Simple and Effective Sign Consistency Using Interval Arithmetic

Federico Bergenti Stefania Monica

Dipartimento di Scienze Matematiche, Fisiche e Informatiche
Università degli Studi di Parma
{federico.bergenti,stefania.monica}@unipr.it

June 20th, 2019

Introduction and Motivation

The research is motivated by practical considerations

- *Polynomial constraints* are ubiquitous
- Polynomial constraints often involve variables that take values from *finite* subsets of the integers to model interesting combinatorial problems
- Typically, finite-domain constraint solvers do not treat polynomial constraints specifically

Example 1 (Grocery)

Four variables x_1, x_2, x_3, x_4 with integer domains $D_1 = D_2 = D_3 = D_4 = [0..711]$ such that

$$x_1 \leq x_2 \leq x_3 \leq x_4$$

$$x_1 \cdot x_2 \cdot x_3 \cdot x_4 = 711 \cdot 10^6$$

$$x_1 + x_2 + x_3 + x_4 = 711$$

The Constraint Language (I)

The signature Σ of the considered constraint language $\mathcal{L}_P$ is

$$\Sigma = \langle \mathcal{V}, \mathcal{F}, \mathcal{P} \rangle$$

where

- $\mathcal{V}$ is a denumerable set of variable symbols
- $\mathcal{F} = O \cup Z$ is the set of constant symbols and function symbols with $O = \{+, *\}$ and $Z = \{0, 1, -1, 2, -2, \ldots\}$
- $\mathcal{P} = \{=, \neq, <, \leq, >, \geq\}$ is the finite set of constraint predicate symbols

As usual

- A *primitive constraint* is any atomic predicate built using the symbols from signature Σ
- A (*non-primitive*) *constraint* is a conjunction of primitive constraints

The Constraint Language (II)

Signature Σ can express primitive constraints that are normally interpreted in terms of equalities, inequalities, and disequalities among (multivariate) polynomials with integer coefficients

If the chosen interpretation restricts variables to take values from finite subsets of the integers, constraints are called *polynomial constraints over finite domains*

Polynomial constraints over finite domains are better studied using (multivariate) polynomial functions because

- The study of the satisfiability of a constraint can be reduced to the study of the sign of a polynomial function
- A specific type of local consistency called *sign consistency* can be introduced

Multi-Indices

Multi-indices are tuple of natural numbers and they are normally introduced to study polynomial functions

Given $n \in \mathbb{N}_+$ and two multi-indices $I \in \mathbb{N}^n$ and $J \in \mathbb{N}^n$, with $I = (i_k)_{k=1}^n$ and $J = (j_k)_{k=1}^n$,

$$I + J = (i_k + j_k)_{k=1}^n$$

$$\sum_{I \leq J} (\cdot) = \sum_{i_1=0}^{j_1} \sum_{i_2=0}^{j_2} \cdots \sum_{i_n=0}^{j_n} (\cdot)$$

Given $\mathbf{x} \in \mathbb{R}^n$ with $\mathbf{x} = (x_k)_{k=1}^n$, the following abbreviation is used

$$\mathbf{x}^I = \prod_{j=0}^{n} x_j^{i_j}$$

Polynomial Functions

A *polynomial function* $p : \mathbb{R}^n \mapsto \mathbb{R}$ of $n \in \mathbb{N}_+$ (real) variables is such that for all $\mathbf{x} \in \mathbb{R}^n$

$$p(\mathbf{x}) = \sum_{I \leq L} a_I \mathbf{x}^I$$

where $L \in \mathbb{N}^n$ is the *multi-degree* of p and $\{a_I\}_{I \leq L} \subset \mathbb{R}$ is the set of its (real) coefficients

The set of polynomial functions of n real variables, with real coefficients and multi-degree less than or equal to L is

$$\Pi_L = \text{span}_{\mathbb{R}}\{P_I\}_{I \leq L} \qquad P_I : \mathbb{R}^n \mapsto \mathbb{R} \quad P_I(\mathbf{x}) = \mathbf{x}^I$$

Similarly, the set of polynomial functions of n integer variables, with integer coefficients and multi-degree less than or equal to L is

$$\tilde{\Pi}_L = \text{span}_{\mathbb{Z}}\{\tilde{P}_I\}_{I \leq L} \qquad \tilde{P}_I : \mathbb{Z}^n \mapsto \mathbb{Z} \quad \tilde{P}_I(\mathbf{x}) = \mathbf{x}^I$$

Polynomial Constraints over Finite Domains (I)

The following definition precisely introduces polynomial constraints over finite domains

Definition 2 (Polynomial constraint over finite domains)

An $n-$ary constraint C whose $n \in \mathbb{N}_+$ variables take values from domains $(D_i)_{i=1}^n$ is a polynomial constraint over finite domains if and only if all domains are finite subsets of $\mathbb{Z}$ and

$$C = \{\mathbf{x} \in \prod_{i=1}^{n} D_i : p(\mathbf{x}) \geq 0\} \quad \text{or} \quad C = \{\mathbf{x} \in \prod_{i=1}^{n} D_i : p(\mathbf{x}) \neq 0\}$$

for a proper polynomial function $p \in \tilde{\Pi}_L$ of n integer variables, with integer coefficients, and with multi-degree less than or equal to multi-index $L \in \mathbb{N}^n$

Polynomial Constraints over Finite Domains (II)

The proposed definition is not restrictive because the following lemma is easily proved

Lemma 3

Given two polynomial functions $p \in \tilde{\Pi}_L$ and $q \in \tilde{\Pi}_L$, the following co-implications hold for all $\mathbf{x} \in \mathbb{Z}^n$

$$
\begin{aligned}
p(\mathbf{x}) \leq q(\mathbf{x}) &\iff q(\mathbf{x}) - p(\mathbf{x}) \geq 0 \\
p(\mathbf{x}) < q(\mathbf{x}) &\iff q(\mathbf{x}) - p(\mathbf{x}) - 1 \geq 0 \\
p(\mathbf{x}) > q(\mathbf{x}) &\iff p(\mathbf{x}) - q(\mathbf{x}) - 1 \geq 0 \\
p(\mathbf{x}) \neq q(\mathbf{x}) &\iff p(\mathbf{x}) - q(\mathbf{x}) \neq 0 \\
p(\mathbf{x}) = q(\mathbf{x}) &\iff p(\mathbf{x}) - q(\mathbf{x}) \geq 0 \wedge q(\mathbf{x}) - p(\mathbf{x}) \geq 0
\end{aligned}
$$

With an abuse of notation, the statements $p(\mathbf{x}) \geq 0$ and $p(\mathbf{x}) \neq 0$ are used to refer to the corresponding constraints

Local Consistency for Polynomial Constraints over Boxes

The enforcement of local consistency is one of the most effective means to solve constraint satisfaction problems

Sign consistency is a specific form of local consistency proposed to reason on polynomial constraints over finite domains

- It is based on the possibility of reducing polynomial constraints over finite domains to $p(\mathbf{x}) \geq 0$ and $p(\mathbf{x}) \neq 0$
- It reduces the study of satisfiability over finite domains to the study of the sign of polynomial functions over integer boxes (hence, its name)
- It is parameterized in terms of a *bounding function* to adapt to the characteristics of studied constraints and to compromise its strength with the computational cost needed to enforce it

Integer Boxes (I)

An *integer interval* from $\underline{c} \in \mathbb{Z}$ to $\overline{c} \in \mathbb{Z}$ is denoted as

$$[\underline{c}..\overline{c}] = \{x \in \mathbb{Z} : \underline{c} \leq x \leq \overline{c}\},$$

and it equals the empty set if and only if $\underline{c} > \overline{c}$

A *singleton integer interval* that contains only $c \in \mathbb{Z}$ is denoted as $[c] = [c..c]$

Given $n \in \mathbb{N}_+$, an *integer box* $D \subset \mathbb{Z}^n$ from $\underline{\mathbf{d}} = (\underline{d}_k)_{k=1}^n \in \mathbb{Z}^n$ to $\overline{\mathbf{d}} = (\overline{d}_k)_{k=1}^n \in \mathbb{Z}^n$ is denoted as

$$D = [\underline{\mathbf{d}}..\overline{\mathbf{d}}] = [\underline{d}_1..\overline{d}_1] \times [\underline{d}_2..\overline{d}_2] \times \cdots \times [\underline{d}_n..\overline{d}_n]$$

and it equals the empty set if and only if $\underline{d}_i > \overline{d}_i$ for some $1 \leq i \leq n$

Integer Boxes (II)

The notation $D_i = [\underline{d}_i..\overline{d}_i] \subset \mathbb{Z}$ with $1 \leq i \leq n$ is used to refer to the integer intervals that compose the nonempty box D

The notation $D_{i \to T}$ is used to refer to the box obtained by replacing the $i-$th integer interval that composes the nonempty box D with the nonempty integer interval $T \subset \mathbb{Z}$

The *bounding box* $\square A$ of a nonempty finite $A \subset \mathbb{Z}^n$ is the inclusion-minimal integer box such that $A \subseteq \square A$

Bounding Functions over Boxes

Definition 4 (Bounding function)

A bounding function β is a computable function that, given a nonempty integer box $B \subset \mathbb{Z}^n$ and a polynomial function $p \in \tilde{\Pi}_L$ of $n \in \mathbb{N}_+$ integer variables, with integer coefficients, and with multi-degree less than or equal to multi-index $L \in \mathbb{N}^n$, computes $(\underline{p}, \overline{p}) \in \mathbb{R}^2$ such that the following conditions jointly hold

$$\underline{p} \leq \min_{\mathbf{x} \in B} p(\mathbf{x}) \qquad \max_{\mathbf{x} \in B} p(\mathbf{x}) \leq \overline{p}$$

Bounding functions are used to extract relevant information from a given constraint when its variables are restricted to take values from a given box

They are used to adapt sign consistency to the characteristics of studied problems

Sign Consistency (I)

Definition 5 (Sign consistency)

Given a bounding function β and a polynomial constraint over finite domains C whose $n \in \mathbb{N}_+$ variables take values from nonempty domains $(D_i)_{i=1}^n$, with $B = \square \prod_{i=1}^n D_i$, a value $v \in D_i$ with $1 \leq i \leq n$ is sign consistent for β with C if and only if

- The constraint is $p(\mathbf{x}) \geq 0$, $\beta(p, B_{i \to [v..v]}) = (\underline{p}, \overline{p})$, and $\overline{p} \geq 0$
- The constraint is $p(\mathbf{x}) \neq 0$, $\beta(p, B_{i \to [v..v]}) = (\underline{p}, \overline{p})$, and $\underline{p} \neq 0$ or $\overline{p} \neq 0$

A domain D_i with $1 \leq i \leq n$ is sign consistent for β with C if and only if all its values are sign consistent for β with C

Constraint C is sign consistent for β if and only if all its domains are sign consistent for β with C

Sign Consistency (II)

Bounding function β is used to compromise the strength of sign consistency with the computational cost needed to enforce it

Three bounding functions have already been studied

- Bounding function β_H relies on the exhaustive enumeration of the elements of the given box
- Bounding function β_B uses classic results on the Bernstein form of polynomials
- Bounding function β_R uses the values of the considered polynomial function at the corners of the given box to compute lower and upper bounds

The relationships between sign consistency and other types of local consistency (hyper-arc consistency, bounds consistency) are studied in an upcoming paper

A Bounding Function Based on Interval Arithmetic (I)

Interval arithmetic can be used to express computations whose arguments and results are integer intervals

Given two nonempty integer intervals $A = [\underline{a}..\overline{a}] \subset \mathbb{Z}$ and $C = [\underline{c}..\overline{c}] \subset \mathbb{Z}$, they can be added

$$A + C = [\underline{a} + \underline{c}, \overline{a} + \overline{c}]$$

and they can be multiplied

$$A \cdot C = [\min\{\underline{a}\,\underline{c}, \underline{a}\,\overline{c}, \overline{a}\,\underline{c}, \overline{a}\,\overline{c}\}, \max\{\underline{a}\,\underline{c}, \underline{a}\,\overline{c}, \overline{a}\,\underline{c}, \overline{a}\,\overline{c}\}]$$

The product among integer intervals can be used to define the $m-$th power of a nonempty integer interval $A \subset \mathbb{Z}$

A Bounding Function Based on Interval Arithmetic (II)

Given $n \in \mathbb{N}_+$, a nonempty integer box $B \subset \mathbb{Z}^n$, and a multi-index $I \in \mathbb{N}^n$, the following abbreviation is adopted

$$B^I = \prod_{j=1}^{n} B_j^{i_j}$$

A polynomial function $p \in \tilde{\Pi}_L$ of $n \in \mathbb{N}_+$ integer variables with integer coefficients and multi-degree less than or equal to multi-index $L \in \mathbb{N}^n$ can be extended to work on integer boxes

$$p(B) = \sum_{I \leq L} [a_I] \, B^I$$

where the notation for singleton intervals is used to treat the coefficients of the polynomial function as integer intervals

A Bounding Function Based on Interval Arithmetic (III)

Definition 6 (Bounding function β_I)

Given a nonempty integer box $B \subset \mathbb{Z}^n$ and a polynomial function $p \in \tilde{\Pi}_L$ of $n \in \mathbb{N}_+$ integer variables, with integer coefficients, and with multi-degree less than or equal to multi-degree $L \in \mathbb{N}^n$, the bounding function β_I is

$$\beta_I(p, B) = (\underline{p}, \overline{p})$$

where $p(B) = [\underline{p}..\overline{p}]$

It is easy to prove that the proposed definition of β_I is coherent with the characteristics of bounding functions

Preliminary Experiments (I)

PolyFD is a C++ library designed to support reasoning on polynomial constraints over finite domains

- It is still at an early-prototype development stage
- Its core algorithms are implemented in ANSI C99
- It offers a C++ interface and a lower-level C interface
- It can be compiled to support arbitrary-precision integer arithmetic using GMP

PolyFD has been recently enhanced with an implementation of an algorithm to enforce sign consistency

- It is based on a variation of AC-3 that enforces sign consistency instead of arc consistency
- It can host both β_I and β_R

Preliminary Experiments (II)

	PolyFD [ms]	Gecode [ms]
Corner	0.647	0.651
Dinner	0.640	0.544
Donald	1.112	1.152
Grocery	32.635	168.002
Safe	1.654	0.863

The effectiveness of sign-consistency enforcement with β_I was preliminary assessed using few problems that ship with Gecode

- Both PolyFD and Gecode were configured to use the same heuristics to select variables and values from domains
- Gecode was configured not to use specialized global constraints (*distinct*, *linear*)

Conclusion

Sign consistency was introduced as a type of local consistency specifically designed for polynomial constraints over finite domains

- It does not require to break constraints into simpler constraints by introducing fresh variables
- Its strength and computational cost can be adapted to the studied problems using bounding functions

Bounding function β_I was introduced to reduce the computational cost typically associated with other bounding functions

- It can be computed (much) faster than β_R and β_B for the considered problems
- It can take advantage of the sparsity patterns of studied polynomials

The effectiveness of sign-consistency enforcement with β_I is still under study and discussed experimental results are still preliminary