An Abstract Domain Extending Difference-Bound Matrices with Disequality Constraints*

Mathias Péron and Nicolas Halbwachs

Vérimag**, Grenoble – France
{Mathias.Peron,Nicolas.Halbwachs}@imag.fr

Abstract. Knowing that two numerical variables always hold different values, at some point of a program, can be very useful, especially for analyzing aliases: if $i \neq j$, then $A[i]$ and $A[j]$ are not aliased, and this knowledge is of great help for many other program analyses. Surprisingly, disequalities are seldom considered in abstract interpretation, most of the proposed numerical domains being restricted to convex sets. In this paper, we propose to combine simple ordering properties with disequalities. "Difference-bound matrices" (or DBMs) is a domain proposed by David Dill, for expressing relations of the form "$x - y \leq c$" or "$c_1 \leq x \leq c_2$". We define dDBMs ("*disequalities* DBMs") as conjunctions of DBMs with simple disequalities of the form "$x \neq y$" or "$x \neq 0$". We give algorithms on dDBMs, for deciding the emptiness, computing a normal form, and performing the usual operations of an abstract domain. These algorithms have the same complexity ($O(n^3)$, where n is the number of variables) than those for classical DBMs, if the variables are considered to be valued in a dense set ($\mathbb{R}$ or $\mathbb{Q}$). In the arithmetic case, the emptiness decision is NP-complete, and other operations run in $O(n^5)$.

Keywords: abstract domains, alias analysis, difference-bound matrices, disequalities, static analysis.

1 Introduction

In many situations, integer variables are used to address objects: it is the case with array indexes, memory addresses and pointers in languages like C, and — this last case being the initial motivation of this work — with the addressing of devices (memories, processors, sensors,...) in systems-on-chips.

It is well-known that this kind of addressing mechanism raises *aliasing* phenomena: these aliasing problems are error-prone, can make the programs obscure, and tremendously complicate their analysis: if $i = j$, then $A[i]$ and $A[j]$ are aliased, meaning that any change to $A[i]$ implicitly changes $A[j]$. Knowing

* This work has been partially supported by the APRON project of the "ACI Sécurité et Informatique" of the French Ministry of Research.

** Verimag is a joint laboratory of Université Joseph Fourier, CNRS and INPG associated with IMAG.

B. Cook and A. Podelski (Eds.): VMCAI 2007, LNCS 4349, pp. 268–282, 2007.

that $i = j$ allows this fact to be precisely captured; knowing that $i \neq j$ allows to keep $A[j]$ unaffected by the changes to $A[i]$; ignoring whether $i = j$ or not forces any change to $A[i]$ or $A[j]$ to potentially affect (i.e., lose information about) the other. So, determining whether two addresses may or must be equal is an important goal.

Most abstract domains classically used to analyze the behavior of numerical variables (like affine equations [Kar76], intervals [CC76], octagons [Min01], octahedra [CC04], polyhedra [CH78]), take equalities into account, but cannot be used for determining disequalities, because they are *convex*. On the other hand, equality and disequality relations, considered alone, are too poor to permit an interesting analysis: the only new relations that one can deduce from a set of equalities/disequalities come from the transitivity of '=' ($(x = y \wedge y = z) \Rightarrow x = z$) and the obvious rule $(x = y \wedge x \neq z) \Rightarrow y \neq z$. This is why it is interesting to combine this kind of relations with other properties, which enrich the deduction power: in this paper, we intend to combine equalities/disequalities with ordering relations. For instance the obvious rule $(x \leq y \leq z \wedge x \neq y) \Rightarrow x \neq z$ may allow non completely trivial deductions.

Our goal is to extend an existing domain with disequalities, without increasing the complexity of the representation and operations. In this paper, we study such an extension of the domain of *difference-bound matrices* [Dil89, ACD93], used for expressing relations of the form $(c_1 \leq x \leq c_2)$ and $(c_1 \leq x - y \leq c_2)$. The simplest kind of disequalities that we can add to these inequalities, are of the form $(x - y \neq 0)$. It is enough for our initial goal, and, coupled with difference-bound matrices, they allow strict inequalities to be expressed. Now, if we consider also disequalities of the form $(x - y \neq c)$, we get systems of constraints of arbitrary size (e.g., $x - y \neq 0 \wedge x - y \neq 2 \wedge x - y \neq 4 \ldots$) which contradicts our goal of not increasing the complexity. So, we will limit ourselves to inequalities of the form $(x - y \neq 0)$ or $(x \neq 0)$.

The content of the paper is the following: Section 2 is a rapid review of the related works. In Section 3, we recall the definition of difference-bound matrices and the main algorithms used for their manipulation, in particular the use of potential graphs. In Section 4 we define "*disequalities* DBMs" (or dDBMs), which are a simple extension of DBMs with simple disequality relations of the form $(x \neq y)$. The notion of potential graph is extended into "disequal potential graph". Section 5 is devoted to the central problem of deciding emptiness of the domain of solutions of a dDBM, and of normalizing dDBMs. Two cases are distinguished, according to whether the solutions are searched in a dense numerical set (like $\mathbb{R}$ or $\mathbb{Q}$) or in the set of integers. In the dense case, we exhibit algorithms for emptiness check and normal form computation, with the same theoretical complexity as in the case of classical DBMs. In the arithmetic case, unfortunately, the emptiness problem is NP-complete, and the complexity of the computation of a normal form increases from n^3 to n^5 (n being the number of variables). However, notice that the dense domain is a correct approximation of the discrete one. Section 6 describes other classical operators on dDBMs, and Section 7 gives some simple examples of application to program analysis.

2 Related Works

Several structures for representing finite unions of convex sets have been proposed in the model-checking community. In particular, "difference decision diagrams" [MLAH99] and "clock difference diagrams" [LPWY99] are more general than the domain we consider, but with an exponential complexity.

Another way of dealing with finite unions of convex set is by using "dynamic partitioning" in abstract interpretation [Bou93, JHR99, MR05, SISG06]. This could be used, for our problem, by separating the cases $i < j$ and $i > j$. However, here also, this can involve an exponential partitioning.

Of course, some non convex abstract domains have also been proposed, like congruences [Gra91, Mas93], but their expressiveness is not comparable to our present proposal. The weakly relational domains proposed by [Min02] are a family of numerical domains, not necessarily convex, based on representation and algorithmic similar to those of DBMs. However, strict conditions on expressible constraints do not allow disequations.

In constraint logic programming, algorithms were proposed to deal with constraints on finite domains. Constraints propagation is expensive, in particular because of representation problems [HS03]. [HS97] considers a restricted class of constraints ($\pm x \pm y \leq c$), corresponding to octagons [Min01], for which they propose a polynomial solver. Disequalities are not considered, because of the NP-completeness of the satisfiability problem. However, [Pug98] notices that, if all variables are pairwise different, the satisfiability can be checked in $O(n \log n)$.

In dependence analysis (which concerns alias analysis among array elements), many approaches are based on the resolution of linear constraints (e.g., [PW98] use the Omega library). Among these works, [SW02] addresses constraints of the form ($\pm x \pm y \leq c$) and disequalities. However, they use algorithmic devoted to more general constraints (Omega Test), and they don't have the same concerns, since they don't need to compute a normal form.

About normal forms of systems of linear inequalities and disequalities, we will use Lassez's works [LM92]. Imbert [Imb93] addresses the problem of eliminating variables from such systems. All these results are too general with respect to the constraints we consider, and only apply when solutions belong to dense sets.

3 Difference-Bound Matrices [Dil89]

Difference-Bound Matrices (DBMs) are a practical representation of potential constraints ($x - y \leq c$) introduced by D. Dill [Dil89].

Let $Var = \{v_1, ..., v_{n-1}\}$ be a finite set of variables, $\mathcal{V}$ (= $\mathbb{Z}$, $\mathbb{Q}$ or $\mathbb{R}$) be the numerical set in which variables and constants take their values, and $\overline{\mathcal{V}}$ be the extension of $\mathcal{V}$ with $+\infty$, ordered as usual. Let C be a set of potential constraints ($v_i - v_j \leq c$) where $c \in \mathcal{V}$ and $v_i, v_j \in Var$. The DBM representing C is a $n \times n$ matrix M defined by (cf. Figure 2(a)):

$$M_{ij} = \inf\{c \mid (v_j - v_i \leq c) \in C\}$$

where $\inf(\emptyset) = +\infty$. In other words, if there is some constraint $v_j - v_i \leq c$ in C, then M_{ij} equals (the tightest) c, otherwise it is $+\infty$.

A special variable $v_0 \in \mathit{Var}$, always valued to zero is used to express bounds on variables: $(v_i \leq c)$ is written $(v_i - v_0 \leq c)$. The set of all possible valuations of the variables represented by a DBM M will be called its domain, and will be noted $\mathcal{D}(M)$.

Potential Graph. DBMs enjoy a useful graphical representation, called potential graphs, interpreting a DBM M as the adjacency matrix of a weighted directed graph (Figure 2(b)). In the potential graph, the variable v_0 corresponds to the node labelled by 0.

Emptiness Test and Closure. Using the potential graph representation, we understand that unfeasible sets of constrains are only those which form a circuit with a strictly negative weight in the graph. As a consequence, in order to test whether the domain of a DBM is empty, we simply have to check for the existence of such a circuit: this could be achieved in polynomial time ($O(n^3)$, e.g., with Bellman-Ford algorithm).

Because any potential graph including a strictly negative cycle is one possible representation of an empty domain, we are interested in finding a normal form for *non-empty* DBMs. Then, the shortest-path closure of their potential graph is well-defined and can be computed by the Floyd-Warshall algorithm that runs in $O(n^3)$ time (Figure 1).

```
for i ← 0 to n − 1 do
    M_ii ← 0
for k ← 0 to n − 1 do
    for i ← 0 to n − 1 do
        for j ← 0 to n − 1 do
            M_ij ← min(M_ij, M_ik + M_kj) ;
```

Fig. 1. The Floyd Warshall algorithm [CLRS90] computes the shortest-path closure of a weighted digraph represented by a matrix M

Through the potential graph, the algorithm computes, for each pair of variables, the implicit constraints obtained by summation over paths of the graph, and uses the tightest one for replacement. The resulting graph represents a DBM with the same domain as the initial one, and minimal bounds for representing this domain: it is indeed a normal form. Figure 2 is an illustration of the execution of the closure algorithm of DBMs.

Notice that, after applying the shortest-path closure, testing for strictly negative cycles can be reduced to check if there is a variable i such that $M^{\leq}_{ii} < 0$, an emptiness test running in linear time.

Classical DBMs can be ordered according to the pointwise extension of the $\leq$ order on $\mathcal{V}$: $M \trianglelefteq M' \iff \forall i, j\ M_{ij} \leq M'_{ij}$. This order has the nice property to

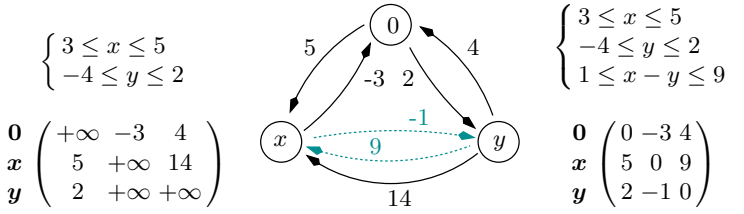


Fig. 2. Application of the closure algorithm on the DBM (a): (b) its potential graph where dashed edges are the implicit constraints computed (null-loops on each variable have been omitted) and (c) the resulting closed DBM

imply inclusion on domains: $M \trianglelefteq M' \Rightarrow \mathcal{D}(M) \subseteq \mathcal{D}(M')$. Moreover, the normal form $\overline{M}$ of a non-empty DBM M is the minimal DBM, with respect to $\trianglelefteq$, with the same domain as M: $\overline{M} = \inf_{\trianglelefteq} \{M' \mid \mathcal{D}(M') = \mathcal{D}(M)\}$.

4 Extending DBMs

Let C be a set of constraints obeying the following grammar, where $c \in \mathcal{V}$ and $v_i, v_j \in Var$.

$$\textit{constraint} \ ::= \ v_i \leq c \mid v_i - v_j \leq c \mid v_i \neq 0 \mid v_i - v_j \neq 0$$

We propose to represent C by means of a pair of matrices $(M^{\leq}, M^{\neq})$, called a dDBM (for *disequalities* DBM): A dDBM is made of a classical DBM $M^{\leq}$ with values in $\overline{\mathcal{V}}$, together with a symmetric boolean matrix $M^{\neq}$ where $M^{\neq}_{ij} = \textit{true}$ iff $(v_i \neq v_j) \in C$. We use the special variable $v_0 \in Var$ in order to represent also non-nullity constraints $(v_i \neq 0)$.

Representing disequality constraints by a matrix may seem costly in space, $M^{\neq}$ being a symmetric matrix and mostly sparse. However this representation has a trivial map with $M^{\leq}$ allowing easier reasoning later.

Domain and Order. All the possible valuations of the variables of a dDBM M will be called its domain, denoted by $\mathcal{D}(M)$. Its definition is straightforward:

$$\mathcal{D}(M) = \{(s_1, ..., s_{n-1}) \in \mathcal{V}^{n-1} \mid \exists s_0 \ \textit{such that} \ \forall i, j \in [0..n-1] \\ s_j - s_i \leq M^{\leq}_{ij} \ \wedge \ M^{\neq}_{ij} \Rightarrow s_j - s_i \neq 0 \ \wedge \ s_0 = 0\}$$

Similarly to DBMs, dDBMs can be provided with an order $\trianglelefteq$:

$$M \trianglelefteq M' \iff \forall i, j \quad M^{\leq}_{ij} \leq M'^{\leq}_{ij} \ \wedge \ M'^{\neq}_{ij} \Rightarrow M^{\neq}_{ij}$$

which enjoys the same connection with domain inclusion: $M \trianglelefteq M' \Rightarrow \mathcal{D}(M) \subseteq \mathcal{D}(M')$. Of course, as for DBMs, the converse implication is not true, because of possible redundant constraints.

Disequal Potential Graph. The disequal potential graph of a dDBM $(M^{\leq}, M^{\neq})$ is obtained by juxtaposing to the potential graph of $M^{\leq}$, the non-directed graph obtained by interpreting $M^{\neq}$ as an adjacency matrix (Figure 3). This mixed graph $\mathcal{G}(M) = (Var, E^{\leq}, E^{\neq}, w)$ is defined by:

$$\begin{array}{ll} E^{\leq} \subseteq Var \times Var & E^{\neq} \subseteq Var \times Var \\ E^{\leq} = \{(v_i, v_j) \mid M^{\leq}_{ij} < +\infty\} & E^{\neq} = \{(v_i, v_j) \mid M^{\neq}_{ij}\} \\ w \in E^{\leq} \to \mathcal{V} & \forall e = (v_i, v_j) \in E^{\leq} \quad w(e) = M^{\leq}_{ij} \end{array}$$

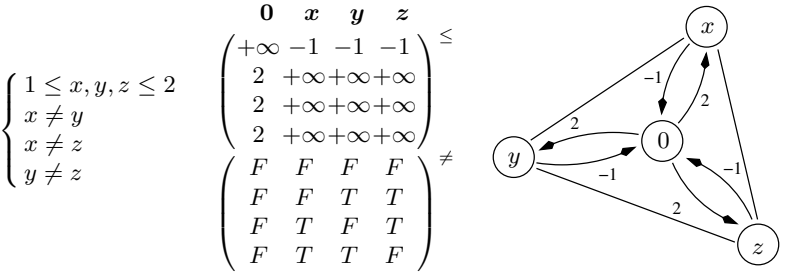


Fig. 3. (a) A set of constraints, (b) its associated dDBM and (c) its disequal potential graph

5 Emptiness Test and Normal Form

As for DBMs, we need to define a normal form, which will work for non-empty dDBMs, in order to decide equivalence of domains by a simple syntactic check, and to easily get all the consequences of given set of constraints. In this section, we provide closure algorithms to compute the normal form of a dDBM, separating the dense case ($\mathcal{V} = \mathbb{Q}$ or $\mathbb{R}$) and the arithmetic case ($\mathcal{V} = \mathbb{Z}$).

By analogy with classical DBMs, we define the normal form $\overline{M}$ of a dDBM M by:

$$\overline{M} = inf_{\trianglelefteq} \{M' \mid \mathcal{D}(M) = \mathcal{D}(M')\}$$

In the arithmetic case, such normalization will narrow some bounds for arithmetic reasons: for instance the set of constraints $\{x \neq 0, x \leq 0\}$ must be replaced by $\{x \neq 0, x \leq -1\}$. Unfortunately, this is not the only difficulties brought by arithmetic: the emptiness test problem will be in the NP-complete class [RH80].

5.1 The Dense Case

Testing Emptiness. dDBMs are extensions of DBMs by disequality constraints. Of course, the domain of a dDBM $(M^{\leq}, M^{\neq})$ can be empty because of the emptiness of the domain of $M^{\leq}$ (which we know how to check), but it can also be empty because of the disequalities.

In the case where variables take values in a dense set, the following result due to Lassez and McAloon [LM92] solves the problem, thanks to the independence of disequalities. The constraints concerned by the theorem are more general than ours, but in our special case, the result is the following:

Theorem 1 (independence of disequalities (Lassez *et al.*, 1992)). *Let I be a system of linear inequalities, and D be a finite set of linear disequalities. Then the conjunction of I and D is feasible if and only if, for each single disequality $d \in D$, the conjunction of I and $\{d\}$ is feasible.*

In other words, for a dDBM $(M^{\leq}, M^{\neq})$, if no single disequality eliminates all the solutions of $M^{\leq}$, there is no way for a *finite* number of disequalities constraints to make together the system unsatisfiable.

As a consequence, in dDBMs, the only way for a disequality constraint to make the system unsatisfiable is to contradict an equality between the corresponding variables. Thus the emptiness test boils down to check, for each disequality constraint between variables, that these variables are not forced equal by the DBM component of the dDBM (Figure 4).

```
empty ← false ;
for i ← 0 to n − 2 as long as ¬empty do
    for j ← i + 1 to n − 1 as long as ¬empty do
        if M^≠_ij then
            empty ← M^≤_ij = 0 ∧ M^≤_ji = 0
```

Fig. 4. The algorithm testing dDBM emptiness in the dense case. Runs in $O(n^2)$ time.

This test is correct when $M^{\leq}$ is in normal form: all equalities must have been expressed to perform this syntactic check.

Normal Form and Closure Algorithm. In order to compute the normal form of a dDBM $(M^{\leq}, M^{\neq})$, we can first apply the closure of DBMs to $M^{\leq}$. This makes sense because, in the dense case, disequality constraints will not involve any narrowing of the bounds in $M^{\leq}$.

Now, $M^{\neq}$ must be completed with all the disequalities resulting from the conjunction of $M^{\leq}$ and $M^{\neq}$. These inequalities are deduced according to 3 rules:

1. $v_i - v_j \leq c,\ c < 0 \ \Rightarrow\ v_i \neq v_j$
2. $v_i = v_j \ \wedge\ v_j \neq v_k \ \Rightarrow\ v_i \neq v_k$
3. $v_i \leq v_j \leq v_k \ \wedge\ v_j \neq v_k \ \Rightarrow\ v_i \neq v_k$

Rules (1) and (2) can easily be applied, in $O(n^3)$, using the disequal potential graph: rule (1) says that any arc with negative weight must be doubled by a disequality edge, rule (2) says that two equal variables are concerned with the same disequalities. The following algorithm (Figure 5) takes these rules into account ($M^{\neq}_{i*}$ and $M^{\neq}_{*j}$ respectively denote the ith row and the jth column of $M^{\neq}$):

for $i \leftarrow 0$ *to* $n-2$ **do**
  for $j \leftarrow i+1$ *to* $n-1$ **do**
    if $M^{\leq}_{ij} < 0 \vee M^{\leq}_{ji} < 0$ **then**
      $M^{\neq}_{ij} \leftarrow true$; $M^{\neq}_{ji} \leftarrow true$
    if $M^{\leq}_{ij} = 0 \wedge M^{\leq}_{ji} = 0$ **then**
      $v \leftarrow M^{\neq}_{i*} \vee M^{\neq}_{j*}$;
      $M^{\neq}_{i*} \leftarrow v$; $M^{\neq}_{*i} \leftarrow v$; $M^{\neq}_{j*} \leftarrow v$; $M^{\neq}_{*j} \leftarrow v$

Fig. 5. Algorithm applying rules (1) and (2) for deducing disequality constraints. Runs in $O(n^3)$ time.

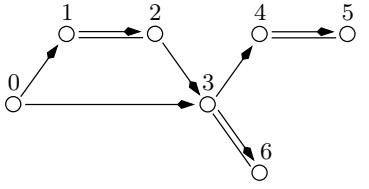
Concerning rule (3), let's first notice that this rule only concerns inequalities of the form $x \leq y$, that it, zero-weighted arcs in the disequal potential graph. Thus, the propagation of rule (3) can be done on a restriction of the disequal potential graph to zero-weighted arcs, and where nodes corresponding to equal variables are merged: let us note $G^\bullet = (V^\bullet, A^\bullet, E^\bullet)$ this reduced graph, where $(V^\bullet, A^\bullet)$ is the directed acyclic graph of zero-weighted arcs, and $(V^\bullet, E^\bullet)$ is the non-directed graph of disequalities. Let $n^\bullet$ be its number of nodes. Now, taking rule (3) into account boils down to propagating an irreflexive and symmetric relation along an order relation. This propagation can be written on $G^\bullet$ as follows:

$$\left.\begin{array}{c}(v_1,v_2) \in A^\bullet, (v_2,v_3) \in A^\bullet \\ (v_1,v_2) \in E^\bullet \vee (v_2,v_3) \in E^\bullet\end{array}\right\} \implies (v_1,v_3) \in E^\bullet$$

and is a kind of transitive closure. Among the numerous algorithms for transitive closure, Koubeck's algorithm [GK79] is particularly interesting, since its worst-case complexity is $O((n^\bullet)^2 n^\bullet_r)$ (where $n^\bullet_r$ is the number of arcs of the transitive reduction of the graph) and its average complexity is $O((n^\bullet)^2 \log n^\bullet)$ [Sim88]. A more recent paper evaluate it to $O((n^\bullet)^2)$ [SCC93].

Figure 6(a) shows an example of mixed graph, and Figure 7 gives our version of Koubeck's algorithm, adapted to solve our closure problem. The only change is that the result $\Phi(v)$ of the algorithm is no longer the set of nodes reachable from v, but its partitioning into 2 sets: the set $\Phi_1(v)$ of nodes which are reachable from v by some path traversing an arc doubled by a disequality edge, and the set $\Phi_2(v)$ of other reachable nodes. The application of the algorithm is illustrated in Figure 6(b). Notice the importance of considering successors of v in increasing topological order: if, when dealing with node 0, we start with node 3 instead of node 1, node 3 and 4 would finally belong to both $\Phi_1(0)$ and $\Phi_2(0)$.

Finally, the new disequalities resulting from rule (3) are all the pairs (v, w) with $w \in \Phi_1(v)$ and must be symmetrically reported in the initial dDBM. Notice that these new disequalities are not subject to rule (1) and take into account rule (2). The phases of complete algorithm for computing the normal form of a dDBM are given in Figure 8.



$v \in V^\bullet$	$\Phi(v) = \Phi_1(v), \Phi_2(v)$
6	$(\emptyset\ ,\ \{6\})$
5	$(\emptyset\ ,\ \{5\})$
4	$(\{5\}\ ,\ \{4\})$
3	$(\{5,6\}\ ,\ \{3,4\})$
2	$(\{5,6\}\ ,\ \{2,3,4\})$
1	$(\{2,3,4,5,6\}\ ,\ \{1\})$
0	$(\{2,3,4,5,6\}\ ,\ \{0,1\})$

(a) **(b)**

Fig. 6. (a) A mixed graph $G^\bullet = (V^\bullet, A^\bullet, E^\bullet)$, labelled in topological order, and (b) the edges to propagate with respect to the order described by arcs, for each node v in $\Phi_1(v)$

for *each* $v \in V^\bullet$ *in decreasing topological order* **do**
  $\Phi(v) \leftarrow (\emptyset, \{v\})$;
  for *each successor* w *of* v *in increasing topological order* **do**
    if $w \notin \Phi(v)$ **then**
      if $(M^\bullet)^{\neq}_{vw}$ **then**
        $\Phi_1(v) \leftarrow \Phi_1(v) \cup \Phi_1(w) \cup \Phi_2(w)$
      else
        $\Phi_1(v) \leftarrow \Phi_1(v) \cup \Phi_1(w)$; $\Phi_2(v) \leftarrow \Phi_2(v) \cup \Phi_2(w)$

Fig. 7. Algorithm computing the propagation of disequality constraints, derived from Koubeck's transitive closure algorithm. Worst-case complexity is $O((n^\bullet)^3)$ time and expected complexity is $O((n^\bullet)^2 \log n^\bullet)$ time [Sim88].

5.2 The Arithmetic Case

Testing Emptiness. Checking constraints satisfiability in $\mathbb{Z}^n$ is classically more difficult than in dense sets. Arithmetic satisfiability of DBMs and octagons is polynomial, but it is no longer the case when combined with disequalities. The dDBM of Figure 3 illustrates the problem: it has solutions in $\mathbb{R}^3$ or $\mathbb{Q}^3$, but not in $\mathbb{Z}^3$, since we can't find three distinct integers between 1 and 2. The complexity of the emptiness problem in arithmetic was studied by [RH80], who showed its NP-completeness by a reduction of the 3-coloration of graphs problem.

A brute force technique consists in considering separately, for each disequality $x - y \neq 0$, the cases $x - y \leq -1$ and $x - y \geq 1$. This leads, for d disequalities, to 2^d problems of emptiness for classical DBMs.

[SW02] suggests an improvement, allowing to decrease the number d of considered disequalities: they define an "inert" disequality, as a disequality which either eliminates *alone* all solutions of the system of inequalities or cannot participate in the absence of such solutions. Lassez theorem states that, in the dense case, all disequalities are inert. For our restricted disequalities, some inert disequalities can be easily detected in the arithmetic case: if some variable v_i involved in a disequal-

1 Apply the shortest-path closure on $M^{\leq}$ *(Figure 1)* ;
2 Add implicit disequality constraints (rules (1) and (2)) to $M^{\neq}$ *(Figure 5)* ;
3 Consider G the disequal potential graph of M where the set of arcs is restricted to those with null weight ;
4 Compute $\mathcal{SCC}$, the set of strongly connected components of the directed graph of G ;
5 Consider $G^{\bullet}$ the mixed reduced graph of G constructed on $\mathcal{SCC}$;
6 Compute $\mathcal{O}$, a topological order on the directed acyclic graph of $G^{\bullet}$;
7 Apply the disequality propagation algorithm (rule (3)) on $G^{\bullet}$ with respect to $\mathcal{O}$ *(Figure 7)* ;
8 Add induced disequality constraints into $M^{\neq}$

Fig. 8. Abstract algorithm of the closure of a dDBM M in the dense case. Runs in $O(n^3)$ time.

ity is not bounded by the system of inequalities (which can be checked in constant time, by checking if either $\overline{M}^{\leq}_{i0}$ or $\overline{M}^{\leq}_{0i}$ is $+\infty$), then the disequality is inert: either it contradicts an equation, or it can be discarded in the emptiness check.

Normal Form. The key novelty, in the arithmetic case, is that disequalities may involve a narrowing of the bounds in inequalities: $(x - y \leq 0 \ \wedge \ x \neq y) \Rightarrow (x - y \leq -1)$. Since narrowed inequalities may in turn involve new narrowings, making explicit all the consequences of a dDBM is clearly an iterative process. Figure 9 shows an example of such an iterative computation: each rewriting consists of a narrowing followed by an update of weights by Floyd-Warshall; the first rewriting corresponds to the narrowing $(y-x \leq 0 \wedge x \neq y) \Rightarrow (y-x \leq -1)$, which involves an update of $z - x \leq 1$ into $z - x \leq 0$; this new inequality is narrowed in turn into $z - x \leq -1$, which involves an update of $y - x \leq -1$ into $y - x \leq -2$ (2nd rewriting).

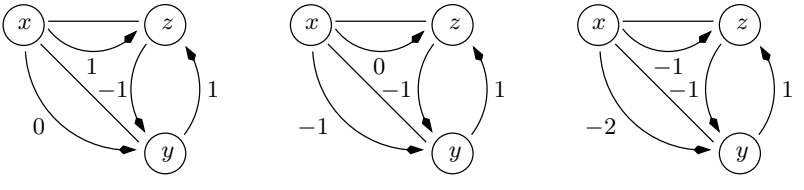


Fig. 9. Propagation of disequality constraints in the arithmetic case

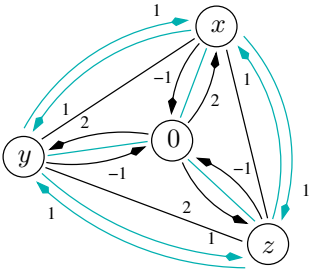
A brute force algorithm, shown in Figure 10 performs the computation in $O(n^5)$. In this algorithm, *Dense-Closure* stands for the computation of the normal form in the dense case, or, more efficiently, only steps **1** (Floyd-Warshall on inequality matrix) and **2** (propagation of rules (1) and (2)) of the algorithm of Figure 8. As a matter of fact, in the arithmetic case, rule (3) is taken into account by iterative applications of narrowing of inequalities and application of Floyd-Warshall.

repeat
  $M \leftarrow$ *Dense-Closure*(M) ;
  $to\_narrow \leftarrow \{(i,j) \mid M^{\leq}_{ij} = 0 \wedge M^{\neq}_{ij}\}$;
  forall $(i,j) \in to\_narrow$ **do**
    $M^{\leq}_{ij} \leftarrow -1$
until $to\_narrow = \emptyset$;

Fig. 10. Closure algorithm of a dDBM M in the arithmetic case. Runs in $O(n^5)$ time.

Nevertheless, this algorithm can be improved by performing weight changes from 0 to -1 on the fly, during the application of Floyd-Warshall.

Remark. The opposite disequal potential graph is the closure of the one of Figure 3(c). Although it does not contain any negative cycle, it represents an empty domain in arithmetic. It shows that testing the emptiness of the domain described by a dDBM defined in $\mathbb{Z}$ is harder than computing its normal form.



6 Operators on dDBMs

6.1 The Lattice of dDBMs

We note $\mathcal{M}$ the set of dDBMs, with a least element $\bot$ added (representing the empty set, $\mathcal{D}(\bot) = \emptyset$). $\mathcal{M}$ is partially ordered as follows:

$$M \sqsubseteq M' \Leftrightarrow \begin{cases} \text{either } M = \bot \\ \text{or} \quad\quad M, M' \neq \bot,\ M \trianglelefteq M' \end{cases}$$

The greatest dDBM, denoted $\top$ is such that, $\forall i,j = 0 \ldots n-1$, $\top^{\leq}_{ij} =$ if $i = j$ then 0 else $+\infty$, and $\top^{\neq}_{ij} = \mathit{false}$.

Lattice Operators. Let M, M' be two dDBMs in normal form. Let us note:

$$\check{M} = \begin{bmatrix} \check{M}^{\leq}_{ij} = \max(M^{\leq}_{ij}, M'^{\leq}_{ij}) \\ \check{M}^{\neq}_{ij} = M^{\neq}_{ij} \vee M'^{\neq}_{ij} \end{bmatrix} \quad , \quad \hat{M} = \begin{bmatrix} \hat{M}^{\leq}_{ij} = \min(M^{\leq}_{ij}, M'^{\leq}_{ij}) \\ \hat{M}^{\neq}_{ij} = M^{\neq}_{ij} \wedge M'^{\neq}_{ij} \end{bmatrix}$$

Then the least upper bound $M \sqcup M'$ and the greatest lower bound $M \sqcap M'$ are defined by:

$$M \sqcup M' = \begin{cases} M \text{ if } M' = \bot \\ M' \text{ if } M = \bot \\ \check{M} \text{ otherwise} \end{cases} , \quad M \sqcap M' = \begin{cases} \bot \text{ if } M = \bot \text{ or } M' = \bot \text{ or } \mathcal{D}(\hat{M}) = \emptyset \\ \hat{M} \text{ otherwise} \end{cases}$$

6.2 Other Operators

Existential quantification and projection. Both operations consist in losing all information, in a dDBM M, about a variable x, while keeping the remaining information about other variables. In the quantification $\exists x, M$, the variable x is eliminated, while in the projection $M\downarrow_x$, x is left as a non-constrained variable. Both operations need first a normalization of M, to gather all the consequences on other variables, then $\exists x, M$ is obtained by suppressing in $\overline{M}^{\leq}$ and $\overline{M}^{\neq}$ all the rows and columns corresponding to x, while for $M\downarrow_x$, the rows and columns corresponding to x in $\overline{M}^{\leq}$ (resp., in $\overline{M}^{\neq}$) must be filled with '$+\infty$' (resp., with '*false*').

Post-condition of an assignment. As usual, the abstract post-condition of an assignment $x \leftarrow e$ can be computed using existential quantification (z is a fresh variable) and projection:

$$[x \leftarrow e](M) = \exists z\,((M \wedge (z = e))\downarrow_x \wedge (x = z))$$

It will be precise when the expression e is of the form $y + c$ or c; otherwise, the term $(z = e)$ cannot be expressed in a dDBM, and all information about x is lost, unless some ad-hoc treatment is applied: for instance, if M includes the constraints $(x = y), (w \neq 0)$ then the precision of $[x \leftarrow x + w](M)$ can be improved with $(y \neq x)$.

Conditions. In order to propagate dDBMs over conditional statements, we must define the abstraction of conditions. Obviously, only conditions expressible in dDBMs can be precisely taken into account, i.e., conjunctions of conditions of the form $x - y \leq c$, $\pm x \leq c$, $x \neq y$, $x \neq 0$. The lattice operator $\sqcup$ can be used to approximate disjunctions of such conditions. Ad-hoc interpretations can be defined for some other kinds of conditions, but otherwise the abstraction will be $\top$.

Widening operator. The lattice of classical DBMs being of infinite depth, so is the lattice of dDBMs; so we must define a widening operator. However, there is no infinite chain of disequality matrices.

Consider $M, M' \in \mathcal{M}$, with $M \sqsubseteq M'$ and M' in normal form (this always improves the precision of the operator). The widening $M \nabla M'$ will remove, as usual, the inequalities of M which are not satisfied in M', but all the disequalities in M' can be kept in the result. In fact, our exact definition depends of $\mathcal{V}$: of course, we want to specialize the bounds 0 and -1 in the arithmetic case in order to preserve the constraint $(x \leq y)$ when we widen $(x < y)$ by this constraint. When $\mathcal{V} = \mathbb{Z}$, without this specialization, the inequality $(x - y \leq 0)$ would get lost in $(x - y \leq -1, x - y \neq 0) \nabla (x - y \leq 0)$.

As usual, $M \nabla M'$ is M', if $M = \bot$. Otherwise, $M \nabla M' = (M^{\nabla \leq}, M'^{\neq})$, where $\forall i, j = 0 \ldots n-1$,

$$M_{ij}^{\nabla\leq} = \begin{cases} M_{ij}^{\leq} \text{ if } M'^{\leq}_{ij} \leq M_{ij}^{\leq} \\ M'^{\leq}_{ij} \text{ if } M_{ij}^{\leq} = -1, M'^{\leq}_{ij} = 0 \text{ and } \mathcal{V} = \mathbb{Z} \\ +\infty \text{ otherwise} \end{cases}$$

7 Application to Program Analysis

A prototype analyzer has been implemented, using the general fixpoint computation engine developed by Bertrand Jeannet for NBAC [Jea].

Figure 11 gives the results of the analysis of a very simple, ad-hoc program. The goal was to show that $(x \neq y)$ at point (3).

```
(1) read(x) ; read(y) ;
if (x = y) then (2) OK
else
    while true do
        (3) if (x = y) then ERR ;
        read(z) ;
        (4) if (x <= y) then
            (5) if (y <= z) then y ← z; (6)
        else
            (7) if (x <= z) then x ← z; (8)
```

	Results
(1)	$\top$
(2)	$x = y$
(3)	$x \neq y$
(4)	$x \neq y$
(5)	$x < y$
(6)	$y = z,\ x < y,\ x < z$
(7)	$x > y$
(8)	$x = z,\ x > y,\ z > y$
ERR	$\bot$

Fig. 11. Example of a toy program and the obtained results

Other simple programs have been successfully analyzed, e.g.: a circular buffer, where we show that, when the buffer is neither full nor empty, the indexes of the first and last elements are always different; the bakery algorithm, which is proved, by means of invariants of the form $p_i \neq 0$, to properly synchronize two processes p_1 and p_2.

Beyond these simple examples, our abstract domain of dDBMs can be used in other kinds of analyzes: for instance, in Deutsch's pointer analysis [Deu94], dDBMs could be used instead of other classical abstract domains, to represent the possible aliases between two linked lists.

8 Conclusion

We have proposed a new numerical domain dealing with both potential constraints and disequalities between variables. The complexity is $O(n^3)$ when variables take their values in a dense set. In the arithmetic case, apart from the emptiness problem which is well-known to be exponential, other operations are in $O(n^5)$.

Our very rough prototype did not allow large examples to be dealt with, so our next task will be to integrate the new domain in an existing analyzer, in particular to check its effectiveness for alias analysis.

Another short-term perspective is to extend this work to octagons [Min01], where disequalities of the form $(x \neq -y)$ could also be expressed. Moreover, in this paper, we wanted, as far as possible, not to increase the complexity of the DBM domain, but if this constraint is released, we could consider more general disequalities.

References

[ACD93] R. Alur, C. Courcoubetis, and D.L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[Bou93] F. Bourdoncle. Abstract debugging of higher-order imperative languages. In *PLDI'93*, pages 46–55, New York, 1993.

[CC76] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *2nd Int. Symp. on Programming*, pages 106–130. Dunod, Paris, 1976.

[CC04] R. C. Clarisó and J. Cortadella. Verification of parametric timed circuits using octahedra. In *Designing correct circuits, DCC'04*, Barcelona, 2004.

[CH78] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *POPL'78*, pages 84–96, January 1978.

[CLRS90] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, 1990.

[Deu94] A. Deutsch. Interprocedural may-alias analysis for pointers: beyond k-limiting. In *PLDI'94*, pages 230–241, 1994.

[Dil89] D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Automatic Verification Methods for Finite State Systems*, pages 197–212. LNCS 407, Springer-Verlag, 1989.

[GK79] A. Goralcikov´a and V. Koubek. A reduct-and-closure algorithm for graphs. In *MFCS*, pages 301–307, 1979.

[Gra91] P. Granger. Static analysis of linear congruence equalities among variables of a program. In *CAAP'91*, pages 169–192, 1991.

[HS97] W. Harvey and P. J. Stuckey. A unit two variable per inequality integer constraint solver for constraint logic programming. In *Twentieth Australasian Computer Science Conference, ACSC'97*, pages 102–111, February 1997.

[HS03] W. Harvey and P. J. Stuckey. Improving linear constraint propagation by changing constraint representation. *Constraints*, 8(2):173–207, 2003.

[Imb93] J.-L. Imbert. Variable elimination for generalized linear constraints. In *ICLP'93*, pages 499–516, 1993.

[Jea] B. Jeannet. The NBAC verification/slicing tool. `http://www.inrialpes.fr/pop-art/people/bjeannet/nbac/`.

[JHR99] B. Jeannet, N. Halbwachs, and P. Raymond. Dynamic partitioning in analyses of numerical properties. In *SAS'99*, Venezia, September 1999.

[Kar76] M. Karr. Affine relationships among variables of a program. *Acta Informatica*, 6:133–151, 1976.

[LM92] J.-L. Lassez and K. McAloon. A canonical form for generalized linear constraints. *J. Symb. Comput.*, 13(1):1–24, 1992.

[LPWY99] K. G. Larsen, J. Pearson, C. Weise, and W. Yi. Clock difference diagrams. *Nordic J. of Computing*, 6(3):271–298, 1999.

[Mas93] F. Masdupuy. Semantic analysis of interval congruences. In *International Conference on Formal Methods in Programming and Their Applications*, pages 142–155, 1993.

[Min01] A. Miné. The octagon abstract domain. In *AST 2001 in WCRE 2001*, IEEE, pages 310–319. IEEE CS Press, October 2001.

[Min02] A. Miné. A few graph-based relational numerical abstract domains. In *SAS'02*, pages 117–132, London, UK, 2002. Springer-Verlag.

[MLAH99] J. B. Møller, J. Lichtenberg, H. R. Andersen, and H. Hulgaard. Difference decision diagrams. In *CSL'99*, pages 111–125. Springer-Verlag, 1999.

[MR05] L. Mauborgne and X. Rival. Trace partitioning in abstract interpretation based static analyzers. In *ESOP'05*, Edinburgh, April 2005.

[Pug98] J.-F. Puget. A fast algorithm for the bound consistency of alldiff constraints. In *AAAI'98/IAAI'98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 359–366, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

[PW98] W. Pugh and D. Wonnacott. Constraint-based array dependence analysis. *TOPLAS*, 20(3):635–678, 1998.

[RH80] D. J. Rosenkrantz and H. B. Hunt III. Processing conjunctive predicates and queries. In *VLDB*, pages 64–72, 1980.

[SCC93] K. Simon, D. Crippa, and F. Collenberg. On the distribution of the transitive closure in a random acyclic digraph. In *ESA'93*, pages 345–356. Springer-Verlag, 1993.

[Sim88] K. Simon. An improved algorithm for transitive closure on acyclic digraphs. *TCS*, 58(1-3):325–346, 1988.

[SISG06] S. Sankaranarayanan, F. Ivančić, I. Shlyakhter, and A. Gupta. Static analysis in disjunctive numerical domains. In *SAS'06*, Seoul, Korea, August 2006.

[SW02] R. Seater and D. Wonnacott. Efficient Manipulation of Disequalities During Dependence Analysis. In *LCPC'02: Proceedings of the 15th international workshop on Languages and Compilers for Parallel Computing*, volume 2481, pages 295–308, December 2002.