ON INTERVALS AND BOUNDS IN BIT-VECTOR ARITHMETIC

Mikoláš Janota and Christoph M. Wintersteiger

Microsoft Research, Cambridge, UK
WHAT ARE BIT VECTORS?

- numbers as in computer (roughly)
WHAT ARE BIT VECTORS?

- numbers as in computer (roughly)
- fixed bit-width
WHAT ARE BIT VECTORS?

- numbers as in computer (roughly)
- fixed bit-width
- +, × wraps around
WHAT ARE BIT VECTORS?

- numbers as in computer (roughly)
- fixed bit-width
- +, \( \times \) wraps around
- negative numbers via 2’s complement
WHAT ARE BIT VECTORS?

- numbers as in computer (roughly)
- fixed bit-width
- \(+, \times\) wraps around
- negative numbers via 2’s complement
- Example: \((x_8 +_8 1_8) \leq^S_8 (y_8 \times_8 3_8)\)
WHAT ARE BIT VECTORS?

- numbers as in computer (roughly)
- fixed bit-width
- $+, \times$ wraps around
- negative numbers via 2’s complement
- **Example:** $(x_8 +_8 1_8) \leq^s_8 (y_8 \times_8 3_8)$
- $x_8 = 0x7f, y_8 = 0x80$
HOW DO WE SOLVE BIT-VECTORS?

- **Bit-blasting** — convert everything to propositional form (SAT).

Exponential in bit-width and losing "domain" knowledge.

It is important to apply **preprocessing** before sending to SAT.

Example:
\[(x^m + 0^m) = x^m\]

Example:
\[(x^m + 4^m) = x^{[m + 0]}\]

Janota, Wintersteiger
HOW DO WE SOLVE BIT-VECTORS?

- **Bit-blasting** — convert everything to propositional form (SAT).
- Exponential in bit-width and losing “domain” knowledge.
HOW DO WE SOLVE BIT-VECTORS?

- **Bit-blasting** — convert everything to propositional form (SAT).
- Exponential in bit-width and losing “domain” knowledge.
- It is important to apply **preprocessing** before sending to SAT.
HOW DO WE SOLVE BIT-VECTORS?

- **Bit-blasting** — convert everything to propositional form (SAT).
- Exponential in bit-width and losing “domain” knowledge.
- It is important to apply *preprocessing* before sending to SAT.
- **Example**: \((x_m + 0_m) = x_m\)
HOW DO WE SOLVE BIT-VECTORS?

• **Bit-blasting** — convert everything to propositional form (SAT).
• Exponential in bit-width and losing “domain” knowledge.
• It is important to apply **preprocessing** before sending to SAT.
• Example: \((x_m + 0_m) = x_m\)
• Example: \((x_m \times 4_m) = x[m - 3 : 0] + 0_2\)
PROBLEM

- inequalities with multiple variables and addition are NP-complete
• inequalities with multiple variables and addition are NP-complete
• conjunction $\land \neg C_i \land \land C_i$, where $C_i$ is one of the following with $x$ a bit-vector variable and $c_1, c_2$ constants.
• inequalities with multiple variables and addition are NP-complete
• conjunction $\land \neg C_i \land \land C_i$, where $C_i$ is one of the following with $x$ a bit-vector variable and $c_1, c_2$ constants.

1. $(c_1 + w x) \leq u (c_2 + w x)$
• inequalities with multiple variables and addition are NP-complete
• conjunction \( \land \neg C_i \land \land C_i \), where \( C_i \) is one of the following with \( x \) a bit-vector variable and \( c_1, c_2 \) constants.
  1. \((c_1 + w \ x) \leq^u (c_2 + w \ x)\)
  2. \(c_1 \leq^u (c_2 + w \ x)\)
• inequalities with multiple variables and addition are NP-complete
• conjunction $\wedge \neg C_i \wedge \bigwedge C_i$, where $C_i$ is one of the following with $x$ a bit-vector variable and $c_1, c_2$ constants.
  1. $(c_1 + w \cdot x) \leq^u (c_2 + w \cdot x)$
  2. $c_1 \leq^u (c_2 + w \cdot x)$
  3. $(c_1 + w \cdot x) \leq^u c_2$
• inequalities with multiple variables and addition are NP-complete

• conjunction $\land \neg C_i \land \land C_i$, where $C_i$ is one of the following with $x$ a bit-vector variable and $c_1, c_2$ constants.

  1. $(c_1 + w x) \leq^u (c_2 + w x)$
  2. $c_1 \leq^u (c_2 + w x)$
  3. $(c_1 + w x) \leq^u c_2$
  4. $x \leq^s c_1$
• inequalities with multiple variables and addition are NP-complete

• conjunction $\bigwedge \neg C_i \land \bigwedge C_i$, where $C_i$ is one of the following with $x$ a bit-vector variable and $c_1, c_2$ constants.

1. $(c_1 +_w x) \leq^u (c_2 +_w x)$
2. $c_1 \leq^u (c_2 +_w x)$
3. $(c_1 +_w x) \leq^u c_2$
4. $x \leq^s c_1$
5. $c_1 \leq^s x$
\[ (0 <_8^s x) \land (200 <^u_8 x) \ldots \text{UNSAT} \]
• \((0 <^{s}_8 x) \land (200 <^{u}_8 x)\) ... UNSAT

\[\begin{array}{c|c|c|c}
0 & 128 & 200 & 0/256 \\
\hline
0 <^{s}_8 x & & & 200 <^{u}_8 x \\
\end{array}\]

• \((x + 100 <^{u}_8 x + 200)\)

\[\begin{array}{c|c|c|c}
0 & 56 & 156 & 0/256 \\
\hline
true & false & true & \\
\end{array}\]
INEQUALITIES

• \((0 \leq^s_8 x) \land (200 \leq^u_8 x) \ldots \text{UNSAT}\)

\[
\begin{array}{c|c|c|c}
0 & 128 & 200 & 0/256 \\
0 \leq^s_8 x & & 200 \leq^u_8 x \\
\end{array}
\]

• \((x + 100 \leq^u_8 x + 200)\)

\[
\begin{array}{c|c|c|c}
0 & 56 & 156 & 0/256 \\
\text{true} & \text{false} & \text{true} & \\
\end{array}
\]

• \((x + 200 \leq^u_8 x + 100)\)

\[
\begin{array}{c|c|c|c}
0 & 56 & 156 & 0/256 \\
\text{false} & \text{true} & \text{false} & \\
\end{array}
\]
<table>
<thead>
<tr>
<th>Expression</th>
<th>Condition</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1 + w \times \leq u c_2 + w \times$</td>
<td>$c_1 \leq c_2$</td>
<td>$\sim [-c_2; -c_1 - 1]$</td>
</tr>
<tr>
<td>$c_1 + w \times \leq u c_2 + w \times$</td>
<td>$c_1 &gt; c_2$</td>
<td>$[-c_1; -c_2 - 1]$</td>
</tr>
<tr>
<td>$c_1 \leq u c_2 + w \times$</td>
<td>$c_1 &lt; c_2$</td>
<td>$\sim [-c_2; c_1 - c_2 - 1]$</td>
</tr>
<tr>
<td>$c_1 \leq u c_2 + w \times$</td>
<td>$c_1 \geq c_2$</td>
<td>$[-c_1 - c_2; -c_2 - 1]$</td>
</tr>
<tr>
<td>$c_1 + w \times \leq u c_2$</td>
<td>$c_1 \leq c_2$</td>
<td>$\sim [c_2 - c_1 + 1; -c_1 - 1]$</td>
</tr>
<tr>
<td>$c_1 + w \times \leq u c_2$</td>
<td>$c_1 &gt; c_2$</td>
<td>$[-c_1; -c_1 + c_2]$</td>
</tr>
<tr>
<td>$x \leq s c_1$</td>
<td>$c_1 &lt; 2^{w-1}$</td>
<td>$\sim [c_1 + 1; 2^{w-1} - 1]$</td>
</tr>
<tr>
<td>$x \leq s c_1$</td>
<td>$c_1 \geq 2^{w-1}$</td>
<td>$[2^{w-1}; c_1]$</td>
</tr>
<tr>
<td>$c_1 \leq s x$</td>
<td>$c_1 &lt; 2^{w-1}$</td>
<td>$[c_1; 2^{w-1} - 1]$</td>
</tr>
<tr>
<td>$c_1 \leq s x$</td>
<td>$c_1 \geq 2^{w-1}$</td>
<td>$\sim [2^{w-1}; c_1 - 1]$</td>
</tr>
</tbody>
</table>
computing the envelope

1 \( P \leftarrow \{ [a; b] | [a; b] \in \mathcal{I} \}; \quad // \text{positive intervals} \\
2 l \leftarrow P = \emptyset ? 0 : \min \{ a | [a; b] \in P \}; \quad // \text{lower bound} \\
3 h \leftarrow P = \emptyset ? 2^w - 1 : \max \{ b | [a; b] \in P \}; \quad // \text{upper bound} \\
4 N \leftarrow \{ \neg [a; b] | \neg [a; b] \in \mathcal{I} \}; \quad // \text{negative intervals} \\
5 N' \leftarrow \text{sort } N \text{ by first element}; \\
6 p, l', h' \leftarrow l, l, l - 1; \\
7 \text{for } \neg [a; b] \in N \cup \neg [2^w; 2^w] \text{ do} \\
8 \quad \text{if } p > h \text{ then break; } \quad // \text{space exhausted} \\
9 \quad \text{if } b < p \text{ then continue; } \quad // \text{redundant interval} \\
10 \quad \text{if } p < a \text{ then} \\
11 \quad \quad \text{if } h' > l' \text{ then } l' \leftarrow p; \quad // \text{first satisfiable point} \\
12 \quad \quad h' \leftarrow a - 1; \quad // \text{update upper bound} \\
13 \quad \quad p \leftarrow b + 1; \quad // \text{move onto next portion} \\
14 \text{return } [l'; h']
Table 1: Conflict count.

<table>
<thead>
<tr>
<th>Example</th>
<th>Avg.</th>
<th>Med</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) unsat</td>
<td>26</td>
<td>36</td>
<td>3</td>
<td>100</td>
</tr>
<tr>
<td>(2) redundant</td>
<td>31</td>
<td>25</td>
<td>7</td>
<td>89</td>
</tr>
<tr>
<td>(2) reduced</td>
<td>29</td>
<td>37</td>
<td>3</td>
<td>92</td>
</tr>
<tr>
<td>(3) unique</td>
<td>21</td>
<td>40</td>
<td>0</td>
<td>123</td>
</tr>
</tbody>
</table>
• Single-variable inequalities appear in tests for overflows.
REMARKS

• Single-variable inequalities appear in tests for overflows.
• As simple preprocessing do not seem to be very effective.
• Single-variable inequalities appear in tests for overflows.
• As simple preprocessing do not seem to be very effective.
• Context sensitive? During search?
Thank You for Your Attention!

Questions?