TBA – To Beat Arithmetic

Harald Zankl

Computational Logic
Institute of Computer Science
University of Innsbruck

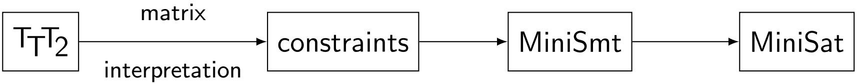
July 4, 2012

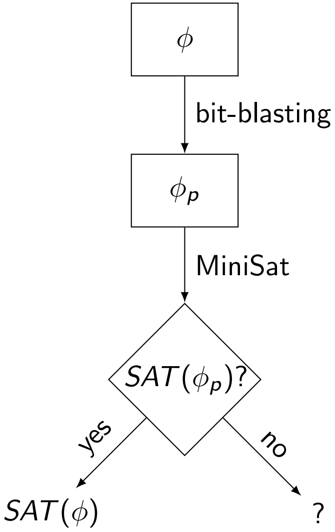
Example (Non-Linear Arithmetic (NIA))

$\exists a \; \exists b \; ((\mathsf{b} + \mathsf{b} > 3 \times \mathsf{b}) \vee (\mathsf{b} > -\mathsf{a})) \wedge (\mathsf{a} \times \mathsf{a} = 2)$ $\quad \mathsf{a} \mapsto \sqrt{2}, \mathsf{b} \mapsto -1$

MiniSmt

- SMT-solver for (ir)rational quantifier-free non-linear arithmetic
- domains $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, "$\mathbb{R}$" $(a + b\sqrt{2})$
- developed at Computational Logic group, Innsbruck





$UNSAT(\phi_p)$

- $UNSAT(\phi)$ or
- too few bits

Example

$$\phi \ := \ a + b = 3$$

$$\rightsquigarrow [a_1, a_0] + [b_1, b_0] = [\top, \top] \text{ ... guess bit-length 2 for } a, b$$

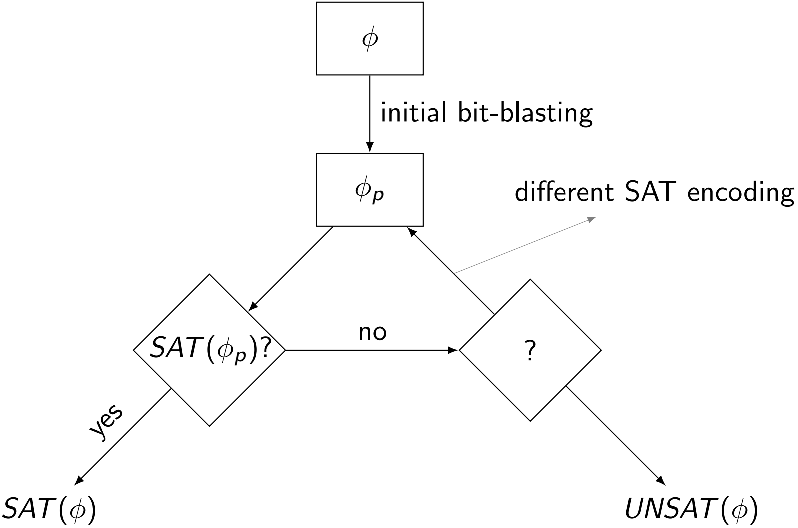
$$\rightsquigarrow [a_1, a_0] + [b_1, b_0] = [s_2, s_1, s_0] \wedge [s_2, s_1, s_0] = [\bot, \top, \top]$$

$$\begin{aligned}
\phi_p := \ & (s_0 \leftrightarrow a_0 \oplus b_0) \wedge (c_1 \leftrightarrow a_0 \wedge b_0) \wedge \\
& (s_1 \leftrightarrow a_1 \oplus b_1 \oplus c_1) \wedge (c_2 \leftrightarrow (a_1 \wedge b_1) \vee (a_1 \wedge c_1) \vee (b_1 \wedge c_1)) \wedge \\
& (s_2 \leftrightarrow c_2) \wedge \\
& (s_0 \leftrightarrow \top) \wedge (s_1 \leftrightarrow \top) \wedge (s_2 \leftrightarrow \bot)
\end{aligned}$$

$$\rightsquigarrow \text{CNF}$$

$$\rightsquigarrow SAT(\phi_p)$$

$$\rightsquigarrow SAT(\phi), \ \{a \mapsto 3, b \mapsto 0\}$$


ϕ
initial bit-blasting
ϕ_p
different SAT encoding
$SAT(\phi_p)$?
no
?
yes
$SAT(\phi)$
$UNSAT(\phi)$

Unsatisfiable Core $\mathcal{C}_{\phi_p}$

- unsatisfiable core is unsatisfiable subset of clauses
- we are only interested in variables occurring in core
- if $\mathit{UNSAT}(\phi_p) \mapsto$ extract unsatisfiable core
  MiniSat ✗, PicoSat ✓

$\mathcal{C}_{\phi_p} \rightsquigarrow \mathcal{C}_{\phi}$

Refine SAT encoding

use $\mathcal{C}_{\phi}$ to increase the bit-length of some variables

Example I: $a + 2 > 6$

$$
\begin{aligned}
\phi \;:=\; & a + 2 > 6 \\
& \rightsquigarrow [a_1, a_0] + [\top, \bot] > [\top, \top, \bot] \\
& \rightsquigarrow [a_1, a_0] + [\top, \bot] = [s_2, s_1, s_0] \wedge [s_2, s_1, s_0] > [\top, \top, \bot] \wedge \cdots \\
& \rightsquigarrow \phi_p \\
& \rightsquigarrow UNSAT(\phi_p),\ \mathcal{C}_{\phi_p} = \{a_1, s_2, s_1\} \\
& \rightsquigarrow \mathcal{C}_\phi = \{a\}
\end{aligned}
$$

$$
\begin{aligned}
\phi & \rightsquigarrow [a_2, a_1, a_0] + [\top, \bot] > [\top, \top, \bot] \\
& \rightsquigarrow [a_2, a_1, a_0] + [\bot, \top, \bot] = [s_3, s_2, s_1, s_0] \wedge \\
& \quad\; [s_3, s_2, s_1, s_0] > [\bot, \top, \top, \bot] \wedge \cdots \\
& \rightsquigarrow \phi'_p \\
& \rightsquigarrow SAT(\phi'_p)
\end{aligned}
$$

Example II: $a > b > c \geqslant 0 \wedge d = 1$

$$
\begin{aligned}
\phi \;:=\;& a > b \;\wedge\; b > c \;\wedge\; c \geqslant 0 \;\wedge\; d = 1 \\
&\rightsquigarrow [a_0] > [b_0] \;\wedge\; [b_0] > [c_0] \;\wedge\; [c_0] \geqslant [\bot] \;\wedge\; [d_0] = [\top] \\
&\rightsquigarrow \phi_p \\
&\rightsquigarrow UNSAT(\phi_p),\; \mathcal{C}_{\phi_p} \\
&\rightsquigarrow \mathcal{C}_\phi = \{b\}
\end{aligned}
$$

$x \in \mathcal{C}_\phi$: find parent boolean connective, take all its arithmetic variables, increase their bit-length used in the encoding

$$
\begin{aligned}
\phi &\rightsquigarrow [a_1, a_0] > [b_1, b_0] \wedge [b_1, b_0] > [c_1, c_0] \wedge [c_1, c_0] \geqslant [\bot] \wedge [d_0] = [\top] \\
&\rightsquigarrow \phi'_p \\
&\rightsquigarrow SAT(\phi'_p) \\
&\rightsquigarrow SAT(\phi),\; \{a \mapsto 3, b \mapsto 2, c \mapsto 0\}
\end{aligned}
$$

Experimental Results

Thank you for your attention