On Intervals and Bounds in Bit-vector Arithmetic

Mikoláš Janota[1] and Christoph M. Wintersteiger[1]

Microsoft Research
{mikjan, cwinter}@microsoft.com

1 Introduction

Bit-vector arithmetic operations rely on modular arithmetic semantics, which often complicates algebraic manipulations. One such example are inequalities. Satisfiability of a system of modular inequalities is NP-complete [1]. In this memo we investigate a simple type of inequalities: only one variable is permitted and there are no multiplications. Satisfiability is solved in polynomial time by translating the problem into conjunctions of positive and negative intervals. This lets us decide systems of such inequalities but also identify uniqueness of solution and remove redundancies. This approach can be used to preprocess bit-vector problems and might also be useful in connection with other approaches, such as abstract interpretation [3,2].

2 Problem Definition

For the purpose of the memo we assume a fixed bit-with $w \in \mathbb{N}^+$. A bit vector variable is seen as an integer variable with possible values $0..(2^w - 1)$. Addition $a +^w b$ is defined as $(a + b) \mod 2^w$. Two types of comparisons are defined: *unsigned* $a \leq^{\mathrm{u}} b$ and *signed* $a \leq^{\mathrm{s}} b$. The unsigned is standard integer comparison, i.e., $a \leq^{\mathrm{u}} b \Leftrightarrow a \leq b$. The signed uses the two's complement representation for signed numbers. Hence, we have $a \leq^{\mathrm{s}} b \Leftrightarrow s(a) \leq s(b)$, where $s(a) \triangleq a < 2^{w-1} \; ? \; a \; : \; a - 2^w$.

Consider a conjunction $\bigwedge_{1..m} \neg(a_i \odot b_i) \wedge \bigwedge_{m+1..n}(a_i \odot b_i)$, where $a_i \odot b_i$ is one of the following forms with x a bit-vector variable and c_1, c_2 constants.

1. $c_1 +^w x \leq^{\mathrm{u}} c_2 +^w x$
2. $c_1 \leq^{\mathrm{u}} c_2 +^w x$
3. $c_1 +^w x \leq^{\mathrm{u}} c_2$
4. $x \leq^{\mathrm{s}} c_1$
5. $c_1 \leq^{\mathrm{s}} x$

Problem: Decide whether the input has a solution and if it has, determine if it is unique.

3 Solution

We show that every constraint of the form above can be represented as a positive or negative interval constraint. We write $[a\,;b]$ and $\backsim[a\,;b]$ for the positive and negative interval ranging from a to b inclusively, respectively. Hence, $[a\,;b]$ for $0 \leq a, b < 2^w$ denotes the set of values from a to b and $\backsim[a\,;b]$ the values v with $(0 \leq v < a) \vee (b < v < 2^w)$. Observe that $[a\,;b] \cup \backsim[a\,;b] = [0\,;2^w - 1]$. For $a > b$, it holds that $[a\,;b] = \emptyset$ and $\backsim[a\,;b] = [0\,;2^w - 1]$.

Table 1. Translation of constraints.

Expression	Condition	Interval
$c_1 +^w x \leq^{\mathrm{u}} c_2 +^w x$	$c_1 \leq c_2$	$\backsim[-c_2\,;\,-c_1-1]$
$c_1 +^w x \leq^{\mathrm{u}} c_2 +^w x$	$c_1 > c_2$	$[-c_1\,;\,-c_2-1]$
$c_1 \leq^{\mathrm{u}} c_2 +^w x$	$c_1 < c_2$	$\backsim[-c_2\,;c_1-c_2-1]$
$c_1 \leq^{\mathrm{u}} c_2 +^w x$	$c_1 \geq c_2$	$[c_1-c_2\,;\,-c2-1]$
$c_1 +^w x \leq^{\mathrm{u}} c_2$	$c_1 \leq c_2$	$\backsim[c_2-c_1+1\,;\,-c_1-1]$
$c_1 +^w x \leq^{\mathrm{u}} c_2$	$c_1 > c_2$	$[-c_1\,;\,-c_1+c_2]$
$x \leq^{\mathrm{s}} c_1$	$c_1 < 2^{w-1}$	$\backsim[c_1+1\,;2^{w-1}-1]$
$x \leq^{\mathrm{s}} c_1$	$c_1 \geq 2^{w-1}$	$[2^{w-1}\,;c_1]$
$c_1 \leq^{\mathrm{s}} x$	$c_1 < 2^{w-1}$	$[c_1\,;2^{w-1}-1]$
$c_1 \leq^{\mathrm{s}} x$	$c_1 \geq 2^{w-1}$	$\backsim[2^{w-1}\,;c_1-1]$

Table 1 presents the conversion from the considered expressions to intervals. The table uses the convention that a negative number $-c$ is automatically translated to the modular representative $2^w - c$.

The equivalence of the conversions is shown by case-splitting on whether the expressions $x +^w c_i$ overflow or not. So for instance $c_1 +^w x \leq^{\mathrm{u}} c_2$ with $c_1 \leq c_2$ holds in two scenarios: (1) $c_1 +^w x$ does not overflow and then it must be that $x \leq c_2 - c_1$; (2) $c_1 +^w x$ overflows and $c_1 +^w x \leq^{\mathrm{u}} c_2$ holds automatically because $(c_1 + x - 2^w) \leq c_1 \leq c_2$. The complement of the two scenarios is therefore $[c_2 - c_1 + 1\,;\,-c_1 - 1]$.

Satisfiability as well as unique value property can be decided in polynomial time. Algorithm 1 present an algorithm that determines the upper and lower bound of the space determined by a set of intervals I. All the positive intervals are intersected obtaining thus lower bounds l and h, respectively. Negative intervals are processed in the increasing order of their lower bound, where the variable p is used as a pointer to the position where we may potentially find a point not covered by a negative interval. If the set of intervals is unsatisfiable then the algorithm returns $[l'\,;h']$ with $l' > h'$, if there's a single solution then $l' = h'$, and $l' < h'$ means there are multiple solutions.[1]

[1] The algorithm is a generalization of a standard solution to "The Interval Point Cover Problem".

Algorithm 1: Compute bounds

input : Set of intervals I

$P \leftarrow \{[a\,;b] \mid [a\,;b] \in I\}$ // get positive intervals

$l \leftarrow P = \emptyset \,?\, 0 : \texttt{min}\,\{a \mid [a\,;b] \in P\}$ // get positive lower bound

$h \leftarrow P = \emptyset \,?\, 2^w - 1 : \texttt{max}\,\{b \mid [a\,;b] \in P\}$ // get positive upper bound

$N \leftarrow \{\backsim[a\,;b] \mid \backsim[a\,;b] \in I\}$ // get negative intervals

$N \leftarrow$ sort N by first element

$p, l', h' \leftarrow l, l, l-1$

for $\backsim[a\,;b] \in N \cup \backsim[2^w\,;2^w]$ **do**

  if $p > h$ **then break** // space exhausted

  if $b < p$ **then continue** // redundant interval

  if $p < a$ **then** // satisfiable portion

    if $h' > l'$ **then** $l' \leftarrow p$ // first satisfiable point

    $h' \leftarrow a - 1$ // update upper bound

  $p \leftarrow b + 1$ // move onto next portion

return $[l'\,;h']$

3.1 Implementation and Redundancies

Algorithm 1 can be implemented in a straightforward fashion. Positive intervals do not need to be explicitly stored. Rather, we maintain a lower and upper bound L and U, respectively. These are updated with every new positive interval. These bounds let us also simplify negative intervals. An interval $\backsim[a\,;b]$ with $a \leq L$ is simplified to $[b+1\,;2^{w-1}]$. Analogously, $\backsim[a\,;b]$ with $b \geq u$ is simplified to $[0\,;a-1]$.

Maintaining the upper and lower bounds lets us quickly detect if a new interval is redundant. A positive interval $[a\,;b]$ is redundant if $L \leq a \wedge b \leq U$ and a negative interval $\backsim[a\,;b]$ is redundant if $a > U \vee b < L$. Note that such detection of redundancies depends on the order of processing the intervals.

4 Examples

To measure the effect of the individual simplifications we construct several examples. (1) For $c_1 < c_2$, the conjunct $x +^w c_1 \leq x +^w c_2 \wedge x +^w c_2 \leq x +^w c_1$ is unsatisfiable. (2) For $x +^w c_1 \leq x +^w c_2 \wedge x +^w d_1 \leq x +^w d_2$ the second conjunct is redundant if $-c_2 \leq -d_2 \leq -d_1 - 1 \leq -c_1 - 1$. (3) For $x +^w c_1 \leq x +^w c_2 \wedge x +^w d_1 \leq x +^w d_2 \wedge x +^w a \leq x +^w b$ we obtain $x = -c_1$ if $-c_2 \leq -a \leq -c_1 - 1 \leq -b - 1 \leq -d_1 - 1$ and $-d_2 = -c_1 + 1$.

We generated 100 random instances for 32-bit numbers for each example and applied the standard bit-blasting technique in Z3 on them. Table 2 overviews the number of conflicts in the SAT solver.

Examples (1) and (2) are completely solved by Algorithm 1. Hence, implementing the algorithm in a preprocessor saves us the number of conflicts presented. For case (2) we compare the reduced and original instance, i.e., the redundant version contains two conjuncts whereas the reduced version only one.

Table 2. Conflict count.

Example	Avg.	Med	Min	Max
(1) unsat	26	36	3	100
(2) redundant	31	25	7	89
(2) reduced	29	37	3	92
(3) unique	21	40	0	123

The comparison shows that the biggest saving are in the case of unsatisfiable constraints and constraints that lead to unique solution. In the case of redundant intervals, however, the results do not indicate improvement.

5 Conclusion and Future Work

This memo shows how a system of inequalities over one variable and no multiplication can be represented as a set of positive and negative intervals. Naturally, such intervals are closed under negation, admit a polynomial satisfiability check and check for solution uniqueness. Preliminary evaluation suggests that such analysis enables significant speedups. In the future we plan to integrate this analysis into Z3 and investigate how could improve bit-vector solving by approximating solutions as intervals.

References

1. Bjørner, N., Blass, A., Gurevich, Y., Musuvathi, M.: Modular difference logic is hard. CoRR abs/0811.0987 (2008), http://arxiv.org/abs/0811.0987
2. Elder, M., Lim, J., Sharma, T., Andersen, T., Reps, T.: Abstract domains of affine relations. ACM Trans. Program. Lang. Syst. 36(4), 11:1–11:73 (Oct 2014), http://doi.acm.org/10.1145/2651361
3. Sharma, T., Thakur, A., Reps, T.: An abstract domain for bit-vector inequalities. Tech. Rep. TR1789, University of Wisconsin-Madison (2013), http://digital.library.wisc.edu/1793/65366