A Few Graph-Based Relational Numerical Abstract Domains

Antoine Miné

A Few Graph-Based Relational Numerical Abstract Domains*

Antoine Miné

École Normale Supérieure de Paris, France,
mine@di.ens.fr,
http://www.di.ens.fr/~mine

Abstract This article presents the systematic design of a class of relational numerical abstract domains from non-relational ones. Constructed domains represent sets of invariants of the form $(v_j - v_i \in C)$, where v_j and v_i are two variables, and C lives in an abstraction of $\mathcal{P}(\mathbb{Z})$, $\mathcal{P}(\mathbb{Q})$, or $\mathcal{P}(\mathbb{R})$. We will call this family of domains *weakly relational domains*. The underlying concept allowing this construction is an extension of potential graphs and shortest-path closure algorithms in exotic-like algebras.

Example constructions are given in order to retrieve well-known domains as well as new ones. Such domains can then be used in the Abstract Interpretation framework in order to design various static analyses. A major benefit of this construction is its modularity, allowing to quickly implement new abstract domains from existing ones.

1 Introduction

Proving the correctness of a program is essential, especially for critical and embedded applications (such as planes, rockets, and so on). Among several correctness criteria, one should ensure that a program can never perform a run-time error (divide by zero, overflow, etc.). A classical method consists in finding a *safety invariant* before each dangerous operation in the program, and checking that the invariant implies the good behavior of the subsequent operation. Because this task is to be performed on the whole program—containing maybe tens of thousands of lines—and must be repeated after even the slightest code modification, we need a purely automatic static analysis approach.

Discovering the tightest invariants of a program cannot be fully mechanized in general, so we have to find some kind of sound approximation. By sound, we mean that the analysis should find an over-approximation of the real invariant. We will always discover *all* bugs in a program. However, we may find false alarms.

* This work was supported in part by the RTD project IST-1999-20527 "DAEDALUS" of the European IST FP5 program.

2 Previous Work

We will work in the well-known *Abstract Interpretation* framework, proposed by Cousot and Cousot in [6, 7], which allows us to easily describe sound and computable semantics approximations.

2.1 Numerical Abstract Domains

The crux of the method is to design a so-called *abstract domain*, that is to say, a practical representation of the invariants we want to study, together with a fixed set of operators and transfer functions (union, intersection, widening, assignment, guard, etc.) used to mimic the semantics of the programming language.

We will consider here *numerical abstract domains*. Given the set $\mathcal{V}$ of the numerical variables of a program with value in the set $\mathbb{I}$ (that can be $\mathbb{Z}$, $\mathbb{Q}$ or $\mathbb{R}$), a numerical abstract domain will represent and manipulate subsets of $\mathcal{V} \mapsto \mathbb{I}$. Well-known *non-relational domains* include the *interval* domain [5] (describing invariants of the form $v_i \in [c_1, c_2]$), the *constant propagation* domain ($v_i = c$), and the *congruence* domain [14] ($v_i \in a\mathbb{Z} + b$). Well-known *relational domains* include the *polyhedron* domain [9] ($\alpha_1 v_1 + \cdots + \alpha_n v_n \leq c$), the *linear equality* domain [18] ($\alpha_1 v_1 + \cdots + \alpha_n v_n = c$), and the *linear congruence equality* domain [15] ($\alpha_1 v_1 + \cdots + \alpha_n v_n \equiv a\ [b]$).

Non-relational domains are fast, but suffer from poor precision: they cannot encode relations between variables of a program. Relational domains are much more precise, but also very costly. Consider, for example, the simple program of Figure 1 that simulates many random walks and stores the hits in an array. Our goal is to discover that, at program point (•), x is in the set $\{-5, -3, -1, 1, 3, 5\}$ of allowed indices for hit, so that the instruction hit[x]++ is correct. The invariants found at (•) by several methods are shown in Figure 2. Remark that, even if the desired invariant is a simple combination of an interval and a congruence relation, all non-relational analyses fail to discover it because they cannot infer the relationship between x and i at program point (★). It is often the case that, in order to find a given invariant at the end of a loop, one must be able to express invariants of a more complex form inside the loop. In this example, the desired result can be obtained by using relational analyses, as shown in Figure 2.

2.2 Graph-Based Algorithms

Pratt remarked in [26] that the satisfiability of a set of constraints of the form ($x - y \leq c$) can be efficiently tested in $\mathbb{Z}$, $\mathbb{Q}$, or $\mathbb{R}$ by looking at the simple loops of a directed weighted graph—so-called *potential graph*. Shostak then extended in [27] this graph-based algorithm to the satisfiability of constraints of the form ($\alpha x + \beta y \leq c$), in $\mathbb{Q}$ or $\mathbb{R}$. Harvey and Stuckey proved in [17] that Shostak's algorithm can be used to check satisfiability of constraints of the form ($\pm x \pm y \leq c$) in $\mathbb{Z}$. These approaches focus only on satisfiability and do not address the problem of manipulating constraint sets.

```
hit: array {-5,-3,-1,1,3,5} ↦ int;
for k=1 to 1000 do
  x=0;
  for i=1 to 5 do
    (★) if random() then x++; else x--;
  done;
  (•) hit[x]++;
done
```

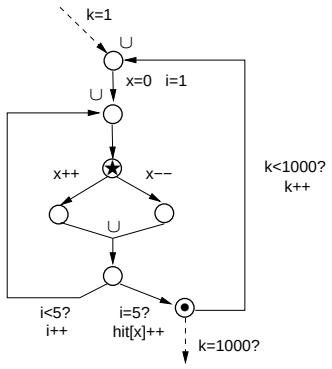

Figure1. A simple random walk program, and its control flow graph.

	Interval	**Congruence**	**Polyhedron**	**Congruence equality**
(★)	$i \in [1, 5]$	—	$i \in [1, 5],\ -x \leq i - 1 \leq x$	$x + i \equiv 1\ [2]$
(•)	—	—	$x \in [-5, 5]$	$x \equiv 1\ [2]$

Figure2. Invariants discovered for the program in Figure 1, at program points (•) and (★), using several non-relational (left) and relational (right) analyses.

Using Pratt's remark, the model-checking community developed a structure called *Difference-Bound Matrix* (DBM, for short) and algorithms based on *shortest-path closure* of weighted graphs to represent and manipulate constraints of the form $(x - y \leq c)$ and $(x \leq c)$. DBMs are used to model-check timed-automata. In [28], Toman and Chomicki introduced *periodicity graphs* that manipulate constraints of the form $(x \equiv y + c\ [k])$, and apply this to constraint logic programming and database query.

Unlike model-checking and constraint programming, we would like to analyze generic programs, and not simply systems closed under restricted constraint forms—such as timed automata, or database query languages. Our methodology is first to choose an invariant form, and then to design a fully-featured abstract domain (including guard and assignment transfer functions, as well as a widening operator) allowing to discover invariants of this form on any program, using maybe coarse over-approximations for those semantics functions that cannot be represented exactly using the chosen invariant form.

In [23], we already presented a DBM-based abstract domain allowing to discover invariants of the form $(x - y \leq c)$ and $(x \leq c)$. In [24], we presented a slight extension, called the *octagon abstract domain*, allowing to discover invariants of the form $(\pm x \pm y \leq c)$.

2.3 Our Contribution

Our goal is to propose a new family of numerical abstract domains, based on shortest-path closure algorithms, that allows to discover invariants of the form $(x-y \in C)$, where C lives in a non-relational domain. This family generalizes the DBM-based abstract domain of [23] and allows us to build new domains, such as the *zone congruence domain* that discovers invariants of the form $(x \equiv y+c\ [k])$. Such relational domains are between, in term of cost and precision, non-relational domains and classical relational domains—such as polyhedron or congruence equality. Thus, we will call these domains *weakly relational domains*.

We claim that such domains are useful as they give, on the example of Figure 1, almost the same result as the relational analyses, for a smaller cost. Do not be fooled by the simplicity of this example program; the abstract interpretation framework allows the design of complex inter-procedural analyzes [1] adapted to real-life programming languages. A numerical abstract domain is just a brick in the design of an analysis; it can be plugged in many existing analyses, such as pointer [10], string cleanness [11], termination analyses [3], analyses of mobility [12], probabilistic programs [25], abstraction of tree-based semantics [21], etc.

The paper is organized as follows. Section 3 reformulates the construction of non-relational numerical abstract domains using the concept of *basis*. Section 4 explains our generic construction of weakly relational domains and applies it in order to retrieve the zone domain and build new abstract domains. Section 5 provides a few applications and ideas for improvement. We conclude in Section 6. Important proofs are postponed to the Annex; reading them may help to understand the definitions chosen in Sections 3-4 (mainly Hypotheses 1).

3 Bases and Non-Relational Numerical Abstract Domains

This sections first recalls the concept of numerical abstract domain. We introduce the new concept of *basis* and show how it can be used to retrieve standard non-relational domains. Introducing such a concept only for this purpose would be formalism for the sake of formalism. However, we will show in the next section how to use this concept to build our weakly relational domains. Hence, bases are the common denominator between classical non-relational domains and our weakly relational domain family.

From the implementation point of view, bases are modules sharing a common signature, and we propose one functor for building non-relational domains from this signature, and one functor for building weakly relational domains. From the mathematical point of view, this approach makes our proofs modular and easier to handle.

3.1 Semantics and Abstract Domains

Let P be a procedure-free, pointer-free program such as the one in Figure 1. Let $\mathcal{V} = \{v_0, \ldots, v_{N-1}\}$ be the set of its numerical variables, with values in the set

$\mathbb{I}$ (that can be $\mathbb{Z}$, $\mathbb{Q}$, or $\mathbb{R}$). We attach to each node of the control flow graph of P a set of *environments* $\mathbf{e}^\sharp \in \mathcal{D}^\sharp \stackrel{\text{def}}{=} \mathcal{P}(\mathbb{I}^N)$ that maps each variable to its value. The information is propagated using the following equations:

- *guards*, corresponding to tests in the initial program, filter the environment: $\mathbf{e}^\sharp_{(\texttt{expr ?})} \stackrel{\text{def}}{=} \{ (x_0, \ldots, x_{N-1}) \in \mathbf{e}^\sharp \mid \texttt{expr}(x_0, \ldots, x_{N-1}) \text{ holds} \}$;
- *assignments* change the value of one variable: $\mathbf{e}^\sharp_{(v_i \leftarrow \texttt{expr})} \stackrel{\text{def}}{=} \{ (x_0, \ldots, \texttt{expr}(x_0, \ldots, x_{N-1}), \ldots) \mid (x_0, \ldots, x_i, \ldots) \in \mathbf{e}^\sharp \}$;
- *union* $\cup$ collects environments at control flow joins.

Because of loop constructs, the control flow graph contains loops and the system of equations described above is recursive. Classical safety semantics consider the *least fixpoint* solution.

This semantics is not decidable in general. One thus constructs an *abstract domain* [6] which is a computer-representable partially ordered set $(\mathcal{D}, \preceq)$ connected to $(\mathcal{D}^\sharp, \subseteq)$ by a monotonic concretization function $\mathit{\Gamma}$. Guard, assignment, and union operators have *sound over-approximations* in $\mathcal{D}$, that is to say:

$$\begin{cases} \mathit{\Gamma}(\mathbf{e}_{(\texttt{expr ?})}) & \supseteq \ (\mathit{\Gamma}(\mathbf{e}))^\sharp_{(\texttt{expr ?})}; \\ \mathit{\Gamma}(\mathbf{e}_{(v_i \leftarrow \texttt{expr})}) & \supseteq \ (\mathit{\Gamma}(\mathbf{e}))^\sharp_{(v_i \leftarrow \texttt{expr})}; \\ \mathit{\Gamma}(\mathbf{e} \sqcup \mathbf{f}) & \supseteq \ \mathit{\Gamma}(\mathbf{e}) \cup \mathit{\Gamma}(\mathbf{f}) \ . \end{cases}$$

Unlike classical data-flow analysis, $\mathcal{D}$ can have an infinite height, so one needs a *widening operator* [6] to compute, in finite-time, an over-approximation of least fixpoints. The widening operator $\triangledown$ should have the following properties [6]:

Definition 1. Widening.

1. $\forall x, y \in \mathcal{C},\ x \preceq x \triangledown y,$ *and* $y \preceq x \triangledown y$.
2. *For every increasing sequence* $(y_n)_{n\in\mathbb{N}}$, *the sequence* $(x_n)_{n\in\mathbb{N}}$ *defined by* $\begin{cases} x_0 = y_0, \\ x_{n+1} = x_n \triangledown y_n, \end{cases}$ *is ultimately stationary. (Ascending chain condition.)*

$\Diamond$

The least fixpoint $\mathrm{lfp}_\perp F$ of an abstract operator F is replaced by the limit of the *stationary* sequence $X_0 = \perp$, $X_{i+1} = X_i \triangledown F(X_i)$—see [2] for more information on when and how to use widenings.

It is a major result of abstract interpretation that, when computing in the abstract domain with widenings, one obtains, in finite time, a sound over-approximation of the initial semantics.

3.2 Bases

We call *basis* a structure that represents and manipulates subsets of $\mathbb{I}$ in a way suitable to build a *non-relational* abstract domain. Such bases will be then used in the following section to build our family of *relational* domains. It is given by:

Definition 2. Basis.

1. *A computer-representable set* $\mathcal{C}$ *with partial order* $\sqsubseteq$ *and least element* $\bot$.
2. *A strict, monotonic, injective concretization* $\gamma : \mathcal{C} \hookrightarrow \mathcal{P}(\mathbb{I})$.
3. *Each element* $C \subseteq \mathbb{I}$ *has an over-approximation* $C^\sharp \in \mathcal{C}$*:* $\gamma(C^\sharp) \supseteq C$.
4. *There exists an over-approximation* $\sqcap$ *for the intersection:*
$$\gamma(C_1^\sharp \sqcap C_2^\sharp) \supseteq \gamma(C_1^\sharp) \cap \gamma(C_2^\sharp) \ .$$
5. *There exists an upper bound* $\sqcup$*:* $\quad C_1^\sharp, C_2^\sharp \sqsubseteq C_1^\sharp \sqcup C_2^\sharp$.
6. *Each* k*-ary arithmetic expression* $\texttt{expr}_k(c_1, \ldots, c_k)$ *has an abstract over-approximated counterpart* $\texttt{expr}_k^\sharp(C_1^\sharp, \ldots, C_k^\sharp)$*:*
$$\gamma(\texttt{expr}_k^\sharp(C_1^\sharp, \ldots, C_k^\sharp)) \supseteq \{\ \texttt{expr}_k(c_1, \ldots, c_k) \mid c_i \in \gamma(C_i^\sharp)\ \} \ .$$
7. *If* $\mathcal{C}$ *has strictly infinite chains, there is a* widening *operator* $\triangledown$ *.* $\qquad \Diamond$

By strictness, $\gamma(\bot) = \emptyset$. Thanks to points 2 and 3, there exists a unique abstract element $\top$ such that $\gamma(\top) = \mathbb{I}$. The least upper bound $\sqcup$ is also an over-approximation for the union: $\gamma(C_1^\sharp \sqcup C_2^\sharp) \supseteq \gamma(C_1^\sharp) \cup \gamma(C_2^\sharp)$.

3.3 A few Classical Bases

We now present a few set of bases that allow us to retrieve the non-relational constant propagation [6], interval [5], and congruence domains [14].

Constant Basis. $\mathcal{C}_{\text{cst}} \stackrel{\text{def}}{=} \{\bot,\ \top\} \cup \{\ c^\sharp \mid c \in \mathbb{I}\ \}$.

All abstract operators are straightforward and not discussed here (see [6] for more details). There is no need for a widening operator.

Interval Basis. $\mathcal{C}_{[a,b]} \stackrel{\text{def}}{=} \{\bot\} \cup \{\ [a,b] \mid a \in \mathbb{I} \cup \{-\infty\},\ b \in \mathbb{I} \cup \{+\infty\},\ a \leq b$[1] $\}$.

Most abstract operators are straightforward (see [5] for more details). We will only recall here the classical widening operator:

$$[a_1, b_1] \triangledown [a_2, b_2] \stackrel{\text{def}}{=} \left[\begin{cases} a_1 & \text{if } a_1 \leq a_2 \\ -\infty & \text{elsewhere} \end{cases} , \quad \begin{cases} b_1 & \text{if } b_1 \geq b_2 \\ +\infty & \text{elsewhere} \end{cases} \right] .$$

In $\mathbb{Q}$ or $\mathbb{R}$, one can alternatively define the *open interval lattice* $\mathcal{C}_{]a,b[}$ the same way. One can even combine these informations to obtain a basis $\mathcal{C}_{\text{I}a,b\text{I}}$ where each bound may or may not be included.

Congruence Basis. $\mathcal{C}^{\mathbb{Z}}_{a\mathbb{Z}+b} \stackrel{\text{def}}{=} \{\bot\} \cup \{\ (a\mathbb{Z}+b) \mid a \in \mathbb{N}^* \cup \{\infty\},\ b \in \mathbb{Z}\ \}$.

A basis is built on $\mathcal{C}^{\mathbb{Z}}_{a\mathbb{Z}+b}$ thanks to the operators described in Figure 4 (using the definitions of Figure 3). However, for the sake of conciseness, Figure 4 does not present abstract k-ary arithmetic expressions, but the binary plus, which is denoted by the infix $\boxplus$ operator (see [14] for more details). There is no strictly increasing infinite chain, so there is no need for a widening operator.

One may also consider to adapt the definitions of Figures 3 and 4 to *rational congruences* [16]: $\mathcal{C}^{\mathbb{Q}}_{a\mathbb{Z}+b} = \{\bot\} \cup \{\ (a\mathbb{Z}+b) \mid a \in \mathbb{Q}^{>0} \cup \{\infty\},\ b \in \mathbb{Q}\ \}$.

[1] Bounds are part of the interval only if finite. Do not be confused by closed interval notations such as $[a, +\infty]$; the interval cannot contain infinite elements.

In the following, $x, x' \in \mathbb{Z}$ and $y, y' \in \mathbb{N}^* \cup \{\infty\}$:

- $y/y' \stackrel{\text{def}}{\iff} y$ is a divisor of y' ($\exists k \in \mathbb{N}^*$ such that $y' = ky$), or $y' = \infty$;
- $x \equiv x'\ [y] \stackrel{\text{def}}{\iff} x \neq x'$ and $y/|x - x'|$, or $x = x'$;
- $\vee$ is the least common multiple, extended by $y \vee \infty \stackrel{\text{def}}{=} \infty \vee y \stackrel{\text{def}}{=} \infty$;
- $\wedge$ is the greatest common divisor, extended by $y \wedge \infty \stackrel{\text{def}}{=} \infty \wedge y \stackrel{\text{def}}{=} y$.

Figure3. Classical arithmetic operators extended to $\mathbb{N}^* \cup \{\infty\}$.

3.4 Building Non-relational Domains from Bases

Building a non-relational domain $(\mathcal{D}, \preceq)$ from a basis $(\mathcal{C}, \sqsubseteq)$ is straightforward:

- We set $\mathcal{D} \stackrel{\text{def}}{=} \mathcal{V} \mapsto \mathcal{C}$.
- The concretization Γ, order $\preceq$, union $\sqcup$, and widening $\triangledown$ are simply point-wise versions of the corresponding operators on the basis.
- Assignments are defined using the abstract counterpart of expressions:
  $$(C_1^\sharp, \ldots, C_i^\sharp, \ldots)_{(v_i \leftarrow \mathit{expr})} \stackrel{\text{def}}{=} (C_1^\sharp, \ldots, \mathit{expr}_N^\sharp(C_1^\sharp, \ldots, C_{N-1}^\sharp), \ldots) \ .$$
- Only non-relational guards ($v_i \in C$?) do some filter job:
  $$(C_1^\sharp, \ldots, C_i^\sharp, \ldots)_{(v_i \in C\ ?)} \stackrel{\text{def}}{=} (C_1^\sharp, \ldots, C_i^\sharp \sqcap C^\sharp, \ldots) \text{ where } \gamma(C^\sharp) \supseteq C \ .$$
  In other guard cases, it is safe to use the identity function.

From an implementation point of view, the non-relational domain is simply a generic functor module, and each basis implementation is a module.

4 Building Weakly Relational Domains from Bases

Now we would like to represent relations of the form $v_j - v_i \in \gamma(C)$ where C lives in a basis $\mathcal{C}$ (instead of $v_i \in \gamma(C)$). A plain basis is not sufficient, we will need a way—a so-called *closure*—to propagate relational information. The main result of this paper can be schemed as follows:

basis (with extra hypotheses)	+ closure	$\Longrightarrow$	weakly relational domain

4.1 Hypotheses on the Basis

Not all bases $\mathcal{C}$ are acceptable. We need the following extra hypotheses:

Hypotheses 1. Acceptable Bases.

1. *There exists* exact *abstract counterparts for the intersection* $\sqcap$ *(which should also be a* lower bound *for* $\sqsubseteq$*), unary minus* $\boxminus$*, and binary plus* $\boxplus$ *operators:*
   - $\gamma(x \boxplus y) = \{\ a + b \mid a \in \gamma(x),\ b \in \gamma(y)\ \}$; *(Abstract plus.)*
   - $\gamma(\boxminus x) = \{\ -a \mid a \in \gamma(x)\ \}$; *(Abstract opposite.)*
   - $x \sqcap y \sqsubseteq x, y$, *so* $\gamma(x \sqcap y) = \gamma(x) \cap \gamma(y)$. *(Abstract intersection.)*

- *Concretization:*
$$\gamma(C) \stackrel{\text{def}}{=} \begin{cases} \{\, ak+b \mid k \in \mathbb{Z} \,\} & \text{if } C = (a\mathbb{Z}+b),\ a \neq \infty; \\ \{\, b \,\} & \text{if } C = (\infty\mathbb{Z}+b); \\ \emptyset & \text{if } C = \bot. \end{cases}$$
- *Order:*
  - $(a\mathbb{Z}+b) \sqsubseteq (a'\mathbb{Z}+b') \stackrel{\text{def}}{\iff} a'/a$ and $b \equiv b'\ [a']$.
  - $\bot \sqsubseteq C,\ \forall C \in \mathcal{C}$.
- *Intersection (exact abstract counterpart for the intersection $\cap$):*
  - $(a\mathbb{Z}+b) \sqcap (a'\mathbb{Z}+b') \stackrel{\text{def}}{=} \begin{cases} (a \vee a')\,\mathbb{Z}+b'' & \text{if } b \equiv b'\ [a \wedge a'], \\ \bot & \text{elsewhere,} \end{cases}$
    where b'' is such that $b'' \equiv b\ [a \vee a'] \equiv b'\ [a \vee a']$ (Bezout Theorem).
  - $\bot \sqcap C \stackrel{\text{def}}{=} C \sqcap \bot \stackrel{\text{def}}{=} \bot,\ \forall C \in \mathcal{C}$.
- *Least Upper Bound:*
  - $(a\mathbb{Z}+b) \sqcup (a'\mathbb{Z}+b') \stackrel{\text{def}}{=} (a \wedge a' \wedge |b-b'|)\,\mathbb{Z} + \min(b,b')$.
  - $\bot \sqcup C \stackrel{\text{def}}{=} C \sqcup \bot \stackrel{\text{def}}{=} C,\ \forall C \in \mathcal{C}$.
- *Sum (exact abstract counterpart for the binary + operator):*
  - $(a\mathbb{Z}+b) \boxplus (a'\mathbb{Z}+b') \stackrel{\text{def}}{=} (a \wedge a')\,\mathbb{Z} + (b+b')$.
  - $\bot \boxplus C \stackrel{\text{def}}{=} C \boxplus \bot \stackrel{\text{def}}{=} \bot,\ \forall C \in \mathcal{C}$.

Figure4. Concretization and abstract operators in $\mathcal{C}^{\mathbb{Z}}_{a\mathbb{Z}+b}$.

2. *Each singleton $\{c\}$, $c \in \mathbb{I}$ must be* exactly *represented by an abstract element $c^\sharp \in \mathcal{C}$: $\gamma(c^\sharp) = \{c\}$.*
3. *For each finite family $(x_i)_{i\in I}$,*
$$\bigsqcap_{i\in I} x_i = \bot \Longrightarrow \exists i,j \in I,\ x_i \sqcap x_j = \bot\ .$$
4. *$\sqcap$ distributes $\boxplus$: for each family $(x_i)_{i\in I}$ and element x of $\mathcal{C}$,*
$$\text{if } \bigsqcap_{i\in I} x_i \neq \bot,\ \text{then } \bigsqcap_{i\in I} (x \boxplus x_i) = x \boxplus \left(\bigsqcap_{i\in I} x_i \right)\ .$$
5. *$\boxminus$ distributes $\boxplus$ and $\sqcap$. $\boxplus$ and $\sqcap$ are commutative and associative.* ✠

These hypotheses were stated in order to prove our main theorem, which is the correctness of the closure operator. Thus, one may have to wait until Theorem 6— and its proof postponed in Annex A—Remark that Hypotheses 1.3-4 are very strong. The full basis $\mathcal{C} = \mathcal{P}(\mathbb{I})$, for instance, does not respect them.

Remark. There exists resemblance between bases respecting Hypotheses 1 and the graph-theory classical notion of *complete dïoid* [13]—an extension of *exotic*

algebras. A complete dïoid is a complete semi-lattice with an addition (our $\sqcap$), and a multiplication (our $\boxplus$) that distributes over the addition. However, full distributivity in dïoids implies that $\bot \boxplus \top = \top$ where we would have preferred $\bot \boxplus \top = \bot$. Thus, in our framework, distributivity is restricted (Hypothesis 1.4).

4.2 Representing Relations

A set of constraints of the form $v_j - v_i \in \gamma(C),\ C \in \mathcal{C}$ is now represented by a *coherent constraint matrix*:

Definition 3. Constraint Matrices.

1. *A* constraint matrix $\mathbf{m}$ *is a* $N \times N$ *matrix with elements in* $\mathcal{C}$*; the element* $\mathbf{m}_{ij}$ *represents the constraint* $v_j - v_i \in \gamma(\mathbf{m}_{ij})$.
2. *We suppose, as an implicit constraint, that* $v_0 = 0$*, so that unary constraints* $v_i \in \gamma(\mathbf{m}_{0i})$ *can be represented as* $v_i - v_0 \in \gamma(\mathbf{m}_{0i})$.
3. $\mathbf{m}$ *is* coherent *if* $\forall i, j,\ \mathbf{m}_{ij} = \boxminus \mathbf{m}_{ji}$ *and* $\forall i,\ \gamma(\mathbf{m}_{ii}) = \{0\}$.
4. $\mathbf{m}$ *represents the set (so-called* concretization *of* $\mathbf{m}$*):*
   $\mathit{\Gamma}(\mathbf{m}) \stackrel{\text{def}}{=} \{\ (x_0, \ldots, x_{N-1}) \in \mathbb{I}^N \mid x_0 = 0,\ \forall i, j,\ x_j - x_i \in \gamma(\mathbf{m}_{ij})\ \}\ .$ $\qquad \Diamond$

Our abstract domain is the set $\mathcal{D}$ of coherent constraint matrices, ordered by the point-wise extension $\preceq$ of the partial order $\sqsubseteq$ on $\mathcal{C}$:

$$\begin{aligned} \mathbf{m} \preceq \mathbf{n} &\stackrel{\text{def}}{\iff} \forall i, j,\ \mathbf{m}_{ij} \sqsubseteq \mathbf{n}_{ij}; \\ \mathbf{m} \doteq \mathbf{n} &\stackrel{\text{def}}{\iff} \forall i, j,\ \mathbf{m}_{ij} = \mathbf{n}_{ij}; \\ \dot{\bot} &\stackrel{\text{def}}{=} \inf_{\preceq} \mathcal{D} \text{ is such that } \forall i, j,\ \dot{\bot}_{ij} = \bot\ . \end{aligned}$$

The concretization function on $\mathcal{D}$ is $\mathit{\Gamma}$, and we have:

Theorem 4. Monotony of $\mathit{\Gamma}$.

1. $\mathbf{m} \preceq \mathbf{n} \Longrightarrow \mathit{\Gamma}(\mathbf{m}) \subseteq \mathit{\Gamma}(\mathbf{n})$.
2. $\mathbf{m} \doteq \mathbf{n} \Longrightarrow \mathit{\Gamma}(\mathbf{m}) = \mathit{\Gamma}(\mathbf{n})$. $\qquad \square$

However, this is *not an equivalence* and we can have two different constraint matrices $\mathbf{m} \dot{\neq} \mathbf{n}$ with the same concretization $\mathit{\Gamma}(\mathbf{m}) = \mathit{\Gamma}(\mathbf{n})$.

4.3 General Closure Operator

Implicit Constraints. Because our abstract domain is relational, the constraints between variables are not independent. One can deduce a constraint on $x - z$ by adding a constraint on $x - y$ to a constraint on $y - z$. Such deduced constraints are called *implicit constraints* because they may not be present explicitly in $\mathbf{m}$. More generally, given any path $\langle i = i_1, \ldots, i_n = j \rangle$ in $\mathbf{m}$, we can construct the following implicit constraint:

$$v_j - v_i \in \gamma\left(\boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}\right)\ .$$

Shortest-Path Closure. A nice property of DBMs [19] and periodicity graphs [28] that will hold for our constraint matrices is that the concretization is entirely determined by the set of implicit constraints of the above form. DBMs use any *shortest-path closure* algorithm in order to make all implicit constraints explicit. Here, we adapt the *Floyd-Warshall algorithm* [4, §25.3], to our matrices.

Definition 5. Closure. *Let* $\mathbf{m}$ *be a coherent matrix. Its* closure *is the result* $\mathbf{m}^\star$ *of the following* modified Floyd-Warshall algorithm*:*

$$\begin{cases} \mathbf{m}^0 & \stackrel{\text{def}}{=} \mathbf{m}; \\ \mathbf{m}_{ij}^{k+1} & \stackrel{\text{def}}{=} \mathbf{m}_{ij}^k \sqcap (\mathbf{m}_{ik}^k \boxplus \mathbf{m}_{kj}^k); \\ \mathbf{m}^\star & \stackrel{\text{def}}{=} \mathbf{m}^N \ . \end{cases}$$

$\Diamond$

The Floyd-Warshall algorithm was chosen because it is easy to understand, straightforward to implement, and easy to adapt to constraint matrices. It performs $\mathcal{O}(N^3)$ elementary basis operations.

Here is the main theorem of this paper. The following results will be used extensively in Section 4.4 in order to design abstract operators. The proof of this theorem relies heavily on Hypotheses 1—in fact, the proof itself motivated the hypotheses.

Theorem 6. Closure.

1. $\Gamma(\mathbf{m}^\star) = \Gamma(\mathbf{m})$.
2. $\Gamma(\mathbf{m}) = \emptyset \iff \exists i,\ \mathbf{m}_{ii}^\star = \bot$.
3. *If* $\Gamma(\mathbf{m}) \neq \emptyset$*, then* $\mathbf{m}^\star$ *enjoys the following properties:*
   - $\mathbf{m}^\star$ *is a coherent matrix;* *(Coherence.)*
   - $\forall i, j,\ \mathbf{m}_{ij}^\star = \bigsqcap_{\langle i=i_1,\ldots,i_n=j\rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}$*;* *(Transitive closure.)*
   - $\forall i, j,\ \forall c \in \gamma(\mathbf{m}_{ij}^\star),\ \exists (x_0, \ldots, x_{N-1}) \in \Gamma(\mathbf{m}),\ x_j - x_i = c$*;* *(Saturation.)*
   - $\mathbf{m}^\star = \inf_{\preceq}\{\ \mathbf{n} \mid \Gamma(\mathbf{m}) = \Gamma(\mathbf{n})\ \}$*;* *(Normal form.)*
   - $\mathbf{m}^{\star\star} = \mathbf{m}^\star$. *(Closure.)*

$\square$

Incremental Closure. When modifying slightly a closed matrix, we do not need to perform the modified Floyd-Warshall algorithm completely to get the closure of the new matrix. If the upper-left $M \times M$ sub-matrix of $\mathbf{m}$ is already closed, we can use the following $\mathcal{O}((N-M)\cdot N^2)$ algorithm:

$$\begin{cases} \mathbf{m}^0 & \stackrel{\text{def}}{=} \mathbf{m}; & \\ \mathbf{m}_{ij}^{k+1} & \stackrel{\text{def}}{=} \mathbf{m}_{ij}^k & \text{if } i,j,k < M; \\ \mathbf{m}_{ij}^{k+1} & \stackrel{\text{def}}{=} \mathbf{m}_{ij}^k \sqcap (\mathbf{m}_{ik}^k \boxplus \mathbf{m}_{kj}^k) & \text{elsewhere;} \\ \mathbf{m}^\star & \stackrel{\text{def}}{=} \mathbf{m}^N \ . & \end{cases}$$

We can adapt easily the algorithm—permuting variables—to get a general incremental closure algorithm performing $\mathcal{O}(N^2 \cdot c)$ elementary basis operations, where c is the number of lines and columns that have changed since the last closure.

4.4 Generic Operators

Emptiness Testing. Testing the satisfiability of a constraint matrix is done using Theorem 6.2. Unlike the constraint programming approach, we do not use a specific loop-based satisfiability algorithm, but let our generic closure algorithm solve both the satisfiability and the normal form problems at once.

Equality, Inclusion Testing. The normal form property of Theorem 6.3 allows us to easily test equality and inclusion of non-empty concretizations:

Theorem 7. Equality and Inclusion Testing.
1. $\mathit{\Gamma}(\mathbf{m}) = \mathit{\Gamma}(\mathbf{n}) \iff \mathbf{m}^\bigstar \doteq \mathbf{n}^\bigstar$.
2. $\mathit{\Gamma}(\mathbf{m}) \subseteq \mathit{\Gamma}(\mathbf{n}) \iff \mathbf{m}^\bigstar \preceq \mathbf{n}$. □

Remark that we do not need to close the right argument while testing inclusion.

Union, Intersection. $\gamma(\mathcal{C})$ is stable under intersection, so we simply extend point-wisely $\sqcap$ to represent the intersection of two concretizations:

$$[\mathbf{m} \dot{\sqcap} \mathbf{n}]_{ij} \stackrel{\text{def}}{=} \mathbf{m}_{ij} \sqcap \mathbf{n}_{ij} \ .$$

Theorem 8. Intersection. $\mathit{\Gamma}(\mathbf{m} \dot{\sqcap} \mathbf{n}) = \mathit{\Gamma}(\mathbf{m}) \cap \mathit{\Gamma}(\mathbf{n})$. □

$\gamma(\mathcal{C})$ is not generally closed under union, neither is $\mathit{\Gamma}(\mathcal{D})$. However, if there exists an upper bound $\sqcup$ in $\mathcal{C}$, we can extend it point-wisely in $\mathcal{D}$:

$$[\mathbf{m} \dot{\sqcup} \mathbf{n}]_{ij} \stackrel{\text{def}}{=} \mathbf{m}_{ij} \sqcup \mathbf{n}_{ij} \ .$$

If $\sqcup$ is a *least* upper bound, $\dot{\sqcup}$ can be used to determine the least upper bound of two concretizations, *provided the arguments are closed matrices.*

Theorem 9. (Least) Upper Bound.
1. *If* $\forall a, b \in \mathcal{C},\ \gamma(a \sqcup b) \supseteq \gamma(a) \cup \gamma(b)$, *then* $\mathit{\Gamma}(\mathbf{m} \dot{\sqcup} \mathbf{n}) \supseteq \mathit{\Gamma}(\mathbf{m}) \cup \mathit{\Gamma}(\mathbf{n})$. *(Upper bound.)*
2. *If* $\gamma(a \sqcup b) = \inf_{\subseteq}\{\ \gamma(c) \mid \gamma(c) \supseteq \gamma(a) \cup \gamma(b)\ \}$, *then* $\mathit{\Gamma}(\mathbf{m}^\bigstar \dot{\sqcup} \mathbf{n}^\bigstar) = \inf_{\subseteq}\{\ \mathit{\Gamma}(\mathbf{o}) \mid \mathit{\Gamma}(\mathbf{o}) \supseteq \mathit{\Gamma}(\mathbf{m}) \cup \mathit{\Gamma}(\mathbf{n})\ \}$. *(Least upper bound.)*
3. $(\mathbf{m}^\bigstar \dot{\sqcup} \mathbf{n}^\bigstar)^\bigstar = \mathbf{m}^\bigstar \dot{\sqcup} \mathbf{n}^\bigstar$. *($\dot{\sqcup}$ respects closure.)*

□

Widening. $\mathcal{D}$ has infinite strictly increasing chains only if $\mathcal{C}$ has. A widening $\dot{\triangledown}$ is obtained on $\mathcal{D}$ by point-wise application of the widening $\triangledown$ on $\mathcal{C}$:

$$[\mathbf{m} \dot{\triangledown} \mathbf{n}]_{ij} \stackrel{\text{def}}{=} \mathbf{m}_{ij} \triangledown \mathbf{n}_{ij} \ .$$

$\dot{\triangledown}$ respects Definition 1. Thus, the least fixpoint of an operator F can be over-approximated by the limit of the stationary sequence $X_{i+1} = X_i \dot{\triangledown} F(X_i)$. One could expect, as for the least upper bound, to get a better precision by closing the arguments of $\dot{\triangledown}$, but this is not the case. Even worse, enforcing the closure of the chain by computing $X_{i+1} = (X_i \dot{\triangledown} F(X_i))^\bigstar$ *breaks the ascending chain condition* and prevents the analysis from terminating in some cases. We advocate here the use of the following iteration: $X_{i+1} = X_i \dot{\triangledown} F(X_i^\bigstar)$.

Guard. We can easily implement tests of the form $(v_j - v_i \in C\ ?)$:

$$\left[\mathbf{m}_{(v_j - v_i \in C\ ?)}\right]_{kl} \stackrel{\text{def}}{=} \begin{cases} \mathbf{m}_{kl} \sqcap C^\sharp & \text{if } (k,l) = (i,j); \\ \mathbf{m}_{kl} \sqcap (\boxminus C^\sharp) & \text{if } (k,l) = (j,i); \\ \mathbf{m}_{kl} & \text{elsewhere;} \end{cases}$$

choosing $C^\sharp$ such that $\gamma(C^\sharp) \supseteq C$.

Tests of the form $(v_j \in C\ ?)$ are implemented by choosing $i = 0$.
For other tests, it is safe to do nothing:

$$\mathbf{m}_{(?)} \stackrel{\text{def}}{=} \mathbf{m} \ .$$

Projection. In order to find the set of values that a variable can take, we use the following theorem derived from the saturation property of the closure:

Theorem 10. $\{\ x \mid \exists (x_0, \ldots, x_{N-1}) \in \Gamma(\mathbf{m})\ \textit{with}\ x_i = x\ \} = \gamma(\mathbf{m}^\bigstar_{0i})$. □

Forget. Forgetting the value of a variable is useful to implement the random assignment $(v_i \leftarrow ?)$, which also serves as a coarse approximation for complex assignments. Before forgetting all information on a variable, one should close the argument matrix so that we do not loose implicit constraints:

$$\left[\mathbf{m}_{(v_i \leftarrow ?)}\right]_{kl} \stackrel{\text{def}}{=} \begin{cases} \top & \text{if } k = i \text{ or } l = i; \\ \mathbf{m}^\bigstar_{kl} & \text{elsewhere .} \end{cases}$$

Theorem 11.
$\Gamma(\mathbf{m}_{(v_i \leftarrow ?)}) = \{\ (x_0, \ldots, x_i, \ldots) \mid \exists x,\ (x_0, \ldots, x, \ldots) \in \Gamma(\mathbf{m})\ \}$. □

Assignment. For assignments of the form $(v_i \leftarrow v_j + c)$, one can find an *exact* abstract counterpart:

$$\left[\mathbf{m}_{(v_i \leftarrow v_i + c)}\right]_{kl} \stackrel{\text{def}}{=} \begin{cases} \mathbf{m}_{kl} \boxplus \{c\} & \text{if } k = i \text{ and } l \neq i; \\ \mathbf{m}_{kl} \boxplus \{-c\} & \text{if } l = i \text{ and } k \neq i; \\ \mathbf{m}_{kl} & \text{elsewhere;} \end{cases}$$

$$\mathbf{m}_{(v_i \leftarrow v_j + c)} \stackrel{\text{def}}{=} (\mathbf{m}_{(v_i \leftarrow ?)})_{(v_i - v_j \in \{c\}\ ?)} \qquad \text{when } i \neq j \ .$$

For generic assignments $(v_i \leftarrow \mathtt{expr}(v_1, \ldots, v_{N-1}))$, one can always fall back to imprecise non-relational analysis, first projecting the variables, then using the abstraction $\mathtt{expr}^\sharp$ of $\mathtt{expr}$ in our basis:

$$\mathbf{m}_{(v_i \leftarrow \mathtt{expr}(v_1, \ldots, v_{N-1}))} \stackrel{\text{def}}{=} (\mathbf{m}_{(v_i \leftarrow ?)})_{(v_i \in \gamma(C^\sharp)\ ?)}$$
$$\text{where } C^\sharp \stackrel{\text{def}}{=} \mathtt{expr}^\sharp(\mathbf{m}^\bigstar_{01}, \ldots, \mathbf{m}^\bigstar_{0(N-1)}) \ .$$

Trying to be the most precise in all cases may lead to complex algorithms. It seems only worth trying to be a little more precise in some widespread cases, such as $(v_i \leftarrow v_j + v_k)$, for instance:

$$\mathbf{m}_{(v_i \leftarrow v_j + v_k)} \stackrel{\text{def}}{=} (\mathbf{m}_{(v_i \leftarrow\ ?)})_{(v_i \in \gamma(\mathbf{m}^\bigstar_{0j} \boxplus \mathbf{m}^\bigstar_{0k})\ ?)\ (v_i - v_j \in \gamma(\mathbf{m}^\bigstar_{0k})\ ?)\ (v_i - v_k \in \gamma(\mathbf{m}^\bigstar_{0j})\ ?)} \ .$$

Interaction with the Closure. Some of the above operators require the matrix argument(s) to be closed. Some do respect closure—the result is closed if the argument(s) is(are)—and some do not (intersection, guard, assignment, etc.). We thus advocate the use of a *lazy* method that remembers when a matrix is in closed form, and recomputes the closure only when needed. When only a few lines and columns of the matrix are changed (guard, assignment, etc.), we can use the incremental closure. It is useless when all coefficients are changed at once (intersection, widening).

4.5 Some Constructed Domains

We are now ready to apply our construction to the bases presented in Section 3.3, thanks to the following theorem:

Theorem 12. *$\mathcal{C}_{\text{cst}}$, $\mathcal{C}_{[a,b]}$, $\mathcal{C}_{\mathrm{I}a,b\mathrm{I}}$, $\mathcal{C}^{\mathbb{Z}}_{a\mathbb{Z}+b}$, and $\mathcal{C}^{\mathbb{Q}}_{a\mathbb{Z}+b}$ respect Hypotheses 1.* □

Translated Equality Domain. The simplest domain is obtained from the constant basis $\mathcal{C}_{\text{cst}}$ and represents constraints of the form $(v_i = v_j + c)$. This domain is not of great practical interest: its expressive power is low as it is a particular case of the following two domains. It is possible that more efficient solutions exist, as we are not very far from simple equality constraints $v_i = v_j$ for which very efficient algorithms are known (such as, the *Union-Find* algorithm [4, §22]).

Zone Domain. In order to represent invariants of the form $(v_i - v_j \leq c)$, one can think of the basis of initial segments $\{\,]-\infty, a] \mid a \in \mathbb{I} \cup \{+\infty\}\}$, but initial segments are not closed under the $\boxminus$ operation (Hypothesis 1.1). Completing this basis, one naturally find the interval basis $\mathcal{C}_{[a,b]}$.

Compared to classical DBMs [23], the domain obtained is a little redundant (each constraint is represented twice), but has exactly the same expressiveness and complexity. It has the advantage of being implemented over any existing interval library, greatly reducing the need for programming. One can also enhance the zone domain in $\mathbb{Q}$ and $\mathbb{R}$ using the $\mathcal{C}_{\mathrm{I}a,b\mathrm{I}}$ basis that manipulates both strict and non strict constraints.

Zone-Congruence Domain. Using the integer congruence basis $\mathcal{C}^{\mathbb{Z}}_{a\mathbb{Z}+b}$, one builds a domain that discovers constraints of the form $(v_i - v_j \equiv a\ [b])$. This construction looks like *periodicity graphs* [28], but we treat here the case of least upper bound and general purpose transfer functions in detail, whereas [28] is only interested in satisfiability, normal form and conjunction. Moreover, we feel that [28] misses the correct proof of the normal form theorem (our Theorem 6) and does not understand that it relies on some strong properties of congruence sets (Hypotheses 1). Our framework can also extend this domain to a domain of *rational congruences*: $(v_i - v_j \equiv a\ [b])$ with $a, b \in \mathbb{Q}$.

Product Domain. *Reduced product* is a well-known technique [6] for improving the precision of an analysis by combining the power of two abstract domains. It often gives better results than two separate analyses, because it conveys information from one domain to the other during the analysis via a so-called *reduction procedure*, which is a couple of binary operators $(\circlearrowleft, \circlearrowright)$ such that:

$$\begin{array}{l} \circlearrowleft\colon \mathcal{D}_1 \times \mathcal{D}_2 \mapsto \mathcal{D}_1; \\ \circlearrowright\colon \mathcal{D}_1 \times \mathcal{D}_2 \mapsto \mathcal{D}_2; \end{array} \quad \begin{cases} C_1 \circlearrowleft C_2 \preceq_1 C_1; \\ C_1 \circlearrowright C_2 \preceq_2 C_2; \\ \Gamma_1(C_1 \circlearrowleft C_2) \cap \Gamma_2(C_1 \circlearrowright C_2) = \Gamma_1(C_1) \cap \Gamma_2(C_2) \ . \end{cases}$$

In our case, the reduction can be defined on bases—as long as Hypotheses 1 are not broken—with the exact same precision benefit. Moreover, reductions are easier to design on non-relational bases. For example, if we use the following reduction between $\mathcal{C}_{[a,b]}$ and $\mathcal{C}^{\mathbb{Z}}_{a\mathbb{Z}+b}$, we obtain a basis allowing the construction of a domain for constraints of the form $(v_i - v_j \in a \cdot [b,c] + d)$:

$$\begin{cases} [a,b] \circlearrowleft (c\mathbb{Z}+d) \stackrel{\text{def}}{=} [\ \min\{x \in (c\mathbb{Z}+d) \mid x \geq a\}, \\ \qquad\qquad\qquad\quad \max\{x \in (c\mathbb{Z}+d) \mid x \leq b\}\ ]; \\ [a,b] \circlearrowright (c\mathbb{Z}+d) \stackrel{\text{def}}{=} (c\mathbb{Z}+d) \ . \end{cases}$$

Failure. So far, all seems to work well. However, one can find some bases used in very common abstract domains that do not respect Hypotheses 1. For example, the *sign* basis [6] $\mathcal{C}_{\pm} = \{\ \bot,\ ]-\infty, 0],\ [0,0],\ [0, +\infty[,\ ]-\infty, +\infty[\ \}$ and the open interval basis $\mathcal{C}_{]a,b[}$ do not respect Hypothesis 1.2. The *interval congruence* basis $\mathcal{C}_{a\mathbb{Z}+[b,c]}$ (introduced in [20]) does not respect Hypothesis 1.1 (it is not stable under intersection). We do not know if it is possible to build weakly relational domains from such bases.

Modularity. As for non-relational domains, the weakly relational domain family is simply a generic module functor, taking the very same bases implementation modules as parameter.

5 Applications and Future Work

Applications. So far, this framework has only been implemented as an OCaml prototype and tried on a few toy examples. At program point $(\bullet)$ of the program in Figure 1, the reduced-product of the zone and zone-congruence domains found the invariant $(x \leq 5,\ x \equiv 1\ [2])$, which is almost as good as polyhedron and congruence equality analyses combined (Figure 2). It failed to discover that $(x \geq -5)$; however, the octagon abstract domain of [24] that also uses graph-based algorithms can do it.

If in the program of Figure 1 the constant 5 is replaced by a variable m the value of which is not known at analysis time, the analyzer still finds the precise symbolic invariant $(x \leq m,\ x \equiv m\ [2])$.

Scalability. It is still unknown whether graph-based abstract domains scale up. Because of the quadratic memory cost, it cannot handle all the variables of a large program at once; one has to split this set into *packets* in which relational information might be important. These packets do not need to be disjoint, and one can use pivot variables to transfer information between packets. We are currently investigating on such methods.

Because our domain family is relational, it is also adapted to symbolic and modular analyses. One can cut down the cost of an analysis and make it incremental by analyzing separately small pieces of a program [8].

Theoretical Extensions. We tried, in this article, to unite some graph-based numerical satisfiability algorithm and extend them up to an abstract domain, in a united framework. However, a few graph-based algorithms are not handled here: the octagon abstract domain [24] ($\pm x \pm y \leq c$ constraints) and Shostak's satisfiability algorithms [27] ($\alpha x + \beta y \leq c$). It would be interesting to unite all those in a general framework and derive a numerical abstract domain for constraints of the form ($\alpha x + \beta y \leq c$).

6 Conclusion

In this paper, we have proposed the systematic construction of a family of relational domains that represent and manipulate constraints of the form ($x - y \in C$). This construction can be seen as a *functor* lifting non-relational domains to relational ones. The memory cost of an abstract state is *quadratic*, and each transfer function application performs, at worse, a *cubic* number of operations in the non-relational domain. The crux of the method is the adaptation of the shortest-path closure algorithm to a normal form, allowing the derivation of most abstract operators and transfer functions.

In this framework, we have successfully retrieved the existing DBM domain, and constructed new ones. It is the author's opinion that these domains fill a precision and complexity gap between former non-relational and relational domains, and can be used to design medium cost, yet precise, analyses.

Acknowledgments. We would like to thank P. Cousot, J. Feret, X. Rival, and C. Hymans, as well as the anonymous referees, for their useful comments.

References

[1] F. Bourdoncle. Interprocedural abstract interpretation of block structured languages with nested procedures, aliasing and recursivity. In Springer-Verlag, editor, *PLILP'90*, volume 456 of *LNCS*, pages 307–323, 1990.

[2] F. Bourdoncle. Efficient chaotic iteration strategies with widenings. In *FMPA'93*, number 735 in LNCS, 1993.

[3] M. Codish and C. Taboch. A semantic basis for termination analysis of logic programs and its realization using symbolic norm constraints. In *ALP'98*, volume 1298 of *LNCS*, pages 31–45. Springer-Verlag, September 1997.
[4] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.
[5] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *ISOP'76*, pages 106–130. Dunod, Paris, France, 1976.
[6] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *ACM POPL'77*, pages 238–252. ACM Press, 1977.
[7] P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, August 1992.
[8] P. Cousot and R. Cousot. Modular static program analysis, invited paper. In *CC'02*, number 2304 in LNCS, pages 159–178, 2002.
[9] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *ACM POPL'78*, pages 84–97. ACM Press, 1978.
[10] A. Deutsch. Interprocedural may-alias analysis for pointers: Beyond k-limiting. In *ACM PLDI'94*, pages 230–241. ACM Press, 1994.
[11] N. Dor, M. Rodeh, and M. Sagiv. Cleanness checking of string manipulations in C programs via integer analysis. In *SAS'01*, number 2126 in LNCS, July 2001.
[12] J. Feret. Occurrence counting analysis for the π-calculus. In *GETCO'00*, volume 39.2 of *BRICS NS-00-3*, 2001.
[13] M. Gondran and M. Minoux. *Graphs and Algorithms*. Wiley, 1984.
[14] P. Granger. Static analysis of arithmetical congruences. In *International Journal of Computer Mathematics*, volume 30, pages 165–190, 1989.
[15] P. Granger. Static analysis of linear congruence equalities among variables of a program. In *TAPSOFT'91*, number 493 in LNCS, pages 169–192, 1991.
[16] P. Granger. Static analyses of congruence properties on rational numbers. In *SAS'97*, volume 1302 of *LNCS*, pages 278–292, 1997.
[17] W. Harvey and P. Stuckey. A unit two variable per inequality integer constraint solver for constraint logic programming. In *ACSC'97*, volume 19, pages 102–111, February 1997.
[18] M. Karr. Affine relationships among variables of a program. *Acta Informatica*, pages 133–151, 1976.
[19] K. Larsen, F. Larsson, P. Pettersson, and W. Yi. Efficient verification of real-time systems: Compact data structure and state-space reduction. In *IEEE RTSS'97*, pages 14–24. IEEE CS Press, December 1997.
[20] F. Masdupuy. Semantic analysis of interval congruences. In *FMPTA'93*, volume 735 of *LNCS*, pages 142–155, 1993.
[21] L. Mauborgne. *Representation of Sets of Trees for Abstract Interpretation*. PhD thesis, École Polytechnique, Palaiseau, France, November 1999.
[22] A. Miné. Representation of two-variable difference or sum constraint set and application to automatic program analysis. Master's thesis, ENS-DI, Paris, France, 2000.
[23] A. Miné. A new numerical abstract domain based on difference-bound matrices. In *PADO II*, volume 2053 of *LNCS*, pages 155–172. Springer-Verlag, May 2001.
[24] A. Miné. The octagon abstract domain. In *AST 2001 in WCRE 2001*, IEEE, pages 310–319. IEEE CS Press, October 2001.
[25] D. Monniaux. An abstract Monte-Carlo method for the analysis of probabilistic programs. In *POPL'01*, number 1824 in ACM, pages 93–101, 2001.

[26] V. Pratt. Two easy theories whose combination is hard. Technical report, Massachusetts Institute of Technology, Cambridge, September 1977.
[27] R. Shostak. Deciding linear inequalities by computing loop residues. *Journal of the ACM*, 28(4):769–779, October 1981.
[28] D. Toman and J. Chomicki. Datalog with integer periodicity constraints. In *Journal of Logic Programming*, pages 189–203. The MIT Press, 1994.

Appendix

A Proof of the Main Theorem

We present here the complete proof of the main theorem, Theorem 6. It is the proof of this theorem that motivated the choice of Hypotheses 1.

Remark that the proof of this theorem is much simpler in the special case of the interval basis $\mathcal{C}_{[a,b]}$ (see Theorem 2 in the author's master thesis [22]).

Remark also that part of this theorem for the congruence case $\mathcal{C}^{\mathbb{Z}}_{a\mathbb{Z}+b}$ is discussed by Toman and Chomicki in [28], but the proof is somewhat eschewed (Lemma 2.12). Our proof relies heavily on the fact that $\mathcal{C}^{\mathbb{Z}}_{a\mathbb{Z}+b}$ verifies Hypothesis 1.3, which is not trivial.

Proof of Theorem 6.

- **Claim:** $\Gamma(\mathbf{m}^\bigstar) = \Gamma(\mathbf{m})$. □

  We have $\forall k,i,j,\ \mathbf{m}^{k+1}_{ij} = \mathbf{m}^k_{ij} \sqcap (\mathbf{m}^k_{ik} \boxplus \mathbf{m}^k_{kj}) \sqsubseteq \mathbf{m}^k_{ij}$ *(Hypothesis 1.1)*, so $\forall k,\ \Gamma(\mathbf{m}^{k+1}) \subseteq \Gamma(\mathbf{m}^k)$. Conversely, $\forall i,j,k,\ (x_0,\ldots,x_{N-1}) \in \Gamma(\mathbf{m}^k)$, we have $x_k - x_i \in \gamma(\mathbf{m}^k_{ik})$, and $x_j - x_k \in \gamma(\mathbf{m}^k_{kj})$. By summation, $x_j - x_i \in \gamma(\mathbf{m}^k_{ik} \boxplus \mathbf{m}^k_{kj})$ *(Hypothesis 1.1)*. Thus $x_j - x_i \in \gamma(\mathbf{m}^{k+1}_{ij})$, and $\forall k,\ \Gamma(\mathbf{m}^k) \subseteq \Gamma(\mathbf{m}^{k+1})$ From these two inequalities, we deduce $\forall k,\ \Gamma(\mathbf{m}^{k+1}) = \Gamma(\mathbf{m}^k)$, so $\Gamma(\mathbf{m}^\bigstar) = \Gamma(\mathbf{m})$. ■

- **Claim:** if $\Gamma(\mathbf{m}) \neq \emptyset$, then $\mathbf{m}^\bigstar$ is coherent □

  Proof. Suppose that $\mathbf{m}$ is coherent. We first prove that $\forall i,j,\ \mathbf{m}^\bigstar_{ij} = \boxminus \mathbf{m}^\bigstar_{ji}$. By recurrence, one would prove that $\forall k,i,j,\ \mathbf{m}^{k+1}_{ij} = \boxminus\, \mathbf{m}^{k+1}_{ji}$ using the identity $\boxminus(\mathbf{m}^k_{ij} \sqcap (\mathbf{m}^k_{ik} \boxplus \mathbf{m}^k_{kj})) = (\boxminus \mathbf{m}^k_{ij}) \sqcap ((\boxminus \mathbf{m}^k_{ik}) \boxplus (\boxminus \mathbf{m}^k_{kj}))$ *(Hypothesis 1.5)*.

  Now, we know that $\forall i,\ \mathbf{m}^\bigstar_{ii} \sqsubseteq \mathbf{m}_{ii}$, so $\forall i,\ \gamma(\mathbf{m}^\bigstar_{ii}) \sqsubseteq \gamma(\mathbf{m}_{ii}) = \{0\}$. If for some i, $\gamma(\mathbf{m}^\bigstar_{ii}) \sqsubset \{1\}$, then $\gamma(\mathbf{m}^\bigstar_{ii}) = \emptyset$ and, obviously, $\Gamma(\mathbf{m}^\bigstar) = \emptyset$. This contradicts the fact that $\Gamma(\mathbf{m}) \neq \emptyset$ because of the preceeding point.

- **Lemma 1:** for any fixed $0 \leq i,j \leq N-1,\ 0 \leq k \leq N$, and path $\langle i = i_1,\ldots,i_n = j\rangle$ in $\mathbf{m}$ such that $i_l < k$ for $1 < l < n$, and $i_s \neq i_t$ for $1 < s < t < n$, we have $\mathbf{m}^k_{ij} \sqsubseteq \bigsqcap_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}$. □

Corollary. Applying this this lemma with $k = N$, we get: for all simple paths $\langle i = i_1, \ldots, i_n = j \rangle$, $\mathbf{m}^{\bigstar}_{ij} \sqsubseteq \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}$. □

Proof. By recurrence. The property is obvious for $k = 0$ as it is equivalent to $\mathbf{m}^0_{ij} \sqsubseteq \mathbf{m}_{ij}$ and we have $\mathbf{m}^0 = \mathbf{m}$. Suppose that the property is true for a $k < N$ and let $\langle i = i_1, \ldots, i_n = j \rangle$ be a path satisfying the hypotheses of the lemma for k+1. If $\forall l \in \{2, \ldots, n-1\}$, $i_l < k$, the property is true by recurrence hypothesis and because $\mathbf{m}^{k+1}_{ij} \sqsubseteq \mathbf{m}^k_{ij}$. On the contrary, if there exists a l such that $i_l \geq k$, we know that it is unique and that $i_l = k$. By definition of $\mathbf{m}^{k+1}$, we have $\mathbf{m}^{k+1}_{ij} \sqsubseteq \mathbf{m}^k_{ik} \boxplus \mathbf{m}^k_{kj}$. We obtain the expected result by applying the recurrence hypothesis to $\langle i = i_1, \ldots, i_l = k \rangle$ in $\mathbf{m}^k_{ik}$, and to $\langle k = i_l, \ldots, i_n = j \rangle$ in $\mathbf{m}^k_{kj}$, and using the associativity of $\boxplus$. ■

- **Lemma 2:** if, for some $0 \leq i, j < N$, $\gamma(\bigsqcap_{1 \leq n,\, \langle i = i_1, \ldots, i_n = j \rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}) = \emptyset$, then $\Gamma(\mathbf{m}) = \emptyset$. □

  Proof. Suppose that $\gamma(\bigsqcap_{1 \leq n,\, \langle i = i_1, \ldots, i_n = j \rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}) = \emptyset$, but $\Gamma(\mathbf{m}) \neq \emptyset$. Take some $(x_0, \ldots, x_{N-1}) \in \Gamma(\mathbf{m})$. For any path $\langle i = i_1, \ldots, i_n = j \rangle$, we have $\forall l \in \{1, \ldots, n-1\}$, $x_{i_{l+1}} - x_{i_l} \in \gamma(\mathbf{m}_{i_l\, i_{l+1}})$. By summation $x_j - x_i \in \gamma(\boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}})$. Thus $x_j - x_i \in \bigcap_{1 \leq n,\, \langle i = i_1, \ldots, i_n = j \rangle} \gamma(\boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}) = \gamma(\bigsqcap_{1 \leq n,\, \langle i = i_1, \ldots, i_n = j \rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}})$ *(Hypothesis 1.1)*, which is not empty. ■

- **Lemma 3:** if $\forall 0 \leq i, j < N$, $\gamma(\bigsqcap_{1 \leq n,\, \langle i = i_1, \ldots, i_n = j \rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}) \neq \emptyset$, then $\forall 0 \leq i, j < N,\ 0 \leq k \leq N$, $\bigsqcap_{1 \leq n,\, \langle i = i_1, \ldots, i_n = j \rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}} \sqsubseteq \mathbf{m}^k_{ij}$. □

  Corollary. When we set $k = N$ in the lemma, we get $\forall i, j,\ \bigsqcap_{\langle i = i_1, \ldots, i_n = j \rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}} \sqsubseteq \mathbf{m}^{\bigstar}_{ij}$. □

  Proof. By recurrence. If $k = 0$, then we have $\mathbf{m}_{ij} \sqsubseteq \mathbf{m}^0_{ij}$ because $\mathbf{m}^0 = \mathbf{m}$, so *a fortiori* the lemma is true. Suppose that the property is true for $k < N$. To prove the property for $k + 1$, we only have to prove that $\bigsqcap_{\langle i = i_1, \ldots, i_n = j \rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}} \sqsubseteq \left(\mathbf{m}^k_{ik} \boxplus \mathbf{m}^k_{kj} \right)$.
  By anti-monotonicity of $\subseteq$ in $\mathcal{P}(\mathcal{C})$ ($A \subseteq B \subseteq \mathcal{C} \implies \bigsqcap A \sqsupseteq \bigsqcap B$), we only consider the set of paths from i to j that pass through variable k:
  $$\begin{aligned}
  &\bigsqcap_{\langle i = i_1, \ldots, i_n = j \rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}} \\
  &\quad \sqsubseteq \bigsqcap_{\langle i = i_1, \ldots, i_m = k, \ldots, i_n = j \rangle} \left((\boxplus_{l=1}^{m-1} \mathbf{m}_{i_l\, i_{l+1}}) \boxplus (\boxplus_{l=m}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}) \right) \\
  &\quad = \left(\bigsqcap_{\langle i = i_1, \ldots, i_n = k \rangle} (\boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}) \right) \boxplus \\
  &\qquad \left(\bigsqcap_{\langle k = i_1, \ldots, \ldots, i_m = j \rangle} (\boxplus_{l=1}^{m-1} \mathbf{m}_{i_l\, i_{l+1}}) \right).
  \end{aligned}$$
  The last equality comes from Hypothesis 1.4 thanks to $\forall i, j$, $\gamma(\bigsqcap_{1 \leq n,\, \langle i = i_1, \ldots, i_n = j \rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}) \neq \emptyset$,
  To obtain the result, we apply the recurrence hypothesis to $\mathbf{m}^k_{ik}$ and $\mathbf{m}^k_{kj}$. ■

Remark: the restricted distributivity of $\sqcap$ over $\boxplus$ is crucial in the proof of this lemma.

- **Lemma 4:** if $\exists i,\ 0 \notin \gamma(\mathbf{m}^{\bigstar}_{ii})$, then $\Gamma(\mathbf{m}) = \emptyset$. $\square$

  Proof. Suppose that for some i, $0 \notin \gamma(\mathbf{m}^{\bigstar}_{ii})$. This means that $\forall x_i \in \mathbb{I}$, $x_i - x_i \notin \gamma(\mathbf{m}^{\bigstar}_{ii})$, so $\Gamma(\mathbf{m}^{\bigstar}) = \emptyset$. By Theorem 6.1, we get $\Gamma(\mathbf{m}) = \emptyset$. ■

- **Lemma 5:** if $\forall i,\ 0 \in \gamma(\mathbf{m}^{\bigstar}_{ii})$, then $\forall i, j$,

  $$\left(\bigsqcap_{\langle i=i_1,\ldots,i_n=j\rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}\right) = \left(\bigsqcap_{\substack{\langle i\,=\,i_1,\,\ldots\,,\,i_n\,=\,j\rangle \\ \text{simple path}}} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}\right).$$

  $\square$

  Proof. The $\sqsubseteq$ part of the equality is a direct consequence of the fact that $\sqcap$ is $\subseteq$-anti-monotonic for elements of $\mathcal{P}(\mathcal{C})$.
  For the $\sqsupseteq$ part, we prove that, for each path with at least one cycle in it, there exists a path with one simple cycle less which has a smaller $\boxplus$ sum. Let $\langle i = i_1, \ldots, i_s, \ldots, i_t = i_s, \ldots, i_n = j\rangle$ be a path and $\langle i_s, \ldots, i_t = i_s\rangle$ a simple cycle in it. By Lemma 1, $\boxplus_{l=s}^{t-1} \mathbf{m}_{i_l\, i_{l+1}} \sqsupseteq \mathbf{m}^{\bigstar}_{ii}$. By hypothesis, we have $0 \in \gamma(\mathbf{m}^{\bigstar}_{ii})$. Thus, $0 \in \gamma(\boxplus_{l=s}^{t-1} \mathbf{m}_{i_l\, i_{l+1}})$, and $\left(\boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}\right) \sqsupseteq \left(\boxplus_{l=1}^{s-1} \mathbf{m}_{i_l\, i_{l+1}}\right) \boxplus \left(\boxplus_{l=t}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}\right)$. ■

- **Lemma 6:** if $\Gamma(\mathbf{m}) \neq \emptyset$, then $\forall i, j,\ \mathbf{m}^{\bigstar}_{ij} = \bigsqcap_{\langle i=i_1,\ldots,i_n=j\rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}$, $\forall i, j, k,\ \mathbf{m}^{\bigstar}_{ij} \sqsubseteq \mathbf{m}^{\bigstar}_{ik} \boxplus \mathbf{m}^{\bigstar}_{kj}$, and $\mathbf{m}^{\bigstar\bigstar} = \mathbf{m}^{\bigstar}$. $\square$

  Proof. Suppose that $\Gamma(\mathbf{m}) \neq \emptyset$. By Lemma 2, $\forall i, j$, $\gamma(\bigsqcap_{1\leq n,\ \langle i=i_1,\ldots,i_n=j\rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}) \neq \emptyset$. Thus, we can apply Lemma 1 and 3 to get $\forall i, j,\ \bigsqcap_{\langle i=i_1,\ldots,i_n=j\rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}} \sqsubseteq \mathbf{m}^{\bigstar}_{ij} \sqsubseteq \bigsqcap_{\substack{\langle i\,=\,i_1,\,\ldots\,,\,i_n\,=\,j\rangle \\ \text{simple path}}} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}$.

  By Lemma 4, $\forall i,\ 0 \in \gamma(\mathbf{m}^{\bigstar}_{ii})$. Thus, we can apply Lemma 5 to get $\forall i, j$, $\mathbf{m}^{\bigstar}_{ij} = \bigsqcap_{\langle i=i_1,\ldots,i_n=j\rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}} = \bigsqcap_{\substack{\langle i\,=\,i_1,\,\ldots\,,\,i_n\,=\,j\rangle \\ \text{simple path}}} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}$.

  Applying a method similar to the one used in Lemma 3, we get: $\forall i, j, k$,
  $$\begin{aligned}\mathbf{m}^{\bigstar}_{ij} &= \bigsqcap_{\langle i=i_1,\ldots,i_n=j\rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}} \\ &\sqsubseteq \bigsqcap_{\langle i=i_1,\ldots,i_m=k,\ldots,i_n=j\rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}} \\ &= \left(\bigsqcap_{\langle i=i_1,\ldots,i_m=k\rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}\right) \boxplus \left(\bigsqcap_{\langle k=i_1,\ldots,i_n==j\rangle} \boxplus_{l=1}^{n-1} \mathbf{m}_{i_l\, i_{l+1}}\right) \\ &= \mathbf{m}^{\bigstar}_{ik} \boxplus \mathbf{m}^{\bigstar}_{kj}.\end{aligned}$$

  Using $\forall i, j, k,\ \mathbf{m}^{\bigstar}_{ij} \sqsubseteq \mathbf{m}^{\bigstar}_{ik} \boxplus \mathbf{m}^{\bigstar}_{kj}$ in the definition of $\mathbf{m}^{\bigstar\bigstar}$, we get, by recurrence $\forall i, j, k\ (\mathbf{m}^{\bigstar})^{k+1}_{ij} = (\mathbf{m}^{\bigstar})^{k}_{ij}$. So, $\mathbf{m}^{\bigstar\bigstar} = \mathbf{m}^{\bigstar}$. ■

- **Lemma 7:** if $\Gamma(\mathbf{m}) = \emptyset$, then $\exists i,\ 0 \notin \gamma(\mathbf{m}^{\bigstar}_{ii})$. □

  Proof. We prove this property by recurrence on the size N of the matrix. If $N = 1$, we have obviously $\Gamma(\mathbf{m}) = \{(0)\} \iff 0 \in \gamma(\mathbf{m}_{00})$, and $\Gamma(\mathbf{m}) = \emptyset \iff 0 \notin \gamma(\mathbf{m}_{00})$. By definition, we have $\mathbf{m}^{\bigstar}_{00} = \mathbf{m}_{00} \sqcap (\mathbf{m}_{00} \boxplus \mathbf{m}_{00})$, so $0 \in \gamma(\mathbf{m}_{00}) \iff 0 \in \gamma(\mathbf{m}^{\bigstar}_{00})$.
  Suppose the property is true for some N. Let $\mathbf{m}$ be a matrix of size $N+1$ such that $\forall i,\ 0 \in \gamma(\mathbf{m}^{\bigstar}_{ii})$, we prove that $\Gamma(\mathbf{m}) \neq \emptyset$. Let $\mathbf{m}'$ be the matrix of size N constructed as follows: $\forall i, j < N, \mathbf{m}'_{ij} = \mathbf{m}_{(i+1)\,(j+1)} \sqcap (\mathbf{m}_{(i+1)\,0} \boxplus \mathbf{m}_{0\,(j+1)})$. We have $\forall i, j,\ \mathbf{m}'_{ij} = \mathbf{m}^{1}_{(i+1)\,(j+1)}$, so $\forall i, j,\ \mathbf{m}'^{\bigstar}_{ij} = \mathbf{m}^{\bigstar}_{(i+1)\,(j+1)}$. We deduce that $\forall i,\ 0 \in \gamma(\mathbf{m}'^{\bigstar}_{ii})$ and, by recurrence hypothesis, $\Gamma(\mathbf{m}') \neq \emptyset$. Let us take $(x_1, \ldots, x_N) \in \Gamma(\mathbf{m}')$. $\forall 1 \leq i, j,\ x_j - x_i \in \gamma(\mathbf{m}'_{(i-1)\,(j-1)}) \subseteq \gamma(\mathbf{m}_{ij})$.
  Let us prove that we can choose x_0 such that $\forall i,\ x_0 - x_i \in \gamma(\mathbf{m}_{i0})$, and $x_i - x_0 \in \gamma(\mathbf{m}_{0i})$. This will prove that $(0, x_1 - x_0, \ldots, x_N - x_0) \in \Gamma(\mathbf{m})$, and so $\Gamma(\mathbf{m}) \neq \emptyset$.
  First remark that $x_i - x_0 \in \gamma(\mathbf{m}_{0i}) \iff x_0 - x_i \in \gamma(\boxminus \mathbf{m}_{0i}) \iff x_0 - x_i \in \gamma(\mathbf{m}_{i0})$. Consider the set $C = \gamma(\bigsqcap_{1 \leq i}(\{x_i\} \boxplus \mathbf{m}_{i0}))$. Then $C \neq \emptyset$, or else, by Hypothesis 1.3 there exists $i, j \geq 1$ such that $\gamma((x_i^{\sharp} \boxplus \mathbf{m}_{i0}) \sqcap (x_j^{\sharp} \boxplus \mathbf{m}_{j0})) = \emptyset$, that is to say $x_j - x_i \notin \gamma(\mathbf{m}_{i0} \boxplus (\boxminus \mathbf{m}_{j0})) = \gamma(\mathbf{m}_{i0} \boxplus \mathbf{m}_{0j})$, which is absurd because $x_j - x_i \in \gamma(\mathbf{m}'^{\bigstar}_{(i-1)\,(j-1)}) \subseteq \gamma(\mathbf{m}'_{(i-1)\,(j-1)}) \subseteq \gamma(\mathbf{m}_{i0} \boxplus \mathbf{m}_{0j})$. So C is not empty and we simply choose any $x_0 \in C$. ■

  Remark: the fact that we can represent singletons, and the stability of $\boxplus$ are crucial in the proof of this lemma.

- **Claim:** if $\Gamma(\mathbf{m}) \neq \emptyset$, then $\forall i_0 \neq j_0$ and $c \in \gamma(\mathbf{m}^{\bigstar}_{i_0 j_0})$, there exists $(x_0, \ldots, x_{N-1}) \in \Gamma(\mathbf{m})$ such that $x_{j_0} - x_{i_0} = c$. □

  Proof. By recurrence on N.
  The case $N = 1$ is not of interest.
  When $N = 2$ and $\Gamma(\mathbf{m}) \neq \emptyset$, $\Gamma(\mathbf{m}) = \Gamma(\mathbf{m}^{\bigstar}) = \{\ (x_0, x_1) \mid x_0 = 0,\ x_1 - x_0 \in \mathbf{m}^{\bigstar}_{01}\ \}$. We can choose, without loss of generality, $i_0 = 0$, $j_0 = 1$, so $c \in \gamma(\mathbf{m}^{\bigstar}_{01})$. Then, the property is obvious.
  Suppose the property is true for some $N > 1$ and let $\mathbf{m}$ be a matrix of size $N + 1$ with non-empty domain. We suppose also, without loss of generality, that $i_0, j_0 > 0$ ($N + 1 > 2$, so one can easily ensure $i_0, j_0 > 0$ using a simple variable permutation). We construct $\mathbf{m}'$ of size N as in Lemma 7: $\forall i, j < N, \mathbf{m}'_{ij} = \mathbf{m}_{(i+1)\,(j+1)} \sqcap (\mathbf{m}_{(i+1)\,0} \boxplus \mathbf{m}_{0\,(j+1)})$. Recall that $\forall i, j,\ \mathbf{m}'^{\bigstar}_{ij} = \mathbf{m}^{\bigstar}_{(i+1)\,(j+1)}$, so, in particular, $c \in \gamma(\mathbf{m}'^{\bigstar}_{i_0 - 1\, j_0 - 1})$.
  Applying the recurrence hypothesis to $\mathbf{m}'$, there exists $(x_1, \ldots, x_N) \in \Gamma(\mathbf{m}')$ such that $x_{j_0} - x_{i_0} \in c$. Then, we can find x_0, as in Lemma 7, such that $(0, x_1 - x_0, \ldots, x_N - x_0) \in \Gamma(\mathbf{m})$ which ends the proof. ■