsymbol{a}') = \ell$ and $a_\ell' \mid a_\ell''$. Note that the number of proper congruences p_1 in $\mathcal{C}_1$ is equal to the number of proper congruences p_2 in $\mathcal{C}_2$; since by Lemma 2, $p_2 \leq p_1$ and, by hypothesis, $p_1 \leq p_2$. Therefore, by Lemma 2, there must exist a proper congruence $\gamma' = \bigl(\langle \boldsymbol{c}', \boldsymbol{x}\rangle \equiv_1 d'\bigr) \in \mathcal{C}_2$ where $\mathrm{piv}_<(\boldsymbol{c}') = \ell$ and $c_\ell' = a_\ell'$. Now as $\mathcal{C}_1$ and $\mathcal{C}_2$ are in strong minimal form, by Definition 5,

$$-\frac{a_\ell'}{2} < a_\ell \leq \frac{a_\ell'}{2} \quad \text{and} \quad -\frac{c_\ell'}{2} < c_\ell \leq \frac{c_\ell'}{2}.$$

Therefore $-a'_\ell < a''_\ell < a'_\ell$. It follows that, as $a'_\ell | a''_\ell$, $a''_\ell = 0$, contradicting the assumption that $\operatorname{piv}_<(\boldsymbol{a}'') = \ell > 0$. Therefore $\boldsymbol{a}'' = \boldsymbol{0}$ and β'' is the relation $b'' \equiv_1 0$ for some $b'' \in \mathbb{Z}$.

It follows that, by (11),

$$\boldsymbol{a} - \boldsymbol{c} = \sum_{i=1}^{m} \frac{(a_{k_i} - c_{k_i})}{a_{k_i}} \boldsymbol{a}_i, \qquad b - d \equiv_1 \sum_{i=1}^{m} \frac{(a_{k_i} - c_{k_i})}{a_{k_i}} b_i.$$

Thus

$$\operatorname{gcon}\bigl(\{\gamma, \beta_1, \dots, \beta_m\}\bigr) \subseteq \operatorname{gcon}\bigl(\{\beta\}\bigr) \quad \operatorname{gcon}\bigl(\{\beta, \beta_1, \dots, \beta_m\}\bigr) \subseteq \operatorname{gcon}\bigl(\{\gamma\}\bigr)$$

so that

$$\operatorname{gcon}\bigl(\{\gamma, \beta_1, \dots, \beta_m\}\bigr) = \operatorname{gcon}\bigl(\{\beta, \beta_1, \dots, \beta_m\}\bigr).$$

Hence equation (10) holds. $\square$

Proof (of Proposition 2). By Proposition 9, we can convert $\mathcal{C}_1$ and $\mathcal{C}_2$ to strong minimal form, $\mathcal{C}'_1$ and $\mathcal{C}'_2$ respectively, so that, for $i = 1, 2$, $\mathcal{L}_i = \operatorname{gcon}(\mathcal{C}'_i)$ and $\mathcal{C}'_i$ is pivot equivalent to $\mathcal{C}_i$. By Lemma 3, for each $\beta \in \mathcal{C}'_1$ and $\gamma \in \mathcal{C}'_2$,

$$\operatorname{gcon}\bigl(\{\beta\}\bigr) \cap \operatorname{affine.hull}(\mathcal{L}_1) = \operatorname{gcon}\bigl(\{\gamma\}\bigr) \cap \operatorname{affine.hull}(\mathcal{L}_1).$$

Thus $\mathcal{L}_1 = \mathcal{L}_2$ as required. $\square$

Proof (of Proposition 10). Suppose $\mathcal{C}_1, \mathcal{C}_2 \neq \varnothing$. By Lemma 2, for each $\beta = \bigl(\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_f b\bigr) \in \mathcal{C}_1$ there exists $\gamma = \bigl(\langle \boldsymbol{c}, \boldsymbol{x}\rangle \equiv_g d\bigr) \in \mathcal{C}_2$ such that $\operatorname{piv}_<(\boldsymbol{a}) = \operatorname{piv}_<(\boldsymbol{c}) = k$ and either $f = g = 0$ or $f \neq 0$ and $fc_k \mid ga_k$.

Also by Lemma 2, for $\gamma = \bigl(\langle \boldsymbol{c}, \boldsymbol{x}\rangle \equiv_g d\bigr) \in \mathcal{C}_2$ there exists $\beta' = \bigl(\langle \boldsymbol{a}', \boldsymbol{x}\rangle \equiv_{f'} b'\bigr) \in \mathcal{C}_1$, such that $\operatorname{piv}_<(\boldsymbol{a}') = \operatorname{piv}_<(\boldsymbol{c}) = k$ and either $f' = g = 0$ or $g \neq 0$ and $ga'_k \mid f'c_k$. However as $\mathcal{C}_1$ is in strong minimal form $\beta = \beta'$. Therefore, by Definition 2, $a_k, c_k > 0$, hence $fc_k = ga_k$.

Therefore, for each $\beta \in \mathcal{C}_1$ there exists $\gamma \in \mathcal{C}_2$ such that $\beta \Uparrow \gamma$ so that we can apply Lemma 3 to prove the thesis. $\square$

Proof (of Proposition 11). In order to show that '∇' is a widening operator, we prove that conditions (1) and (2) in Definition 4 hold. Let $\mathcal{L}_1 = \operatorname{gcon}(\mathcal{C}_1)$, $\mathcal{L}_2 = \operatorname{gcon}(\mathcal{C}_2) \in \mathbb{G}_n$, where $\mathcal{L}_1 \subseteq \mathcal{L}_2$, $\mathcal{C}_1$ is in minimal form and $\mathcal{C}_2$ is in strong minimal form.

By Definition 6, if $\mathcal{L}_1 = \varnothing$ or $\dim(\mathcal{L}_1) < \dim(\mathcal{L}_2)$, then $\mathcal{L}_1 \nabla \mathcal{L}_2 = \mathcal{L}_2$. Therefore, in this case, condition (1) holds. Clearly, the empty grid can occur only as the first element of a strictly increasing chain of grids; moreover, if $\mathcal{L}$ and $\mathcal{L}'$ are any two successive and distinct grids in the increasing chain of condition (2) in Definition 4, then $0 \leq \dim(\mathcal{L}) \leq \dim(\mathcal{L}') \leq n$. Hence, the case when $\mathcal{L}_1 = \varnothing$ or $\dim(\mathcal{L}_1) < \dim(\mathcal{L}_2)$ hold can occur no more than a finite number of times in such a chain.

Suppose now that $\mathcal{L}_1 \neq \varnothing$ and $\dim(\mathcal{L}_1) = \dim(\mathcal{L}_2)$, so that the second case of the widening computation applies (note that, due to the inclusion hypothesis, $\dim(\mathcal{L}_1) > \dim(\mathcal{L}_2)$ cannot hold), and let $\mathcal{C}_S$ be as given in equation (6) in Definition 6. Then, since $\mathcal{C}_S \subseteq \mathcal{C}_2$, condition (1) holds. By Proposition 2, if $\mathcal{C}_S = \mathcal{C}_2$, we have $\mathcal{L}_1 = \mathcal{L}_2$; thus, if $\mathcal{L}_1 \neq \mathcal{L}_2$, we have $\#\mathcal{C}_S < \#\mathcal{C}_2$. By Lemma 2, as $\mathcal{C}_1$ and $\mathcal{C}_2$ are in minimal form, it follows that $\#\mathcal{C}_2 \leq \#\mathcal{C}_1$ so that, if $\mathcal{L}_1 \neq \mathcal{L}_2$, $\#\mathcal{C}_S < \#\mathcal{C}_1$. Therefore condition (2) of Definition 4 holds. $\square$