

# Abduction of Linear Arithmetic Constraints

Michael J. Maher

National ICT Australia  
Sydney, Australia

Michael.Maher@nicta.com.au

**Abstract.** Abduction is usually carried out on partially-defined predicates. In this paper we investigate abduction applied to fully-defined predicates, specifically linear arithmetic constraints over the real numbers. Abduction in this context has application to query answering using views and type inference, and potential relevance to analysis of concurrent/constraint/logic programs. We show that only rarely do abduction problems over linear arithmetic constraints have unique most general answers. We characterize the cases where most general answers exist. In general there may be infinitely many maximally general answers, or even answers that are not represented by maximally general answers. We take steps towards representing such answers finitely.

## 1 Introduction

Abduction is the inference rule that derives, from  $B$  and  $C$ ,  $A$  such that  $A, B \vdash C$ . It was considered by Peirce [11] to be – along with deduction and induction – one of the fundamental forms of reasoning. Mostly, abduction has been addressed in a setting of partially determined predicates or propositions. *Constraint abduction* [10] refers to abduction in a setting where  $A$ ,  $B$  and  $C$  come from a set of pre-defined, completely-defined relations (constraints, in the sense of constraint logic programming [4]) and  $A$ , the answer, ensures  $A \wedge B \rightarrow C$ .

Traditional forms of abduction differ from constraint abduction in that they address predicates that are incompletely defined by some properties. Abductive logic programming [5] and abductive constraint logic programming (ACLP) [6] differ further from constraint abduction in that the inference relation  $\vdash$  is not material implication. Furthermore, constraints are used in ACLP only in support of the abduction of predicates; constraints alone are not abduced from.

It is usual in works on abduction to require that  $A$  is consistent with  $B$ , that is, that  $A$  is *not* stronger than  $\neg B$ . However, if  $B \wedge C$  is unsatisfiable then, for any answer  $A$ ,  $A \wedge B$  is unsatisfiable. (This because if  $(A \wedge B) \rightarrow C$  then  $(A \wedge B) \rightarrow (B \wedge C)$ .)

Thus we can divide the simple problem into two cases:

- $B \wedge C$  is unsatisfiable

In this case the answers are exactly those constraints stronger than  $\neg B$ .

- $B \wedge C$  is satisfiable

In this case we are only interested in answers  $A$  that are consistent with  $B$ , that is, constraints that are *not* stronger than  $\neg B$ . Notice that this implies that the strengthening of an answer  $A$  might not be an answer

In this paper we focus on the latter case.

Clearly, determining whether such a problem has an answer is trivial: we can simply choose  $A$  to be equivalent to  $C$ . Instead, we will address the issue of characterizing all answers and/or finitely representing them. In particular, we will seek to identify when a single answer can represent all answers. In this paper we will address linear arithmetic constraints. In a companion paper [10], equational constraints over the Herbrand universe are addressed. It turns out that different kinds of constraints require different techniques to characterize the answers.

In the next section we outline some applications of constraint abduction. Then, following some brief background on constraints, constraint abduction is formally defined. In Section 5 an abstract notion of rank is introduced and several properties are established of constraint domains that support such a notion. These properties are used in characterizing answers to constraint abduction problems on linear arithmetic equalities in Section 6. Section 7 addresses the same issue for linear arithmetic inequalities but the development is more complicated, partly because this constraint domain does not support a rank.

## 2 Applications of Constraint Abduction

We outline situations in type inference and query answering with views where constraint abduction is needed. Constraint abduction is also important for analysis of logic programs [1, 3, 7], although that work focuses on finite, artificially constructed domains.

### 2.1 Type Inference

Type systems have developed in two different but compatible directions. Though these developments have addressed functional languages, they will be exemplified here in a logic language. Work on index types [17] and “practical” dependent types [15] introduces extra parameters to types which provide a refinement of the types and a consequent ability to assert type-checkable statements about functions/predicates.

For example, the type of the `append` predicate might be

$$\text{append}(\text{list}(\alpha, m), \text{list}(\alpha, n), \text{list}(\alpha, m+n))$$

which asserts, in addition to the parametric polymorphism of `append`, that the length of the third argument is the sum of the lengths of the first two arguments. Here the type `list( $\alpha$ ,  $n$ )` is defined by

$$\begin{aligned} \text{list}(\alpha, n) \text{ is } [] \text{ where } n = 0 \\ \text{or } [\alpha \mid \text{list}(\alpha, m)] \text{ where } n = m+1 \end{aligned}$$

where the length of a list is implicitly defined as part of the type. In this case, every occurrence of `append` gives rise to an equation  $x_1 + x_2 = x_3$  over integer variables when type-checking.

Work on guarded recursive types [16] presents the opportunity to refine types in a different way. These can allow different rules in the definition of a predicate to be typed differently, based on differing patterns in the head.

For example, given the type definition

```

type both( $\alpha$ ) is i( $\alpha$ ) where  $\alpha = \text{int}$ 
                    or b( $\alpha$ ) where  $\alpha = \text{bool}$ 

```

which expresses a discriminated union type, we can define

```

p(i(X), Y) :- Y = X + 1.
p(b(X), Y) :- Y = true.

```

When attempting to infer the acceptable types for  $X$  and  $Y$ , the rules of  $p$  generate expressions  $t_X = \text{int} \rightarrow t_Y = \text{int}$  and  $t_X = \text{bool} \rightarrow t_Y = \text{bool}$  where  $t_X$  and  $t_Y$  are variables representing the type of  $X$  and  $Y$  respectively.

In general, such refined types lead to similar implication constraint expressions [12]. Principal types associate a single parameterized type to each type variable, perhaps under some restrictions (such as those coming from type classes), such that other acceptable types are instances of the principal type. An algorithm is given in [13] for computing principal types, where they exist, in a guarded recursive type system. Principal types correspond to most general answers of constraint abduction problems over  $\mathcal{FT}$ , and abduction of such constraints was investigated in [10].

For the program above,  $p$  has a principal type

$$p(\text{both}(\alpha), \alpha)$$

which can be inferred as the answer  $t_X = t_Y$  of the joint constraint abduction problem involving both implication constraint expressions.

The use of both these refinements of types leads for a need to handle expressions of the form  $C_1 \rightarrow C_2$ , where the  $C_i$  are constraints involving both equations on type expressions and other constraints on index type variables. In particular, linear arithmetic constraints are useful to express relations on sizes of data structures. Although we might expect to solve such constraints with an integer constraint solver, solving the constraints over the reals has a lower complexity and experience suggests that solving the constraints over the reals is sufficient [15]. Since we would like to assign a single principal type to each expression, it turns out that we are looking for a most general answer to the constraint abduction problem involving  $C_1$  and  $C_2$ .

## 2.2 Query Answering

In database systems we sometimes want to answer a query as much as possible using previously defined relations. In a distributed database, this can limit the copying of large relations from one site to another. In data integration via a mediated schema, this provides a way to coherently query data from autonomous data sources. In both cases, previously defined relations are formulated as *views* – queries over the base relations – and the problem is to (partially) answer an input query, using the available views. We can allow pre-defined relations (constraints) in queries and views. In particular, for this paper, these are linear arithmetic constraints. See [2] for a more thorough and detailed survey of work on query answering using views than can be presented here.

If we have an input query

$$Q(\tilde{x}) : -P_1(\tilde{x}, \tilde{y}), \dots, P_n(\tilde{x}, \tilde{y}), C(\tilde{x}, \tilde{y})$$

and views

$$V_i(\tilde{u}) : -P_{i1}(\tilde{u}, \tilde{v}), \dots, P_{ik_i}(\tilde{u}, \tilde{v}), B_i(\tilde{u}, \tilde{v})$$

then we are looking for a query

$$Q'(\tilde{x}) : -V'_1(\tilde{x}, \tilde{w}), \dots, V'_m(\tilde{x}, \tilde{w}), A(\tilde{x}, \tilde{w})$$

such that the relation  $Q'$  is a subrelation of  $Q$ , independent of the data in the base relations. Here  $P$  refers to a base relation,  $V$  to a view, and  $A$ ,  $B$ , and  $C$  to constraints. Given the use of views  $V'_1, \dots, V'_m$ , and some extra conditions on either these views or the constraint domain,  $A$  is required to satisfy  $A \wedge \bigwedge_{i=1}^m B_i \rightarrow C$  to ensure that  $Q'$  is a subrelation of  $Q$ . That is,  $A$  must be an answer to the constraint abduction problem involving  $\bigwedge_{i=1}^m B_i$  and  $C$ . Obviously, a finite representation of such answers is important to the computation of the subrelation  $Q'$ .

### 3 Constraints

The syntax and semantics of constraints are defined by a constraint domain. Given a signature  $\Sigma$ , and a set of variables  $Vars$  (which we assume is infinite), a *constraint domain* is a pair  $(\mathcal{D}, \mathcal{L})$  where  $\mathcal{D}$  is a  $\Sigma$ -structure and  $\mathcal{L}$  (the language of constraints) is a set of  $\Sigma$ -formulas closed under conjunction and renaming of free variables. When  $\mathcal{D}$  is the real numbers and  $\mathcal{L}$  is all conjunctions of linear equalities (respectively, inequalities) then these constraint domains will be denoted  $\mathfrak{R}_{LinEqn}$  (resp.  $\mathfrak{R}_{LinIneq}$ ). We will also discuss constraint domains  $\mathbf{Q}_{LinEqn}$  and  $\mathbf{Q}_{LinIneq}$ , where  $\mathcal{D}$  is the rational numbers, and  $\mathbf{Z}_{LinEqn}$  and  $\mathbf{Z}_{LinIneq}$ , where  $\mathcal{D}$  is the integers. For linear constraints, constraint domains based on  $\mathfrak{R}$  or  $\mathbf{Q}$  are elementarily equivalent, so all results for  $\mathfrak{R}$  extend to  $\mathbf{Q}$ . We use  $\mathcal{FT}$  to denote the constraint domain of equations over finite terms.

For most of the results of this paper we assume that constraint languages are generated, by conjunction and variable renaming, from a set of primitive constraints. Thus a constraint can be viewed as a set of primitive constraints, and every subset of a constraint is a constraint. We say that such constraint languages are *generated from primitive constraints*. This is certainly true of the constraint languages of the two constraint domains that are the main focus of this paper:  $\mathfrak{R}_{LinEqn}$  and  $\mathfrak{R}_{LinIneq}$ . For these constraint domains we also have the property that the constraint language is closed under existential quantification:  $\forall c \in \mathcal{L} \forall x \in Vars \exists c' \in \mathcal{L} \mathcal{D} \models c' \leftrightarrow \exists x c$ . As a consequence, it will suffice to consider answers  $A$  such that  $vars(A) \subseteq vars(B) \cup vars(C)$ . (In contrast, this does not hold for  $\mathcal{FT}$  [10].) We will write  $\exists$  to express the existential closure of a formula, and sometimes use a comma to express conjunction.

We say  $C$  is *more general than*  $C'$  (or, equivalently,  $C'$  is *stronger than*  $C$  or,  $C'$  is *more specific than*  $C$ ) if  $C' \rightarrow C$ . Two constraints  $C$  and  $C'$  are *equivalent* if  $C \rightarrow C'$  and  $C' \rightarrow C$ . In this paper we assume that the syntactic representation of  $A$  is not important, so that we are interested in equivalence classes of constraints (where two constraints are equivalent if their sets of solutions are exactly the same).

The constraints modulo equivalence form a partially ordered set, where  $C_1 \leq C_2$  iff  $C_1 \rightarrow C_2$ . The poset has top element *true* and bottom element *false*. Any pair of constraints  $C_1$  and  $C_2$  has a greatest lower bound defined by  $C_1 \wedge C_2$ , as a consequence

of the assumption that  $\mathcal{L}$  is closed under conjunction. On the other hand, the existence of a least upper bound  $C_1 \sqcup C_2$  for any pair of constraints depends on the constraint domain. A *chain* is a totally ordered subset of the poset.

## 4 Constraint Abduction

We can now formally define the simple constraint abduction problem.

**Definition 1** *The Simple Constraint Abduction (SCA) Problem is as follows:*

*Given a constraint domain  $(\mathcal{D}, \mathcal{L})$ , and given two constraints  $B, C \in \mathcal{L}$  such that  $\mathcal{D} \models \exists B \wedge C$ , for what constraints  $A \in \mathcal{L}$  does*

$$\mathcal{D} \models (A \wedge B) \rightarrow C$$

*and*

$$\mathcal{D} \models \exists (A \wedge B)$$

*An instance of the problem has a fixed constraint domain and fixed constraints  $B$  and  $C$ .*

Throughout this paper,  $A$ ,  $B$  and  $C$  refer to the constraints in a simple constraint abduction problem. We omit reference to the constraint domain when it is clear from the context and, for example, simply write  $A \wedge B \rightarrow C$ . In an abuse of terminology, we often will refer to an instance as a SCA problem.

There are two classes of problem instances where the SCA problem is easy. If  $B \leftrightarrow true$  then  $C$  is an answer. If  $B \rightarrow C$  then  $true$  is an answer. We refer to these instances as *trivial*.

Of all the answers, we are most interested in the *maximally general answers*, that is, constraints  $A$  such that  $(A \wedge B) \rightarrow C$  and for every  $A'$ , if  $A \rightarrow A'$  and  $(A' \wedge B) \rightarrow C$  then  $A' \rightarrow A$ . (That is, there is no answer strictly more general than  $A$ .)

Under some circumstances, the maximally general answers represent all answers.

**Definition 2 (Abductive Ascending Chain Property)** *A problem has the Abductive Ascending Chain (AAC) property if whenever all constraints  $A_i$  in a chain satisfy  $(A_i \wedge B) \rightarrow C$ , the least upper bound of the chain exists and is an answer.*

*If this is true for every problem in a constraint domain  $(\mathcal{D}, \mathcal{L})$  then we say  $(\mathcal{D}, \mathcal{L})$  has the Abductive Ascending Chain property.*

Some constraint domains – such as  $\mathcal{FT}$ , various finite domains, and  $\mathfrak{R}_{LinEqn}$  – do not have infinite ascending chains. Thus these constraint domains vacuously have the Abductive Ascending Chain property.

If a constraint domain has the AAC property then all answers are represented by the maximally general answers.

**Proposition 1** *If a problem has the Abductive Ascending Chain property then all answers can be obtained, modulo equivalence, as a conjunction of a maximally general answer and another constraint.*

Thus, under the AAC property assumption,  $A$  is an answer iff for some maximally general answer  $A'$ ,  $A \rightarrow A'$  and  $A \wedge B$  is satisfiable.

Of particular interest are the problem instances in which one constraint represents all answers. In such cases there is a compact representation of the answers to the problem. An answer  $A$  to a SCA problem is a *most general answer* if, for every answer  $A'$  of the problem,  $A' \rightarrow A$ . Clearly a most general answer is unique up to equivalence of constraints. The main focus of this paper is on characterizing SCA problems that have a most general answer.

It is often convenient to eliminate unnecessary primitive constraints from a constraint, to simplify reasoning. A primitive constraint  $a$  in  $A$  is *redundant* in  $A$  if  $(A - a) \rightarrow a$ <sup>1</sup>.  $A$  is *redundancy-free* if there is no redundant constraint in  $A$ . A constraint  $a$  in  $A$  is *redundant* in  $A$  with respect to  $B$  if  $(A - a) \wedge B \rightarrow a$ .  $A$  is *redundancy-free* wrt  $B$  if there is no constraint in  $A$  that is redundant wrt  $B$ .

A *core* of a constraint  $C$  wrt another constraint  $B$  is a subset  $C'$  of  $C$  such that  $(B \wedge C') \leftrightarrow (B \wedge C)$ , and  $C'$  is redundancy-free wrt  $B$ . A core can be obtained from  $C$  by repeatedly deleting constraints that are redundant wrt  $B$  until there are no redundant constraints remaining.

In general, there is not a unique core.

*Example 1.* Consider the constraint domain  $\mathcal{R}_{LinEqn}$ . Let  $B$  be  $x + y = 0$  and let  $C$  be  $2x - y = 0, 3x + y = 0$ . Then each of the constraints in  $C$  is redundant wrt  $B$ , but not simultaneously. Thus each constraint in  $C$  is a core of  $C$  wrt  $B$ . Notice that the two cores are not equivalent.

Nevertheless, the replacement of  $C$  by a core does not alter the simple abduction problem.

**Proposition 2** *Let  $A, B$  and  $C$  be constraints and let  $C'$  be a core of  $C$  wrt  $B$ . Then  $A \wedge B \rightarrow C$  iff  $A \wedge B \rightarrow C'$*

Obviously, a core of  $C$  wrt  $B$  is an answer for the corresponding SCA problem. Although a core of  $C$  wrt  $B$  might seem to be a good candidate for a maximally general answer, and perhaps even a most general answer, it is not always maximally general.

*Example 2.* Consider the constraint domain  $\mathcal{FT}$ . Let  $B$  be  $x = a$  and let  $C$  be  $f(x, y) = f(a, b)$ . Then  $C$  is the core of  $C$  wrt  $B$ , but  $C$  is not a maximally general answer. This problem has a most general answer  $y = b$ .

Even if the use of a core  $C'$  of  $C$  wrt  $B$  does not immediately result in a maximally general answer, it does simplify the SCA problem by eliminating some irrelevant constraints. We can simplify further by reducing  $B$  to a core  $B'$  wrt  $C'$ . In this case we have  $(B' \wedge C') \leftrightarrow (B \wedge C)$  but the simplification does not preserve answers to the SCA problem.

<sup>1</sup> If  $A$  is a constraint that, when considered as a set of primitive constraints, contains  $a$  then we write  $A - a$  to denote the conjunction of all primitive constraints in  $A$  except  $a$ .

*Example 3.* Consider the constraint domain  $\mathfrak{R}_{LinIneq}$ . Let  $B$  be  $x + y \geq 0$ , and  $C$  be  $x \geq 0 \wedge y \geq 0$ . Then  $C'$ , the core of  $C$  wrt  $B$ , is the same as  $C$  and  $B'$ , the core of  $B$  wrt  $C'$ , is *true*. However, let  $A$  be  $x - y = 0$ . Clearly  $A \wedge B \rightarrow C$  but  $A \wedge B' \not\rightarrow C'$ .

However, it is not difficult to show that any answer to the SCA problem formed by  $B'$  and  $C'$  is also an answer to the problem formed by  $B$  and  $C$ .

We say a constraint  $A$  is *equivalent to  $C$  modulo  $B$*  if  $(A \wedge B) \leftrightarrow (B \wedge C)$ . Most general answers  $A$  have this property. Clearly, any core of  $C$  wrt  $B$  is equivalent to  $C$  modulo  $B$ . We might expect that such constraints are maximally general answers, but consider the following example. Let  $B$  be  $y = d$  and  $C$  be  $u = v$ . Two answers are  $u = v$  and  $y = d \wedge u = v$ , and both are equivalent to  $C$  modulo  $B$ , but clearly the second is not a maximally general answer.

This example is representative of how constraints equivalent to  $C$  modulo  $B$  might not be maximal. However the property of being equivalent to  $C$  modulo  $B$  is a kind of upper limit on answers to a SCA problem:

**Proposition 3** *Suppose  $A$  is equivalent to  $C$  modulo  $B$  and  $A'$  is an answer more general than  $A$ . Then  $A'$  is equivalent to  $C$  modulo  $B$ .*

*Thus if the problem has the Abductive Ascending Chain Property then every constraint equivalent to  $C$  modulo  $B$  is stronger than (or equivalent to) a maximally general answer equivalent to  $C$  modulo  $B$ .*

Clearly many maximally general answers are equivalent to  $C$  modulo  $B$ . However, not all maximally general answers have this property. In particular, it can depend on the constraint domain.

*Example 4.* Consider a SCA problem over  $\mathcal{FT}$  where  $B$  is  $y = a$  and  $C$  is  $x = h(u)$ . Apart from  $C$ , this SCA problem has the maximally general answer  $A$ :  $x = h(y) \wedge u = a$ . (Notice that  $A \not\rightarrow C$ ; for example, in a context where  $y = b$ .) Although  $A$  is maximally general it results in  $A \wedge B$  being strictly stronger than  $B \wedge C$ .

We say an answer  $A$  is *fully maximal* if  $A$  is a maximally general answer and  $A \wedge B$  is maximally general among all expressions  $A' \wedge B$  where  $A'$  is an answer. Equivalently,  $A$  is fully maximal if  $A$  is a maximally general answer and  $A$  is equivalent to  $C$  modulo  $B$ . The answer  $A$  of Example 4 is not fully maximal. In [10], fully maximal answers were proposed as a way to handle the proliferation of maximally general answers over  $\mathcal{FT}$ .

## 5 Constraint Domains with Rank

It is difficult to establish further properties of constraint abduction without introducing conditions on the constraint domain. We now consider constraint domains that support a weak notion of rank in the following sense. A well-founded set is an ordered set that does not contain an infinite, strictly-decreasing sequence.

**Definition 3** *Consider a constraint domain  $\mathcal{D}$ . Let rank be a function mapping constraints to a well-founded set. Let  $X$ ,  $Y$  and  $Z$  range over constraints. Consider the following axioms:*

1. if  $X \rightarrow Y$  then  $\text{rank}(X) \geq \text{rank}(Y)$
2. if  $X \rightarrow Y$  and  $\text{rank}(X) = \text{rank}(Y)$  then  $X \leftrightarrow Y$
3. if  $X \rightarrow Y$  and  $\text{rank}(X) > \text{rank}(Y)$  then there exists  $Z$  such that  $X \rightarrow Z$ ,  $Z \rightarrow Y$ , and  $\text{rank}(X) > \text{rank}(Z) \geq \text{rank}(Y)$

We say a constraint domain supports a rank if a function rank satisfying these axioms can be defined.

It is well-known that  $\mathfrak{R}_{LinEqn}$  supports a rank. The notion of rank for  $\mathcal{FT}$  identified in [9] also satisfies the axioms, and so  $\mathcal{FT}$  supports a rank. Although there are many abstract formulations of dimension, an abstract formulation of rank seems to have been missing.

All constraint domains that support a rank have the Abductive Ascending Chain property. As a result, maximally general answers represent all answers.

**Proposition 4** *Suppose a constraint domain  $\mathcal{D}$  supports a rank. Then  $\mathcal{D}$  has no infinite strictly increasing sequence. Consequently,  $\mathcal{D}$  has the Abductive Ascending Chain property.*

**Proposition 5** *Suppose a constraint domain  $\mathcal{D}$  supports a rank that maps constraints to the natural numbers, and satisfies*

$$\text{rank}(C_1 \wedge C_2) \leq \text{rank}(C_1) + \text{rank}(C_2)$$

for all constraints  $C_1$  and  $C_2$ .

Consider a SCA problem over  $\mathcal{D}$ .

If  $A$  is equivalent to  $C$  modulo  $B$  and  $\text{rank}(A \wedge B) = \text{rank}(A) + \text{rank}(B)$  then  $A$  is a maximally general answer of the problem.

## 6 Real Linear Equations

The constraint domain  $\mathfrak{R}_{LinEqn}$  consists of linear equations over the real numbers. Any equalities can be expressed by two inequalities. The solved form of  $\mathfrak{R}_{LinEqn}$  is a conjunction of equations of the form  $\tilde{x} = \tilde{t}(\tilde{y})$  where  $\tilde{x} \cap \tilde{y} = \emptyset$ . A solved form also represents a substitution that replace each  $x_i$  by  $t_i(\tilde{y})$ . The solved form is essentially the same as achieved by Gauss-Jordan elimination.

In this section we assume that the rank of a constraint  $A$  is defined in the traditional way: the number of independent equations in  $A$  or, equivalently, the number of non-trivial equations in the solved form of  $A$ . It is straightforward to see that the axioms for rank specified in Definition 3 are satisfied by this definition. Furthermore,  $\text{rank}(A \wedge B) \leq \text{rank}(A) + \text{rank}(B)$  for any constraints  $A$  and  $B$  in  $\mathfrak{R}_{LinEqn}$ . Thus the propositions of the previous section apply to  $\mathfrak{R}_{LinEqn}$ . The rank of a constraint is closely related to redundancy, since the rank is the number of independent equations. There are some further relationships that are relevant to this paper.

**Lemma 1.** *Suppose  $A$  and  $B$  are redundancy-free. Then the following are equivalent:*

- $\text{rank}(A \wedge B) = \text{rank}(A) + \text{rank}(B)$

- $A$  is redundancy-free wrt  $B$
- $B$  is redundancy-free wrt  $A$ .

In  $\mathfrak{R}_{LinEqn}$ , there is a close relationship between maximally general answers and constraints equivalent to  $C$  modulo  $B$ . As a result, there is no distinction between maximally general answers and fully maximal answers.

**Proposition 6** *Consider a SCA problem over  $\mathfrak{R}_{LinEqn}$ . Every maximally general answer is also a fully maximal answer.*

By Proposition 4, in  $\mathfrak{R}_{LinEqn}$  the maximally general answers represent all answers. However there may be infinitely many maximally general answers.

*Example 5.* Let  $B$  be  $y = x$  and  $C$  be  $y = -x$ . Let  $A_m$  be  $y = mx$  for any constant  $m$ . Then  $A_m$  is a maximally general answer for  $m \neq 1$ . There is similar behavior at higher dimensions.

Intuitively,  $B$  and  $C$  define intersecting, inequivalent hyperplanes. Any rotation of  $C$  about its intersection with  $B$  (except for  $B$  itself) is an answer.

As an immediate consequence of this observation we must look to characterize the maximally general answers rather than compute them individually. Fortunately, in  $\mathfrak{R}_{LinEqn}$  we can do this.

**Theorem 1.** *Consider the SCA problem over  $\mathfrak{R}_{LinEqn}$ .*

*$A$  is a maximally general answer of the problem iff  $A$  is equivalent to  $C$  modulo  $B$  and  $\text{rank}(A \wedge B) = \text{rank}(A) + \text{rank}(B)$ .*

In  $\mathfrak{R}_{LinEqn}$  a SCA problem has a most general answer only in trivial cases.

**Theorem 2.** *Consider the SCA problem over  $\mathfrak{R}_{LinEqn}$ .*

*There is a most general answer of the problem iff  $B \leftrightarrow \text{true}$  or  $B \rightarrow C$ .*

Notice that, if it exists, the most general answer is a core of  $C$  wrt  $B$ .

As a corollary to the proof of the above theorem we have a 1- $\infty$  law for the number of maximally general answers of a SCA problem over  $\mathfrak{R}_{LinEqn}$ .

**Corollary 1.** *Every SCA problem over  $\mathfrak{R}_{LinEqn}$  either has a most general answer or has infinitely many maximally general answers.*

## 7 Real Linear Inequalities

The constraint domain  $\mathfrak{R}_{LinIneq}$  consists of linear inequalities over the real numbers.

It will be convenient to write all inequalities in the form  $t \leq 0$  where  $t$  is a linear expression containing at most one occurrence of each variable, and one constant. All linear inequalities can be placed in this form.

A linear combination of inequalities  $C_i$  (or  $t_i \leq 0$ ),  $i = 1, \dots$  is an inequality  $\sum_i \alpha_i t_i + (-\delta) \leq 0$ , where the  $\alpha_i$  are constants and  $\delta$  is a non-negative constant, and

may be written  $\sum_i \alpha_i C_i + (-\delta \leq 0)$ . In a non-negative (positive) linear combination all  $\alpha_i$ 's are non-negative (respectively, positive).

If  $c$  is a single inequality  $t \leq 0$  then  $c^-$  denotes the corresponding equality  $t = 0$ . An *implicit equality* in a set of inequalities  $C$  is a non-tautologous inequality  $c$  in  $C$  such that  $C \rightarrow c^-$ . Geometrically, an equation defines a hyperplane, and the hyperplane of  $c^-$  is said to be the supporting hyperplane of  $c$  (it is the boundary of the half-space defined by  $c$ ). We say a constraint is *full dimensional* if it has no implicit equalities.

Two classes of techniques are used to prove the results in this section. The first is an algebraic approach to the duality of linear programming, based on linear combinations of inequalities, which allows us to address implicit equalities using the characterization of [8]. The second is the use of convexity, density, and topological characterizations of boundaries in the construction of answers. Unfortunately, space constraints prevent the presentation of proofs.

## 7.1 Abduction

It is substantially more difficult to address, for  $\mathfrak{R}_{LinIneq}$ , the issues addressed in the previous section for  $\mathfrak{R}_{LinEqn}$ . One problem is that  $\mathfrak{R}_{LinIneq}$  does not support a rank.

*Example 6.* Let  $X_i$  be  $x \leq 1 - \frac{1}{i}$  for  $i = 1, 2, \dots$ , so that  $X_i \rightarrow X_{i+1}$  and  $X_{i+1} \not\rightarrow X_i$ . By axioms 1 and 2, if there is a rank then  $\text{rank}(X_i) > \text{rank}(X_{i+1})$ . However this implies that the codomain of rank is not well-founded. Hence  $\mathfrak{R}_{LinIneq}$  does not support a rank.

A second problem is that, unlike many other constraint domains, the AAC property fails for  $\mathfrak{R}_{LinIneq}$ , so that the maximally general answers may not represent all answers of a SCA problem.

**Proposition 7** *The constraint domain  $\mathfrak{R}_{LinIneq}$  does not have the Abductive Ascending Chains Property.*

*Proof.* It is sufficient to exhibit one SCA problem with an ascending chain of answers where the least upper bound is not an answer. Let  $B$  be  $-1 \leq x, x \leq 0, 0 \leq y, y \leq 2$  and  $C$  be  $0 \leq x, x \leq 1, 0 \leq y, y \leq 1$ . The second constraint in  $B$  is an implicit equality in  $B \wedge C$ . Note that  $B \wedge C$  defines the line segment  $x = 0, 0 \leq y, y \leq 1$ .

Let  $A_k$  be  $x \geq 0, x \geq k(y-1)$ , where  $k$  is a rational number. Then  $A_k$  is an answer, for  $0 < k < \infty$ , and within that range for  $k$ , if  $s < t$  then  $A_t \rightarrow A_s$ . The least upper bound of  $\{A_k \mid k > 0\}$  is  $A_0$ . However,  $A_0$  is not an answer. Consequently, this SCA problem does not have a maximally general answer.

Intuitively, for smaller and smaller values of  $k > 0$ ,  $A_k$  defines a larger and larger proportion of the half-plane  $x \geq 0$ . Furthermore, the boundary of each  $A_k$  passes only through the line  $y = 1$  when  $x = 0$ , and ensures  $B \wedge A_k$  is equivalent to  $B \wedge C$ . However the least upper bound does not have this property:  $B \wedge A_0$  defines the line segment  $x = 0, 0 \leq y, y \leq 2$ , and this does not imply  $y \leq 1$  in  $C$ .

In addition to the trivial cases,  $\mathfrak{R}_{LinIneq}$  has another case where the existence of a most general answer is obvious. If  $B$  and  $C$  only involve a single variable  $x$ , say, then the problem is *one-dimensional* and the constraints are simply bounds on  $x$ . In this case there is always a most general answer. For the remainder of this section we implicitly assume that the problem is not one-dimensional in this way.

## 7.2 Abduction without Implicit Equalities

The presence or absence of implicit equalities is a major factor in the existence of most general answers. This is, perhaps, already apparent from considering  $\mathfrak{R}_{LinEqn}$ . We begin by establishing some properties that hold when implicit equalities are absent from  $B$ .

**Lemma 2.** *Consider the constraint domain  $\mathfrak{R}_{LinIneq}$ . Suppose  $A \wedge B$  is consistent,  $A$  is redundancy-free wrt  $B$ ,  $A \rightarrow A'$ , and  $A' \wedge B \rightarrow A$ .*

*If no  $B_0 \in B$  is an implicit equality in  $A \wedge B$ , then  $A' \leftrightarrow A$ .*

The lemma has a useful corollary.

**Corollary 2.** *Consider a SCA problem over the constraint domain  $\mathfrak{R}_{LinIneq}$  where no constraint in  $B$  is an implicit equality of  $B \wedge C$ .*

*If  $A$  is equivalent to  $C$  modulo  $B$  and  $A$  is redundancy-free wrt  $B$  then  $A$  is a maximally general answer and a fully maximal answer for the SCA problem.*

*In particular, any core  $C'$  of  $C$  wrt  $B$  is a maximally general answer and a fully maximal answer. If the SCA problem has a most general answer, then  $C'$  is the most general answer.*

The hypothesis that  $B$  has no implicit equalities in  $B \wedge C$  is necessary, as the following example demonstrates.

*Example 7.* Let  $B$  be  $-1 \leq x, x \leq 0, 0 \leq y, y \leq 1$  and  $C$  be  $0 \leq x, x \leq 1, -1 \leq y, y \leq 0$ . The second and third constraints in  $B$  are implicit equalities in  $B \wedge C$ .

Let  $A_k$  be  $y \leq kx$  where  $k > 0$  is a constant. Then, for each  $k$ ,  $A_k$  is a maximally general answer. To see this note that any more general constraint  $A'$  would have the form  $y \leq kx + c$  where  $c > 0$  is a constant. But then  $x = 0, y = \min(c, 1)$  is a solution of  $A' \wedge B$ , but not of  $C$ . Thus  $A'$  is not an answer.

Hence the SCA problem does not have a most general answer and, indeed, has infinitely many maximally general answers.

As we saw in Proposition 7, for some SCA problems over  $\mathfrak{R}_{LinIneq}$  not all answers are represented by maximally general answers. Nevertheless, we now turn our attention to characterizing the instances of SCA over  $\mathfrak{R}_{LinIneq}$  that have most general answers. This is substantially more complicated than for  $\mathfrak{R}_{LinEqn}$  and we will need several lemmas to develop different parts of the characterization.

**Lemma 3.** *Let  $C'$  be a core of  $C$  wrt  $B$ . Suppose  $C' \not\rightarrow B$  and no inequality in  $B$  is an implicit equality in  $B \wedge C'$ .*

*If some  $C_0 \in C'$  is an implicit equality in  $B \wedge C'$ , then there is no most general answer for the SCA problem.*

The following example illustrates this lemma.

*Example 8.* Let  $B$  be  $x \geq y$  and let  $C$  be  $y \geq 0, y \leq 0$ . Then the SCA problem satisfies the conditions of the Lemma 3. The proof of the lemma constructs the answer  $A$  as  $y \geq 0, x \leq 0$ . Then  $A \wedge B$  is satisfied only by  $x = 0, y = 0$ . Clearly  $A \not\rightarrow C$  since

$x = -1, y = 1$  satisfies  $A$ , but not  $C$ . Similarly,  $C \not\rightarrow A$  since  $x = 1, y = 0$  satisfies  $C$ , but not  $A$ . Furthermore, there is no constraint  $A'$  that is more general than both  $A$  and  $C$  and is an answer to the SCA problem: the least upper bound of  $A$  and  $C$  is  $y \geq 0$ , but this is not an answer.

We say two inequalities,  $C_1$  and  $C_2$  *oppose* each other if their corresponding hyperplanes are parallel  $C_1 \not\rightarrow C_2, C_2 \not\rightarrow C_1$ , and  $C_1 \wedge C_2$  is satisfiable. Thus  $C_1 \wedge C_2$  defines a polytope with two non-intersecting faces (or a single hyperplane if the boundaries of  $C_1$  and  $C_2$  coincide). The polytope is bounded in the direction perpendicular to the hyperplanes, but unbounded in every other orthogonal direction.

**Lemma 4.** *Let  $C'$  be a core of  $C$  wrt  $B$  and let  $B'$  be a core of  $B$  wrt  $C'$ . Suppose the SCA problem is non-trivial,  $B \wedge C$  is full-dimensional, and  $C' \not\rightarrow B$ .*

*Then there is a most general answer for the SCA problem iff  $B'$  and  $C'$  each consists of a single inequality, and they oppose each other.*

*Example 9.* If  $B$  is  $x + y \geq 2$  and  $C$  is  $x + y \leq 4$  then the most general answer to this SCA problem is  $C$ . However, if  $B$  is  $x + y \geq 2, x \geq 0$  and  $C$  is  $x + y \leq 4$  then  $C$  is a maximally general answer, but it is not more general than the answer  $2x + y \leq 4$  and hence, by Corollary 2, there is no most general answer.

We say that a constraint  $B$  *touches* a constraint  $C$  if  $C \rightarrow B$  and, for some  $B_0 \in B$ ,  $B_0 \cap C$  is satisfiable. This describes a situation where the polyhedron of  $B$  encloses the polyhedron of  $C$  but (at least) one facet of  $B$  makes glancing contact (i.e., touches) the boundary of  $C$ .

**Lemma 5.** *Let  $C'$  be a core of  $C$  wrt  $B$ . Suppose  $C' \rightarrow B$  and no inequality of  $B$  is an implicit equality of  $B \wedge C$ . Then*

*the SCA problem has a most general answer iff  $B$  does not touch  $C'$*

*Example 10.* Let  $B$  be  $y \leq x + 1$  and let  $C$  be  $x \geq 0, y \leq 1$ .  $C$  is a core of  $C$  wrt  $B$ . Then  $B$  touches  $C$  at the point  $x = 0, y = 1$ . There is an answer  $y \leq -x + 1, y \geq -x + 1$  that is not less general than  $C$ . Thus this SCA problem does not have a most general answer.

In the proof of Lemma 5 an answer is constructed that is something like  $y \leq -x + 1, y \geq -x + 1, -1 \leq x, x \leq 0$ .

We can now characterize when most general answers exist, under the assumption that  $B$  does not participate in any implicit equalities in  $B \wedge C$ .

**Theorem 3.** *Consider an SCA problem over  $\mathfrak{R}_{LinIneq}$ . Let  $C'$  be a core of  $C$  wrt  $B$  and let  $B'$  be a core of  $B$  wrt  $C'$ . Suppose no inequality of  $B$  is an implicit equality of  $B \wedge C$ .*

*The problem has a most general answer iff and only if one of the following conditions is satisfied:*

- *the problem is trivial,*
- *the problem is one-dimensional,*
- *$C' \rightarrow B$  and  $B$  does not touch  $C'$*
- *$C' \not\rightarrow B$ , and  $B'$  and  $C'$  are single opposing inequalities*

### 7.3 Abduction with Implicit Equalities

The previous section characterized the availability of a most general answer when  $B$  contains no implicit equalities. We now establish some lemmas addressing the issue in the presence of implicit equalities. If  $C$  contains both an inequality that is an implicit equality in  $B \wedge C$  and one that is not then there is no most general answer.

**Lemma 6.** *Consider a SCA problem. Let  $C'$  be a core of  $C$  wrt  $B$  and suppose it contains both an implicit equality  $C_0$  of  $B \wedge C$  and an inequality  $C_1$  that is not an implicit equality. Then this SCA problem does not have a most general answer.*

The next result is in some sense an extension of Lemma 4, since it corresponds to the case where the two opposing inequalities have the same supporting hyperplane.

**Lemma 7.** *Suppose  $B$  and  $C$  are full-dimensional, but  $B \wedge C$  is one smaller in dimension.*

*This SCA problem has a most general answer iff a core  $C'$  of  $C$  wrt  $B$  is a single inequality and a core  $B'$  of  $B$  wrt  $C'$  is a single inequality.*

*Example 11.* This example is a variant of Example 9. If  $B$  is  $x + y \geq 2$  and  $C$  is  $x + y \leq 2$  then the most general answer to this SCA problem is  $C$ . However, if  $B$  is  $x + y \geq 2, x \geq 0$  and  $C$  is  $x + y \leq 2$  then  $C$  is a maximally general answer, but it is not more general than the answer  $2x + y \leq 2$  and hence, by Corollary 2, there is no most general answer.

Except for the trivial case, if  $B$  has implicit equalities in  $B$  then there is no most general answer.

**Lemma 8.** *Suppose there is a non-tautological equation  $e$  such that  $B \rightarrow e$ . Then the SCA problem has a most general answer iff  $B \rightarrow C$ .*

Informally, this result holds because if  $e$  is  $t = 0$  then any constraint  $s \leq 0$  in a core  $C'$  of  $C$  wrt  $B$  could be replaced by  $s + t \leq 0$ , giving a different answer.

Although we cannot use a most general answer to represent all answers in this case, a simple observation allows us to reduce this problem to a simpler one.

**Lemma 9.** *Suppose  $B$  has implicit equalities and we write them as explicit equations  $\hat{\theta}$  in solved form. Then  $A \wedge B \rightarrow C$  iff  $A\theta \wedge B\theta \rightarrow C\theta$*

Thus we can extract all the implicit equalities from  $B$ , which can be done with standard techniques, and apply them as a substitution. The answers of the SCA problem on  $B\theta$  and  $C\theta$  are answers to the original problem and, furthermore, constraints that are equivalent to such an answer, modulo  $\hat{\theta}$ , are also answers to the original problem, and these are the only answers. Here  $\theta$  is the substitution corresponding to solved form  $\hat{\theta}$ .

However the substitution  $\theta$  does not simplify the problem into a form addressed in the previous subsection because  $B$  might have an implicit equality in  $B \wedge C$  that is not an implicit equality in  $B$ . For example, if  $B$  is  $x \leq y$  and  $C$  is  $y \leq z, z \leq x$  then  $B \wedge C$  implies  $x = y$ . Thus we need a stronger reduction.

**Lemma 10.** *Suppose  $B \wedge C$  has implicit equations  $\hat{\theta}$ . Then  $A \wedge B \rightarrow C$  iff  $A \wedge B \rightarrow \hat{\theta}$  and  $A\theta \wedge B\theta \rightarrow C\theta$ .*

This still does not directly simplify the problem to the cases of the previous subsection because inequalities in  $B$  might be implicit equalities in  $\hat{\theta}$ . That is, for some  $B_0 \in B$ , it might be that  $\hat{\theta} \rightarrow B_0^-$ . Nevertheless, it shows that the problem of representing answers for SCA problems can be broken down into two specialized parts: abduction from implicit equalities, and abduction in a full-dimensional context.

We can now characterize when a SCA problem has a most general answer. It is apparent that the presence of implicit equalities in  $B$  precludes most general answers, except in the case addressed in Lemma 7.

**Theorem 4.** *Consider an SCA problem over  $\mathfrak{R}_{LinIneq}$ . Let  $C'$  be a core of  $C$  wrt  $B$  and let  $B'$  be a core of  $B$  wrt  $C'$ .*

*The problem has a most general answer if and only if one of the following conditions is satisfied:*

- *the problem is trivial,*
- *the problem is one-dimensional,*
- *no inequality of  $B$  is an implicit equality of  $B \wedge C$ ,  $C' \rightarrow B$  and  $B$  does not touch  $C'$*
- *no inequality of  $B$  is an implicit equality of  $B \wedge C$ ,  $C' \not\rightarrow B$ , and  $B'$  and  $C'$  are single opposing inequalities*
- *$B$  is full-dimensional,  $B \wedge C$  is one dimension less, and both  $B'$  and  $C'$  consist of a single inequality.*

In [7] it was shown that a constraint language with primitive constraints of the form  $ax + by \otimes 0$  where  $a, b \in \{-1, 0, 1\}$ ,  $\otimes \in \{<, \leq\}$  has the property that every SCA problem has a most general answer. However, as several of our examples suggest, a slightly richer language loses this property.

## 8 Discussion

Only simple constraint abduction has been addressed in this paper, although the applications discussed earlier require extensions to this basic idea. Query answering requires that  $A$  be restricted to variables in the query  $Q'$ , while type inference requires the simultaneous solution of several SCA problems. The relationship between these extensions and SCA problems is addressed in [10]. In particular, using Proposition 6 of [10] and quantifier elimination we can determine whether a SCA problem can be solved under a given variable restriction. For  $\mathfrak{R}_{LinEqn}$  and  $\mathfrak{R}_{LinIneq}$  the quantifier elimination is relatively straightforward. Maximally general answers for the variable restricted abduction problem are simply those maximally general answers for the SCA problem that satisfy the variable restriction. Simultaneous solution of SCA problems can be reduced to solution of the individual SCA problems when the AAC property holds. However, this leaves the problem unaddressed for constraint domains like  $\mathfrak{R}_{LinIneq}$ .

While the results of this paper are also valid for linear constraints over the rational numbers, they do not necessarily extend to the integers. However, it is easy to see

that the example showing  $\mathfrak{R}_{LinIneq}$  does not have the AAC property also applies to  $\mathbf{Z}_{LinIneq}$ . This is an indication that characterizing SCA problems with most general answers over this domain will be no easier than for  $\mathfrak{R}_{LinIneq}$ .

**Acknowledgement:** I thank an anonymous referee for careful reading of an earlier draft.

## References

1. R. Giacobazzi, Abductive analysis of modular logic programs, *Journal of Logic and Computation* 8(4):457-484, 1998.
2. A.Y. Halevy, Answering queries using views: A survey, *VLDB J.* 10(4): 270–294, 2001.
3. J. M. Howe, A. King, & L. Lu, Analysing Logic Programs by Reasoning Backwards in: M. Bruynooghe and K.-K. Lau (Eds), *Program Development in Computational Logic*, LNCS 3049, 152–188, 2004.
4. J. Jaffar & M.J. Maher, Constraint Logic Programming: A Survey, *Journal of Logic Programming* 19 & 20, 503–581, 1994.
5. A.C. Kakas, R.A. Kowalski, F. Toni, Abductive Logic Programming, *J. Logic and Computation* 2(6): 719–770, 1992.
6. A.C. Kakas, A. Michael, C. Mourlas, ACLP: Abductive Constraint Logic Programming, *J. Logic Programming* 44(1-3): 129–177, 2000.
7. A. King, & L. Lu, Forward versus Backward Verification of Logic Programs, *Proc. ICLP*, 315–330, 2003.
8. J-L. Lassez & M.J. Maher, On Fourier’s Algorithm for Linear Arithmetic Constraints, *Journal of Automated Reasoning* 9, 373–379, 1992.
9. J-L. Lassez, M.J. Maher & K.G. Marriott, Unification Revisited, in: *Foundations of Deductive Databases and Logic Programming*, J. Minker (Ed), Morgan Kaufmann, 587–625, 1988.
10. M.J. Maher, Herbrand Constraint Abduction, *Proc. LICS*, 397–406, 2005.
11. C.S. Peirce, *Collected Papers of Charles Saunders Peirce*, C. Hartshorne & P. Weiss (Eds), Belknap Press of Harvard University Press, 1965.
12. V. Simonet & F. Pottier, Constraint-based type inference with guarded algebraic data types, submitted for publication, 2004. Available at [pauillac.inria.fr/~fpottier/biblio/pottier.html](http://pauillac.inria.fr/~fpottier/biblio/pottier.html)
13. P.J. Stuckey, M. Sulzmann & J. Wazny, Existentially Quantified Type Classes, draft paper, 2004.
14. J. Wang, M. Maher & R. Topor, Rewriting Unions of General Conjunctive Queries Using Views, *Proc. Conf. on Extending Database Technology*, LNCS 2287, 52–69, 2002.
15. H. Xi, *Dependent Types in Practical Programming*, Ph.D. thesis, Carnegie-Mellon University, 1998.
16. H. Xi, C. Chen & G. Chen, Guarded recursive datatype constructors, *Proc. POPL*, 224–235, 2003.
17. C. Zenger, *Indizierte Typen*, Ph.D. thesis, Universität Karlsruhe, 1999.