Abstracting Numeric Constraints with Boolean Functions

Jacob M. Howe[1] and Andy King

Computing Laboratory, University of Kent, Canterbury, CT2 7NF

Abstract

A simple, syntactic algorithm for abstracting numeric constraints for groundness analysis is presented and proved correct. The technique uses neither projection nor temporary variables, and plugs a gap in the abstract interpretation literature.

Key words: Abstract interpretation; Compilers; Constraint programming.

1 Introduction

Groundness analysis is an important theme of logic programming and abstract interpretation. Groundness analyses identify those program variables which at run time will be bound to terms that contain no variables (ground terms) [1,3]. For constraint languages, like CLP($\mathcal{R}$) [7], an analogous problem is deducing which variables are definite, that is, completely fixed by the store [2,4,5]. Groundness and definiteness are strongly related, and groundness is often used for both concepts.

Little has been written about how to abstract numeric constraints, that is, taking a numeric constraint as input and computing as output a Boolean formula that accurately describes the grounding behaviour of the constraint. For example, [4,5] just give some example groundness abstractions in a table; no algorithm for calculating an abstraction for an arbitrary constraint is described. In addition, [2] also explains the rôle of temporary variables and projection in abstraction. The procedure is as follows: first, a numeric constraint, for instance, $w = x + y * z$, is written in three-variable form, for example, $w = x + v, v = y * z$, where v is a fresh, temporary variable. Second, table lookup is used to map three-variable forms to Boolean formulae,

[1] Supported by EPSRC grant GR/MO8769.

for example $w = x + v$ and $v = y * z$ map to $f_1 = (w \leftarrow (x \wedge v)) \wedge (x \leftarrow (w \wedge v)) \wedge (v \leftarrow (w \wedge x))$ and $f_2 = (v \leftarrow (y \wedge z))$. The grounding behaviour of the constraint $w = x + v, v = y * z$ is described by $f_1 \wedge f_2$. Third, temporary variables are removed by projection, for example, v can be eliminated by $\exists v.(f_1 \wedge f_2) = (w \leftarrow (x \wedge y \wedge z)) \wedge (x \leftarrow (w \wedge y \wedge z))$.

The third step can be omitted but this typically only defers projection. Furthermore, retaining the temporary variables can degrade the time and space efficiency of many the representations of Boolean functions [1,3,5,6,8,9] that have been proposed for groundness analysis. Projection is particularly inconvenient for the analysis of [6] because variable elimination is not required elsewhere in the analysis. This paper addresses these problems by presenting a simple, syntactic algorithm for abstracting numeric constraints that neither uses projection nor introduces temporary variables. The algorithm gives abstractions which are guaranteed to be at least as precise as those given using the three variable form method.

The paper is organised as follows: Section 2 presents a semantics for (non-Herbrand) CLP($\mathcal{R}$) constraints. Abstraction is also formalised. Section 3 details three-variable form. Section 4 explains how abstraction can be recast as the problem of recognising those variables in a numeric constraint which will take a unique value when the others are grounded. Section 5 concludes.

2 Abstract Interpretation

2.1 Concrete Domain

Let $\mathbb{R}$ denote the real numbers and let Σ_N denote the set of functor symbols of CLP($\mathcal{R}$) [7], $\{+, *, /, -, \text{abs}, \text{arccos}, \text{arcsin}, \cos, \max, \min, \text{pow}, \sin\}$, where $-$ is unary minus. (Binary subtraction $x - y$ abbreviates $x + (-y)$, thus is not modeled directly). Let $\perp \notin \mathbb{R}$ denote a special symbol reserved for error handling and let V denote a denumerable set of variables. Let $X_\perp$ denote $X \cup \{\perp\}$. Put $\Sigma = \mathbb{R}_\perp \cup \Sigma_N$ and $\Sigma_V = \Sigma \cup V$. Let T and T_V denote the (ground) and (non-ground) terms generated from Σ and Σ_V respectively. A valuation is a total map, $\psi : V \to \mathbb{R}_\perp$, and the set of valuations is denoted Ψ. Let $\mathcal{D}$ be an interpretation of the symbols of Σ. $\mathcal{D}(d) = d$ for all $d \in \mathbb{R}_\perp$. Each symbol $f \in \Sigma_N$ of arity n is interpreted by $\mathcal{D}$ as a map $f : \mathbb{R}_\perp^n \to \mathbb{R}_\perp$. For example, $+ : \mathbb{R}_\perp \times \mathbb{R}_\perp \to \mathbb{R}_\perp$ maps numbers $d_1, d_2 \in \mathbb{R}$ to their sum, otherwise, if either $d_1 = \perp$ or $d_2 = \perp$, it maps to $\perp$. The other symbols in Σ_N are interpreted in the usual way, except in the following three cases: $(d/0) = \perp$ for every $d \in \mathbb{R}_\perp$, $\text{arcsin}(d) = \perp$ iff $\neg(-1 \leq d \leq 1), \text{arccos}(d) = \perp$ iff $\neg(-1 \leq d \leq 1)$. Let Π denote the set of binary constraint symbols

$\{=, \leq\}$. $\mathcal{D}$ interprets $d_1 = d_2$ as the predicate which is true iff $d_1, d_2 \in \mathbb{R}$ and $d_1 = d_2$; $\mathcal{D}$ interprets $d_1 \leq d_2$ in the obvious way. Let $\mathcal{C}$ denote the set of constraints generated by T_V and Π, which is closed under conjunction ($\wedge$), existential quantification ($\exists$) and renaming (ρ). A valuation, ψ, naturally extends to terms and constraints using the interpretation $\mathcal{D}$. Entailment of constraints, $\models$, is defined by $c_1 \models c_2$ iff $\forall \psi \in \Psi . \psi(c_1) \Rightarrow \psi(c_2)$. Equivalence, $\equiv$, is defined by $c_1 \equiv c_2$ iff $c_1 \models c_2$ and $c_2 \models c_1$. $\langle \mathcal{C}/\equiv, \models, \wedge \rangle$ is a (bounded) meet semi-lattice with the bottom and top elements $false$ and $true$, where $\models$ and $\wedge$ are lifted to equivalence classes of constraints. Let $\mathcal{P}(X)$ denote the powerset of X. For the purposes of abstract interpretation, the concrete domain is taken to be the lattice $\langle \mathcal{P}(\mathcal{C}/\equiv), \subseteq, \cup, \cap \rangle$.

In common with many constraint solvers, the solver of CLP($\mathcal{R}$) [7] is partial in the sense that it can only detect the satisfiability/unsatisfiability of constraints that become linear. To model this, let $\mathcal{L}$ denote the set of linear constraints, that is (where $d \in \mathbb{R}$), $true, false$, $\sum_{i=1}^{n} d_i * x_i = d$, $\sum_{i=1}^{n} d_i * x_i \leq d \in \mathcal{L}$. $\mathcal{L}$ is closed under conjunction, existential quantification and renaming. The transition system $\multimap \subseteq (\mathcal{L} \times \mathcal{C})^2$ is defined by: $\langle c, d \wedge d' \rangle \multimap \langle c'', d' \rangle$ iff there exists $c' \in \mathcal{L}$ such that $c \models \wedge_{i \in I}(x_i = d_i)$, $d \wedge \wedge_{i \in I}(x_i = d_i) \equiv c'$ and $c'' \equiv c \wedge c'$ (where I is a possibly empty index set). For example, $\langle true, (y = x * \sin(z) \wedge z = \pi/2) \rangle \multimap \langle 2 * z = \pi, y = x * \sin(z) \rangle \multimap \langle (x = y \wedge 2 * z = \pi), true \rangle$. Let $\multimap^\star$ abbreviate zero or more $\multimap$ transitions. The transition system $\multimap$ is confluent in that if $\langle true, c \rangle \multimap^\star \langle c', d' \rangle \not\multimap$ and $\langle true, c \rangle \multimap^\star \langle c'', d'' \rangle \not\multimap$, then $c' \equiv c''$ and $d' \equiv d''$. Let $f : \mathcal{L} \to \mathcal{L}$ be a map such that $f(c) = f(c')$ iff $c \equiv c'$. This gives the (deterministic) map $c \multimap \langle f(c'), d' \rangle$ where $\langle true, c \rangle \multimap^\star \langle c', d' \rangle \not\multimap$. This is necessary to formulate abstraction as a mapping. Let $\mathcal{C}_=$ denote those $c \in \mathcal{C}$ of form $(t = t')$ and such that $c \not\multimap \langle c', d' \rangle$ with $c' \models false$. Let $\mathrm{var}(o)$ denote the (free) variables in syntactic object o and let $\mathrm{mvar}(o)$ denote the variable occurrences in syntactic object o (as a multiset), for example, $\mathrm{mvar}(f(u, v, v)) = \{u, v, v\}$. Let $M^\downarrow$ denote the set of singularly occurring elements of the multiset M. Finally, let $\mathcal{M}(X) = \{M | M^\downarrow \in \mathcal{P}(X)\}$.

2.2 Abstract Domain

Groundness and groundness dependencies are often represented by Boolean functions [1,3]. Let X denote a finite set of variables, and let $Bool_X$ denote the set of Boolean formulae over X. Each $f \in Bool_X$ represents an $|X|$-ary Boolean function, so function and formula are used interchangeably. The formula $\wedge Y$ is sometimes written Y. A Boolean function f is positive iff $X \models f$. Let Pos_X denote the set of positive Boolean functions over X, augmented with the logical constant $false$. $\langle Pos_X, \models, \vee, \wedge \rangle$ is a complete lattice – the abstract domain. $\exists \{y_1, \ldots, y_n\}.f$ abbreviates $\exists y_1 \ldots \exists y_n . f$.

The concretisation map, $\gamma_X \colon Pos_X \to \mathcal{P}(\mathcal{C}/\!\equiv)$, details how formulae represent constraints. Concretisation is defined by

$$\gamma_X(f) = \{[c]_\equiv \mid \forall c' \in \mathcal{C} \,.\, (c \wedge c') \not\equiv false \Rightarrow assign_X(c \wedge c') \models f\}$$

Here, $assign_X \colon \mathcal{C} \to Bool_X$ is given by $assign_X(c) = Y \wedge (\wedge\{\neg y \mid y \in X \setminus Y\})$ and $Y = \{y \in X \mid \exists d \in \mathbb{R} \,.\, c \multimap \langle c', d' \rangle \wedge c' \models (y = d)\}$. The abstraction map, $\alpha_X \colon \mathcal{P}(\mathcal{C}/\!\equiv) \to Pos_X$, is defined by $\alpha_X(C) = \wedge\{f \in Pos_X \mid C \subseteq \gamma_X(f)\}$. $\alpha_X(c)$ abbreviates $\alpha_X(\{[c]_\equiv\})$.

Proposition 1 *α_X, γ_X form a Galois insertion.*

Proof. By the definition of α_X, there is a Galois connection. To show that it is an insertion, it is sufficient to demonstrate that γ_X is injective. Suppose that $\gamma_X(f_1) = \gamma_X(f_2)$ and $f_1 \neq f_2$. Then there exists $g = X_1 \wedge (\wedge\{\neg x | x \in X_2\})$, where $X = X_1 \cup X_2$ and $X_1 \cap X_2 = \emptyset$, such that (without loss of generality) $g \models f_1$, but $g \not\models f_2$. Let $c = \wedge\{x = 1 \mid x \in X_1\}$. Picking $c' = true$, it is seen that $assign_X(c \wedge c') = g$. Thus $[c]_\equiv \in \gamma_X(f_1)$, but $[c]_\equiv \notin \gamma_X(f_2)$. A contradiction. Therefore γ_X is injective. $\square$

3 Computing Abstractions With Projection

$\mathcal{C}$ is not finite and thus γ_X (α_X) cannot be interpreted as an algorithm for computing $\gamma_X(f)$ ($\alpha_X(c)$) for arbitrary $f \in Pos_X$ ($c \in \mathcal{C}$). This motivates the translation of a constraint into three-variable form.

Definition 2 *$\leadsto \subseteq \mathcal{C}^2$ is the least binary relation such that:*

(1) $(c \wedge t = t') \leadsto (c \wedge x = t \wedge x = t')$ *if* $t \notin V \cup \mathbb{R}$, $t' \notin V \cup \mathbb{R}$ *and* $x \notin \mathrm{var}(c \wedge t = t')$;
(2) $(c \wedge t = t') \leadsto (c \wedge t = t'' \wedge y = t_i)$ *if* $t \in V \cup \mathbb{R}$, $t' = f(t_1, ..., t_i, ..., t_n)$, $t'' = f(t_1, ..., y, ..., t_n)$, *where* $t_i \notin V \cup \mathbb{R}$, *and* $y \notin \mathrm{var}(c \wedge t = t')$;
(3) $(c \wedge t = t') \leadsto (c \wedge t' = t)$ *if* $t \notin V \cup \mathbb{R}$ *and* $t' \in V \cup \mathbb{R}$.

Proposition 3 *$\leadsto$ finitely terminates.*

Example 4 $(x = ((\sin(y)/2) * z) + 7) \leadsto (x = u + 7 \wedge u = (\sin(y)/2) * z) \leadsto (x = u + 7 \wedge u = v * z \wedge v = \sin(y)/2) \leadsto (x = u + 7 \wedge u = v * z \wedge v = w/2 \wedge w = \sin(y)) \not\leadsto$.

Table 1
Three-variable groundness abstraction for CLP($\mathcal{R}$)

c	$\alpha_X^{\mathsf{tbl}}(c)$	c	$\alpha_X^{\mathsf{tbl}}(c)$
$t = d * t'$	$V \leftrightarrow V'$	$t = t' * t''$	$V \leftarrow (V' \wedge V'')$
$t = t' + t''$	$f_2(V, V', V'')$	$t = -t'$	$V \leftrightarrow V'$
$t = t'/d$	$V \leftrightarrow V'$	$t = d/t'$	$V \leftrightarrow V'$
$t = t'/t''$	$f_1(V, V', V'')$	$t = \mathrm{pow}(t', t'')$	$V \leftarrow (V' \wedge V'')$
$t = \cos(t')$	$V \leftarrow V'$	$t = \sin(t')$	$V \leftarrow V'$
$t = \arccos(t')$	$V \leftrightarrow V'$	$t = \arcsin(t')$	$V \leftrightarrow V'$
$t = \min(t', t'')$	$V \leftarrow (V' \wedge V'')$	$t = \max(t', t'')$	$V \leftarrow (V' \wedge V'')$
$t = \mathrm{abs}(t')$	$V \leftarrow V'$	$t \leq t'$	true

where $d \in \mathbb{R} \setminus \{0\}$; $t, t', t'' \in \mathbb{R} \cup V$; $V = \mathrm{var}(t), V' = \mathrm{var}(t'), V'' = \mathrm{var}(t'')$; $f_1(x, y, z) = (x \leftarrow (y \wedge z)) \wedge (y \leftarrow (x \wedge z))$ and $f_2(x, y, z) = f_1(x, y, z) \wedge (z \leftarrow (x \wedge y))$

A constraint $c \in \mathcal{C}$ is said to be in three-variable form iff $c \not\rightsquigarrow$. Let $\rightsquigarrow^\star$ abbreviate zero or more reductions. This leads to the following abstraction technique:

Definition 5 *The abstraction map $\alpha_X^{\mathsf{tvf}}: \mathcal{C} \to Pos_X$ is defined as follows. Suppose $c \multimap \langle c', d' \rangle$. If $c' \models false$ then $\alpha_X^{\mathsf{tvf}}(c) = false$, else $\alpha_X^{\mathsf{tvf}}(c) = \alpha_X^{\mathsf{tvf}'}(c' \wedge d')$ where*

$$\alpha_X^{\mathsf{tvf}'}(c) = \begin{cases} \alpha_X^{\mathsf{tvf}'}(c') \wedge \alpha_X^{\mathsf{tvf}'}(c'') & \text{if } c = c' \wedge c'' \\ \exists(\mathrm{var}(c') \setminus \mathrm{var}(c)).\alpha_X^{\mathsf{tvf}'}(c') & \text{if } c \rightsquigarrow^\star c' \not\rightsquigarrow \text{ and } c \neq c'' \wedge c''' \\ \alpha_X^{\mathsf{tbl}}(c) & \text{otherwise.} \end{cases}$$

The reduction $c \multimap \langle c', d' \rangle$ can be performed using CLP($\mathcal{R}$) machinery [7]. Table 1 defines $\alpha_X^{\mathsf{tbl}}(c)$ for a (non-compound) three-variable constraint c. To see that α_X^{tvf} is well-defined, let $c \rightsquigarrow^\star c_1 \not\rightsquigarrow$ and $c \rightsquigarrow^\star c_2 \not\rightsquigarrow$. A renaming $\rho : Y_1 \to Y_2$ exists with $Y_1 = \mathrm{var}(c_1) \setminus \mathrm{var}(c)$, $Y_2 = \mathrm{var}(c_2) \setminus \mathrm{var}(c)$ and $\rho(c_1) = c_2$. Moreover, $\exists Y_2.\alpha_X^{\mathsf{tvf}}(c_2) = \exists \rho(Y_1).\alpha_X^{\mathsf{tvf}}(\rho(c_1)) = \exists \rho(Y_1).\rho(\alpha_X^{\mathsf{tvf}}(c_1)) = \exists Y_1.\alpha_X^{\mathsf{tvf}}(c_1)$ since ρ is bijective and $\mathrm{var}(\rho(\alpha_X^{\mathsf{tvf}}(c_1))) \setminus \rho(Y_1) = \mathrm{var}(\alpha_X^{\mathsf{tvf}}(c_1)) \setminus Y_1$. Intuitively, α_X^{tvf} is well-defined since any extra variables introduced by $\rightsquigarrow$ are eliminated. Observe $\alpha_X(c) \models \alpha_X^{\mathsf{tvf}}(c)$ for all $c \in \mathcal{C}$.

Proposition 6 *Table 1 is safe, that is, $\alpha_X(c) \models \alpha_X^{\mathsf{tbl}}(c)$, where c is in three-variable form, $c \neq c' \wedge c''$ and $\mathrm{var}(c) \subseteq X$.*

Proof. Safety is demonstrated only for the case $x = y * z$; other cases may be treated similarly. Let $\{x, y, z\} \subseteq X$. Assume, for the sake of a contradiction, that there exists $c' \in \mathcal{C}$ such that $assign_X(x = y * z \wedge c') \not\models x \leftarrow (y \wedge z)$. Thus $assign_X(x = y*z \wedge c') = (\neg x) \wedge y \wedge z$ and $\langle true, x = y*z \wedge c' \rangle \multimap^\star \langle c'', d'' \rangle$ where $c'' \models (y = d_1) \wedge (z = d_2)$. Hence $\langle c'', d'' \rangle \multimap^\star \langle c''', d''' \rangle$ where $c''' \models (x = d_1 * d_2)$ which contradicts $assign_X(x = y*z \wedge c') = (\neg x) \wedge y \wedge z$ and the assumption. $\square$

Observe that $\alpha_X \neq \alpha_X{}^{\mathsf{tvf}}$ since $\alpha_X^{\mathsf{tvf}}(x - y * (1/x) = 0) = true$ and $\alpha_X(x - y * (1/x)) = (y \leftarrow x)$. Moreover, it should be noted that the table does not accurately describe the grounding behaviour of some unusual (and specific) constraints in three-variable form. For example, $\alpha_X(x = \min(y, y)) = x \leftrightarrow y$, whereas $\alpha_X^{\mathsf{tbl}}(x = \min(y, y)) = x \leftarrow y$. In practice it is expected that such constraints will not occur, however, the table could be extended to include these extra cases.

4 Computing Abstractions Without Projection

Abstraction may be recast as the problem of recognising those variables in a constraint which will take a unique value when the other variables in the expression are grounded. This is achieved precisely by the map det^*. The approach is formulated in terms of approximations to det^*, and is at least as accurate as the three-variable form method.

Definition 7 *The map* $\mathsf{det}^* : \mathcal{C}_= \rightarrow \mathcal{P}(V)$ *is given by* $x \in \mathsf{det}^*(c)$ *iff* $\forall \phi : \mathrm{var}(c) \setminus \{x\} \rightarrow \mathbb{R}.\exists! d \in \mathbb{R}_\perp.\phi(c) \equiv (x = d)$.

An abstraction map could be defined in terms of det^*. However, computing this may require non-trivial symbolic manipulation of c. For example, $\mathsf{det}^*(x = y * (1/z)) = \{x, y\}$ requires the recognition that $y * (1/z) \equiv y/z$. To build an abstraction map in terms of a simple pattern recogniser (together with $\multimap$) det^* is approximated by a class of maps $\mathcal{D}et$.

Definition 8 $\mathcal{D}et$ *is the least set of maps* $\mathsf{det} : \mathcal{C}_= \rightarrow \mathcal{P}(V)$ *that satisfy (where* $f \in \Sigma_N$*):*

safety: $\mathsf{det}(c) \subseteq \mathsf{det}^*(c)$;
precision 1: *if* $c = (x = t)$ *and* $c \not\multimap$, *then* $\mathrm{mvar}(c)^{\downarrow} \cap \mathsf{det}^*(c) \subseteq \mathsf{det}(c)$;
precision 2: *if* $y \in \mathsf{det}(x = t)$ *and* $y \notin \mathrm{var}(t')$, *then* $y \in \mathsf{det}(t = t')$;
precision 3: *let* $c = (t = f(t_1, ..., t_i, ..., t_n))$ *and* $c' = (t = f(t_1, ..., y, ..., t_n))$, *with* $y \notin \mathrm{var}(c)$, *if* $y \in \mathsf{det}(c')$, *then* $\mathsf{det}(y = t_i) \setminus \mathrm{var}(c') \subseteq \mathsf{det}(c)$ *and* $\mathsf{det}(c') \setminus \mathrm{var}(y = t_i) \subseteq \mathsf{det}(c)$.

The conditions in definition 8 relate to the abstraction map in the following way. The safety condition ensures $\alpha_X(c) \models \alpha_X^{\mathsf{det}}(c)$ if $\mathsf{det} \in \mathcal{D}et$. Precision 1 guarantees $\alpha_X^{\mathsf{det}}(c) \models \alpha_X^{\mathsf{tvf}}(c)$ for a non-compound three-variable constraint c and the compositionality properties of precision 2 and precision 3 (with precision 1) ensures that $\alpha_X^{\mathsf{det}}(c) \models \alpha_X^{\mathsf{tvf}}(c)$ for arbitrary c.

Proposition 9 $\mathsf{det}^* \in \mathcal{D}et$.

Proof. Only precision 3 is non-trivial. Suppose that $x \in \mathsf{det}^*(y = t_i) \setminus \mathrm{var}(c')$. It is demonstrated that $x \in \det^*(c)$. Since $y \in \mathsf{det}^*(c)$ and $x \notin \mathrm{var}(c')$, and given $\phi : \mathrm{var}(c) \setminus \{x\} \to \mathbb{R}$, it can be seen that there is a unique $d \in \mathbb{R}$ such that $\phi(c') \equiv (y = d)$. Put $\phi' = \phi \cup \{y \mapsto d\}$. Hence:

$$
\begin{array}{ll}
 & \phi(t = f(t_1, ..., t_i, ..., t_n)) \\
\equiv & \phi'(t = f(t_1, ..., t_i, ..., t_n)) \\
\equiv & \phi'(t = f(t_1, ..., y, ..., t_n) \wedge y = t_i) \\
\equiv & \phi'(y = t_i) \\
\equiv & x = d' \qquad\qquad\qquad\qquad \text{since } x \in \mathsf{det}^*(y = t_i)
\end{array}
$$

Hence $x \in \mathsf{det}^*(c)$. Similarly, the second condition holds. □

An abstraction map, parameterised by det, can now be defined.

Definition 10 *The abstraction map* $\alpha_X^{\mathsf{det}}: \mathcal{C} \to Pos_X$ *is defined as follows. Suppose* $c \multimap \langle c', d' \rangle$. *If* $c' \models false$ *then* $\alpha_X^{\mathsf{det}}(c) = false$, *else* $\alpha_X^{\mathsf{det}}(c) = \alpha_X^{\mathsf{det}'}(c' \wedge d')$ *where*

$$
\alpha_X^{\mathsf{det}'}(c) = \begin{cases} true & \text{if } c = (t \leq t') \\ \alpha_X^{\mathsf{det}'}(c') \wedge \alpha_X^{\mathsf{det}'}(c'') & \text{if } c = (c' \wedge c'') \\ \wedge_{v \in \mathsf{det}(c)}(v \leftarrow (\mathrm{var}(c) \setminus \{v\})) & \text{otherwise} \end{cases}
$$

Theorem 11 *If* $c \in \mathcal{C}$ *and* $\mathsf{det} \in \mathcal{D}et$, *then* $\alpha_X(c) \models \alpha_X^{\mathsf{det}}(c) \models \alpha_X^{\mathsf{tvf}}(c)$.

Proof. The first entailment is established by demonstrating the any Boolean formula that is entailed by α_X^{det} is also entailed by α_X. The second entailment is established by demonstrating that each $\leadsto$ reduction results in a constraint whose α_X^{det} abstraction is not stronger than that of the previous constraint. The base case demonstrates that $\alpha_X^{\mathsf{det}}(c) \models \alpha_X^{\mathsf{tvf}}(c)$ for c in the lookup table.

Consider $c \in \mathcal{C}$ such that $c \multimap \langle c', d' \rangle$. If $c' \models false$, the result is immediate. Let $\mathsf{det} \in \mathcal{D}et$.

To show $\alpha_X(c) \models \alpha_X^{\mathsf{det}}(c)$, consider $c' \wedge d' = c_1 \wedge ... \wedge c_n$. If $\alpha_X(c_i) \models \alpha_X^{\mathsf{det}'}(c_i)$, then $\alpha_X(c) \models \alpha_X^{\mathsf{det}}(c)$. Suppose, for a contradiction, that $\alpha_X(c_i) \not\models \alpha_X^{\mathsf{det}'}(c_i)$, for some c_i. Then there exists $f_x = x \leftarrow Y$, where $x \in \mathsf{det}(c_i)$ and $Y = \mathrm{var}(c_i) \setminus \{x\}$, such that $\exists c'' \in \mathcal{C}.assign_X(c_i \wedge c'') \not\models f_x$ and $c_i \wedge c'' \not\equiv false$. Hence $assign_X(c_i \wedge c'') \models (\neg x) \wedge Y$. Thus for every $y \in Y$ there is $e \in \mathbb{R}$ such that $c_i \wedge c'' \multimap^\star \langle c''', d''' \rangle$ and $c''' \models (y = e)$; indeed, it may be assumed that $\forall y \in Y. \exists e \in \mathbb{R}. c''' \models (y = e)$. By safety, since $x \in \mathsf{det}(c_i)$, for some $e' \in \mathbb{R}$, $c_i \wedge c''' \models (x = e')$. Therefore, $c_i \wedge c'' \multimap^\star \langle c''', d''' \rangle \multimap^\star \langle c'''', d'''' \rangle$ and $c'''' \models (x = e')$. A contradiction. Thus $\alpha_X(c_i) \models f_x$ and the result follows.

To show $\alpha_X^{\mathsf{det}}(c) \models \alpha_X^{\mathsf{tvf}}(c)$. Let $c' \wedge d' = c_1 \wedge ... \wedge c_n$. It is enough to show that $\alpha_X^{\mathsf{det}}(c_i) \models \alpha_X^{\mathsf{tvf}}(c_i)$. Proof is by induction in the length of $\leadsto$. For the base case, consider $c'' = (t = f(t_1, \ldots, t_n))$, where $f \in \Sigma_N$ and $c'' \not\rightsquigarrow$ (the $\leq$ case is obvious). Suppose $\alpha_X^{\mathsf{tvf}}(c'') \models (x \leftarrow Y)$, where $x \notin Y$. By inspection of Table 1, $x \in (\mathrm{mvar}(c''))^{\downarrow}$. By Proposition 6, $\alpha_X(c'') \models \alpha_X^{\mathsf{tvf}}(c'')$. Hence $\alpha_X(c'') \models (x \leftarrow Y)$. Let $\phi : \mathrm{var}(c'') \setminus \{x\} \to \mathbb{R}$ and put $c''' = \wedge\{y = \phi(y) | y \in Y\}$. Hence either $c'' \wedge c''' \multimap^{\star} \langle c'''', d''''\rangle$ and $c'''' \models false$, in which case $x \in \mathsf{det}^*(c'')$, since $\phi(c'') \equiv (x = \bot)$, or $c'' \wedge c''' \multimap \langle c''''', d'''''\rangle$ and $c''''' \models (x = e)$, in which case $(x = e) \equiv \phi(c'' \wedge c''') \equiv \phi(c'')$ and $x \in \mathsf{det}^*(c'')$. Since $x \in \mathsf{det}^*(c'')$, by precision 1, $x \in \mathsf{det}(c'')$. By inspection of Table 1, $Y = \mathrm{var}(c'') \setminus \{x\}$. Hence $\alpha_X^{\mathsf{det}}(c'') \models \alpha_X^{\mathsf{tvf}}(c'')$.

For the inductive case, suppose $c_i \rightsquigarrow c'' \rightsquigarrow^{\star} c''' \not\rightsquigarrow$. By hypothesis, $\alpha_X^{\mathsf{det}}(c'') \models \exists(\mathrm{var}(c''') \setminus \mathrm{var}(c'')).\alpha_X^{\mathsf{tvf}}(c''')$, so it is enough to show $\alpha_X^{\mathsf{det}}(c_i) \models \exists(\mathrm{var}(c'') \setminus \mathrm{var}(c_i)).\alpha_X^{\mathsf{det}}(c'')$.

(1) To show $\alpha_X^{\mathsf{det}}(t = t') \models \exists x.\alpha_X^{\mathsf{det}}(x = t \wedge x = t')$. Since $\exists x.\alpha_X^{\mathsf{det}}(x = t \wedge x = t')$

$$\begin{aligned} &= \exists x.(\alpha_X^{\mathsf{det}}(x = t) \wedge \alpha_X^{\mathsf{det}}(x = t')) \\ &= \exists x.((\wedge_{v \in \mathsf{det}(x=t)} v \leftarrow (\mathrm{var}(x = t) \setminus \{v\})) \\ &\qquad \wedge(\wedge_{u \in \mathsf{det}(x=t')} u \leftarrow (\mathrm{var}(x = t') \setminus \{u\}))) = f \end{aligned}$$

Suppose $f \models (y \leftarrow Y)$. To show $\wedge_{v \in \mathsf{det}(t=t')}(v \leftarrow \mathrm{var}(t = t') \setminus \{v\}) \models (y \leftarrow Y)$, suppose, without loss of generality, $y \in \mathsf{det}(x = t')$. Since $x \in \mathsf{det}(x = t')$, $y \notin \mathrm{var}(x = t')$. By precision 2, $y \in \mathsf{det}(t = t')$ and thus the result follows.

(2) To show $\alpha_X^{\mathsf{det}}(t = t') = \exists y.\alpha_X^{\mathsf{det}}(t = t'' \wedge y = t_i)$, where $t' = f(t_1, ..., t_i, ..., t_n)$, $t'' = f(t_1, ..., y, ..., t_n)$ and $y \notin \mathrm{var}(t')$. Since $\exists y.\alpha_X^{\mathsf{det}}(t = t'' \wedge y = t_i)$

$$\begin{aligned} &= \exists y.(\alpha_X^{\mathsf{det}}(t = t'') \wedge \alpha_X^{\mathsf{det}}(y = t_i)) \\ &= \exists y.((\wedge_{v \in \mathsf{det}(t=t'')} v \leftarrow (\mathrm{var}(t = t'') \setminus \{v\}))) \\ &\qquad \wedge(\wedge_{u \in \mathsf{det}(y=t_i)} u \leftarrow (\mathrm{var}(y = t_i) \setminus \{u\}))) \end{aligned}$$

Using precision 3, the result can be established analogously to the previous case.

(3) To show $\alpha_X^{\mathsf{det}}(t = t') = \alpha_X^{\mathsf{det}}(t' = t)$, where $t' \in \mathbb{R} \cup V$ and $t \notin \mathbb{R} \cup V$. Immediate. $\square$

Next, a specific map in $\mathsf{det}_1 \in \mathcal{D}et$ is described. The map syntactically identifies those variables that occur once in a numeric constraint expression and which take a unique value when the variables are grounded.

Definition 12 *The map* $\mathsf{det}_1 : \mathcal{C}_= \to \mathcal{P}(V)$ *is defined by:* $\mathsf{det}_1(t = t') = (\mathsf{det}_1(t) \cup \mathsf{det}_1(t'))^{\downarrow}$, *where* $\mathsf{det}_1 : T_V \to \mathcal{M}(V)$ *is given by (where* $d \in \mathbb{R} \setminus \{0\}$*)*

Table 2
Example groundness abstractions for $X = \{w, x, y, z\}$

c_i	c'_i	$\alpha_X^{\mathsf{tvf}}(c_i)$	$\alpha_X^{\mathsf{det}_1}(c_i)$	$\alpha_X^{\mathsf{det}^*}(c_i)$	$\alpha_X(c_i)$
$w = x * (y + z)$	c_1	f_1	f_1	f_1	f_1
$w = x + w$	$x = 0$	f_2	f_2	f_2	f_2
$w = x/0$	$false$	f_3	f_3	f_3	f_3
$w = x + x/w$	c_4	f_4	f_4	f_5	f_5
$(w \leq x \wedge \text{abs}(x) \leq w)$	c_5	f_4	f_4	f_4	f_6

$$
\mathsf{det}_1(t) = \begin{cases}
\{t\} & \text{if } t \in V \\
\mathsf{det}_1(t') & \text{if } t = -t,\ t = d * t',\ t = t'/d \text{ or } t = d/t' \\
\mathsf{det}_1(t') \cup \mathsf{det}_1(t'') & \text{if } t = t' + t'' \\
\mathsf{det}_1(t') \cup \mathsf{det}_1(t') \cup \mathsf{det}_1(t'') \cup \mathsf{det}_1(t'') & \text{if } t = t' * t'',\ t = \text{pow}(t', t'') \\
 & \quad t = \min(t', t'') \text{ or } t = \max(t', t'') \\
\mathsf{det}_1(t') \cup \mathsf{det}_1(t'') \cup \mathsf{det}_1(t'') & \text{if } t = t'/t'' \\
\mathsf{det}_1(t') \cup \mathsf{det}_1(t') & \text{if } t = \text{abs}(t'),\ t = \cos(t') \text{ or } t = \sin(t') \\
\mathsf{det}_1(t') & \text{if } t = \arccos(t') \text{ or } t = \arcsin(t')
\end{cases}
$$

Proposition 13 $\mathsf{det}_1 \in \mathcal{D}et$.

Example 14 Consider $c = (x + v = x * y + z/w)$ and suppose $c \multimap \langle true, c\rangle$. Observe that $\mathsf{det}_1(c) = (\mathsf{det}_1(x+v) \cup \mathsf{det}_1(x*y+z/w))^{\downarrow} = (\mathsf{det}_1(x) \cup \mathsf{det}_1(v) \cup \mathsf{det}_1(x) \cup \mathsf{det}_1(x) \cup \mathsf{det}_1(y) \cup \mathsf{det}_1(y) \cup \mathsf{det}_1(z) \cup \mathsf{det}_1(w) \cup \mathsf{det}_1(w))^{\downarrow} = \{v, w, w, x, x, x, y, y, z\}^{\downarrow} = \{v, z\}$. Hence $\alpha_X^{\mathsf{det}_1}(c) = (v \leftarrow w \wedge x \wedge y \wedge z) \wedge (z \leftarrow v \wedge w \wedge x \wedge y)$.

Example 15 Table 2 details the abstractions for various constraints where $f_1 = w \leftarrow (x \wedge y \wedge z)$, $f_2 = x$, $f_3 = false$, $f_4 = true$ $f_5 = x \leftarrow w$, $f_6 = w \leftrightarrow x$. The abstraction algorithms are defined in terms of $\multimap$. It is assumed that $c_i \multimap \langle c', d'\rangle$ and $c'_i = c' \wedge d'$. In practice, $\multimap$ is evaluated by posting the constraint to the store and then retrieving it. The net effect is to evaluate ground terms and to group together like terms. For example, $2 * x = cos(\pi) * z + x + \max(2, 3) \multimap \langle x = -z + 3, true\rangle$. This $\multimap$ is used in this example. Note that $\alpha_X(c_i) \models \alpha_X^{\mathsf{det}^*}(c_i) \models \alpha_X^{\mathsf{det}_1}(c_i) \models \alpha_X^{\mathsf{tvf}}(c_i)$. The abstractions for c_1, c_2, c_3 all agree, illustrating that all methods give good accuracy. The abstractions of c_4 show that det_1 can still be strengthened. $\alpha_X(c_5)$ shows that, in general, systems of inequations need to be considered to compute the best abstraction. Note that if a weaker $\multimap$ were used, a stronger det could be defined to give abstractions of a similar strength. The three variable form method would not be so flexible.

5 Conclusion

This paper has described a simple algorithm for abstracting numeric constraints. This method does not introduce temporary variables, utilises available CLP($\mathcal{R}$) machinery, and is at least as precise as the three-variable method. Whilst other works have given precise definitions of abstraction, they have not addressed how to efficiently compute the map. This paper plugs this hole. The algorithm can be easily implemented and has been used with the analyser in [8]. Future work will look at the more general case of mixing Herbrand and linear constraints.

References

[1] T. Armstrong, K. Marriott, P. Schachte, and H. Søndergaard. Two Classes of Boolean Functions for Dependency Analysis. *Science of Computer Programming*, 31(1):3–45, 1998.

[2] N. Baker and H. Søndergaard. Definiteness Analysis for CLP($\mathcal{R}$). *Australian Computer Science Communications*, 15(1):321–332, 1993.

[3] M. Codish and B. Demoen. Analyzing Logic Programs Using "Prop"-ositional Logic Programs and a Magic Wand. *Journal of Logic Programming*, 25(3):249–274, 1995.

[4] M. García de la Banda. *Independence, Global Analysis, and Parallelism in Dynamically Scheduled Constraint Logic Programming*. PhD thesis, Universidad Politécnica de Madrid, 1994.

[5] M. García de la Banda, M. Hermenegildo, M. Bruynooghe, V. Dumortier, G. Janssens, and W. Simoens. Global Analysis of Constraint Logic Programs. *ACM Transactions on Programming Languages and Systems*, 18(5):564–614, 1996.

[6] A. Heaton, M. Abo-Zaed, M. Codish, and A. King. A Simple Polynomial Groundness Analysis for Logic Programs. *Journal of Logic Programming*, 2000. Forthcoming.

[7] N. Heintze, J. Jaffar, S. Michaylov, P. Stuckey, and R. Yap. *The CLP($\mathcal{R}$) Programmer's Manual Version 1.1*, 1991.

[8] J. M. Howe and A. King. Implementing Groundness Analysis with Definite Boolean Functions. In G. Smolka, editor, *European Symposium on Programming*, volume 1782 of *Lecture Notes in Computer Science*, pages 200–214. Springer-Verlag, 2000.

[9] A. King, J. Smaus, and P. Hill. Quotienting Share for Dependency Analysis. In S. D. Swierstra, editor, *European Symposium on Programming*, volume 1576 of *Lecture Notes in Computer Science*, pages 59–73. Springer-Verlag, 1999.