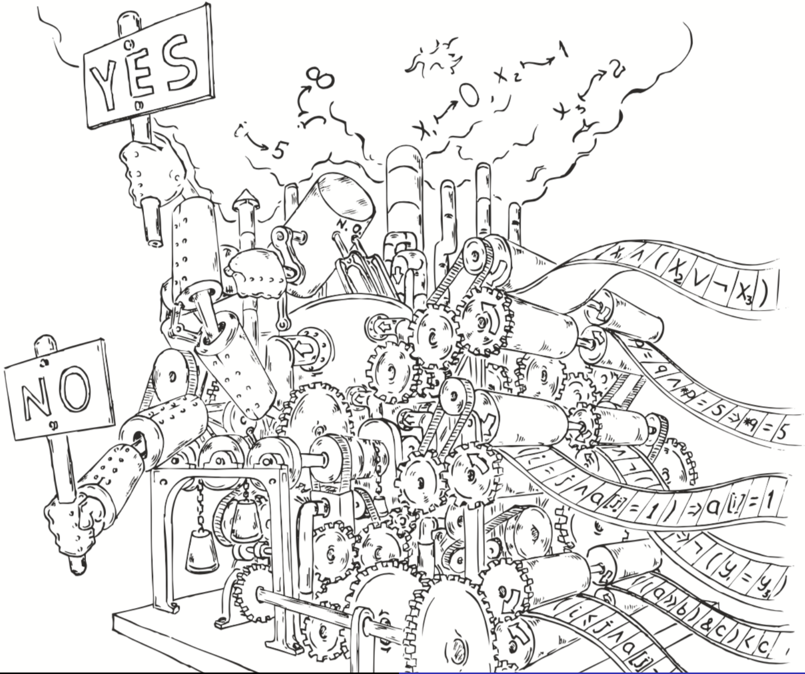

YES
NO
N.O.
X₂ ∨ ¬X₃)
= 5 ⇒ *q = 5
= 1) ⇒ a[j] = 1
¬(y₁ = y₂)
((a>>b)&c)<c

SAT
SMT solvers
5. Linear arithmetic and Bit vectors

Roman Kholin

Lomonosov Moscow State University

Moscow, 2023

Linear arithmetic

Syntax

- $formula : formula \wedge formula \mid \neg formula \mid (formula) \mid atom$
- $atom : sum \text{ op } sum$
- $op : = \mid \leq \mid <$
- $sum : term \mid sum + term$
- $term : identifier \mid constant \mid constant \text{ identifier}$

$2z_1 + 3z_2 \leq 5 \wedge z_2 + 5z_2 - 10z_3 \geq 6 \wedge z1 + z3 = 3$

Decison procedure

Simplex method

Bit-Vector Arithmetic

Syntax

- $formula : formula \wedge formula \mid \neg formula \mid (formula) \mid atom$
- $atom : term\ \ rel\ \ term \mid Boolean\text{-}Identifier \mid term[constant]$
- $rel : < \mid =$
- $term : term \text{ op } term \mid identifier \mid \sim term \mid constant \mid atom?term : term \mid term[constant : constant] \mid ext(term)$
- $op : + \mid - \mid \cdot \mid / \mid << \mid >> \mid \& \mid | \mid \bigoplus \mid \circ$

Motivation

$$(x - y > 0) \iff (x > y)$$

```
unsigned char number = 200;
number = number + 100;
printf("Sum: %d\n", number);
```

$$
\begin{array}{r}
11001000 = 200 \\
+\ 01100100 = 100 \\
\hline
= 00101100 = 44
\end{array}
$$

Definitions

λ-notation

$\lambda i \in \{0, \dots, l-1\}.f(i)$

Bit vector

A bit vector b is a vector of bits with a given length l (or dimension):
$b : \{0, \dots, l-1\} \to \{0, 1\}$

Definitions

Operator $\ll|\gg$

$|_{[l]} : (bvec_l \times bvec_l) \rightarrow bvec_l$
$a|b ::= \lambda i.(a_i \vee b_i)$

Binary encoding

$x = \langle b \rangle_U$ - binary encoding, where
$\langle \cdot \rangle_U : bvec_l \rightarrow \{0, \ldots, 2l - 1\}$,
$\langle b \rangle_U ::= \Sigma_{i=0}^{l-1} b_i \cdot 2^i$

Two's complement

$x = \langle b \rangle_S$ - two's complement, where
$\langle \cdot \rangle_S : bvec_l \rightarrow \{-2^{l-1}, \ldots, 2l - 1 - 1\}$,
$\langle b \rangle_S ::= -2^{l-1} \cdot b_{l-1} + \Sigma_{i=0}^{l-2} b_i \cdot 2^i$

$$\langle 11001000 \rangle_U = 200 ,$$
$$\langle 11001000 \rangle_S = -128 + 64 + 8 = -56 ,$$
$$\langle 01100100 \rangle_S = 100 .$$

Definitions

- Addition and subtraction:

$$
\begin{aligned}
a_{[l]} +_U b_{[l]} = c_{[l]} &\iff \langle a\rangle_U + \langle b\rangle_U = \langle c\rangle_U \mod 2^l \ ,\\
a_{[l]} -_U b_{[l]} = c_{[l]} &\iff \langle a\rangle_U - \langle b\rangle_U = \langle c\rangle_U \mod 2^l \ ,\\
a_{[l]} +_S b_{[l]} = c_{[l]} &\iff \langle a\rangle_S + \langle b\rangle_S = \langle c\rangle_S \mod 2^l \ ,\\
a_{[l]} -_S b_{[l]} = c_{[l]} &\iff \langle a\rangle_S - \langle b\rangle_S = \langle c\rangle_S \mod 2^l \ .
\end{aligned}
$$

- Unary minus:

$$
-a_{[l]} = b_{[l]} \iff -\langle a\rangle_S = \langle b\rangle_S \mod 2^l \ .
$$

- Relational operators:

$$
\begin{aligned}
a_{[l]U} < b_{[l]U} &\iff \langle a\rangle_U < \langle b\rangle_U \ ,\\
a_{[l]S} < b_{[l]S} &\iff \langle a\rangle_S < \langle b\rangle_S \ ,\\
a_{[l]U} < b_{[l]S} &\iff \langle a\rangle_U < \langle b\rangle_S \ ,\\
a_{[l]S} < b_{[l]U} &\iff \langle a\rangle_S < \langle b\rangle_U \ .
\end{aligned}
$$

Definitions

- Multiplication and division:

$$
\begin{aligned}
a_{[l]} \cdot_U b_{[l]} = c_{[l]} &\iff \langle a\rangle_U \cdot \langle b\rangle_U = \langle c\rangle_U \mod 2^l\ , \\
a_{[l]} /_U b_{[l]} = c_{[l]} &\iff \langle a\rangle_U / \langle b\rangle_U = \langle c\rangle_U \mod 2^l\ , \\
a_{[l]} \cdot_S b_{[l]} = c_{[l]} &\iff \langle a\rangle_S \cdot \langle b\rangle_S = \langle c\rangle_S \mod 2^l\ , \\
a_{[l]} /_S b_{[l]} = c_{[l]} &\iff \langle a\rangle_S / \langle b\rangle_S = \langle c\rangle_S \mod 2^l\ .
\end{aligned}
$$

Definitions

Extension:

$$ext_{[m]U}(a_{[l]}) = b_{[m]U} \iff \langle a\rangle_U = \langle b\rangle_U\ ,$$
$$ext_{[m]S}(a_{[l]}) = b_{[m]S} \iff \langle a\rangle_S = \langle b\rangle_S\ .$$

Shifting:

$$a_{[l]} << b_U = \lambda i \in \{0,\ldots,l-1\}. \begin{cases} a_{i-\langle b\rangle_U} & : i \geq \langle b\rangle_U \\ 0 & : \text{otherwise} \end{cases}$$

$$a_{[l]U} >> b_U = \lambda i \in \{0,\ldots,l-1\}. \begin{cases} a_{i+\langle b\rangle_U} & : i < l - \langle b\rangle_U \\ 0 & : \text{otherwise}\ . \end{cases}$$

$$a_{[l]S} >> b_U = \lambda i \in \{0,\ldots,l-1\}. \begin{cases} a_{i+\langle b\rangle_U} & : i < l - \langle b\rangle_U \\ a_{l-1} & : \text{otherwise}\ . \end{cases}$$

Algorithm

- $T(\varphi)$ - the set of terms in φ
- $e(t)$ - vector of variables for a given $t \in T(\varphi)$

Algorithm 6.2.1: BV-FLATTENING

Input: A formula φ in bit-vector arithmetic
Output: An equisatisfiable Boolean formula $\mathcal{B}$

function BV-FLATTENING
    $\mathcal{B}:=e(\varphi)$; ▷ the propositional skeleton of φ
    for each $t_{[l]} \in T(\varphi)$ **do**
        for each $i \in \{0, \ldots, l-1\}$ **do**
            set $e(t)_i$ to a new Boolean variable;
    for each $a \in At(\varphi)$ **do**
        $\mathcal{B}:=\mathcal{B}\wedge$ BV-CONSTRAINT(e, a);
    for each $t_{[l]} \in T(\varphi)$ **do**
        $\mathcal{B}:=\mathcal{B}\wedge$ BV-CONSTRAINT(e, t);
    return $\mathcal{B}$;

Algorithm

For all constant:

$$\bigwedge_{i=0}^{l-1} (C_i \iff e(t)_i)$$

For «|» operator:

$$\bigwedge_{i=0}^{l-1} ((a_i \vee b_i) \iff e(t)_i)$$

Algorithm

Full adder:

$$sum(a, b, cin) \doteq (a \oplus b) \oplus cin \,,$$
$$carry(a, b, cin) \doteq (a \wedge b) \vee ((a \oplus b) \wedge cin)$$

Carry bits:

$$c_i \doteq \begin{cases} cin & : i = 0 \\ carry(x_{i-1}, y_{i-1}, c_{i-1}) & : otherwise \end{cases}$$

Adder:

$$add(x, y, cin) \doteq \langle result, cout \rangle \,,$$
$$result_i \doteq sum(x_i, y_i, c_i) \quad for\ i \in \{0, \ldots, l-1\}$$
$$cout \doteq c_n \,.$$

$$\bigwedge_{i=0}^{l-1} (add(a, b, 0).result_i \iff e(t)_i)$$

Relational Operators

$$\bigwedge_{i=0}^{l-1} a_i = b_i \iff e(t)$$

$$\langle a \rangle_U < \langle b \rangle_U \iff \neg add(a, \sim b, 1).cout$$

$$\langle a \rangle_S < \langle b \rangle_S \iff (a_{l-1} \iff b_{l-1}) \oplus add(a,\ b, 1).cout$$

Shifts

$$ls(a_{[l]}, b_{[n]U}, -1) \doteq \; a \;,$$

$$ls(a_{[l]}, b_{[n]U}, s) \doteq$$

$$\lambda i \in \{0, \ldots, l-1\}. \begin{cases} (ls(a, b, s-1))_{i-2^s} & : i \geq 2^s \wedge b_s \\ (ls(a, b, s-1))_i & : \neg b_s \\ 0 & : \text{otherwise} \;. \end{cases}$$

Multiplication and Division

$$\begin{aligned} mul(a, b, -1) &\doteq 0, \\ mul(a, b, s) &\doteq mul(a, b, s-1) + (b_s ? (a << s) : 0) \end{aligned}$$

$$\begin{aligned} b \neq 0 &\implies e(t) \cdot b + r = a \\ b \neq 0 &\implies r < b\,. \end{aligned}$$

Some Operators Are Hard

n	Number of variables	Number of clauses
8	313	1001
16	1265	4177
24	2857	9529
32	5089	17057
64	20417	68929

Algorithm

Algorithm 6.3.1: INCREMENTAL-BV-FLATTENING

Input: A formula φ in bit-vector logic
Output: "Satisfiable" if the formula is satisfiable, and "Unsatisfiable" otherwise

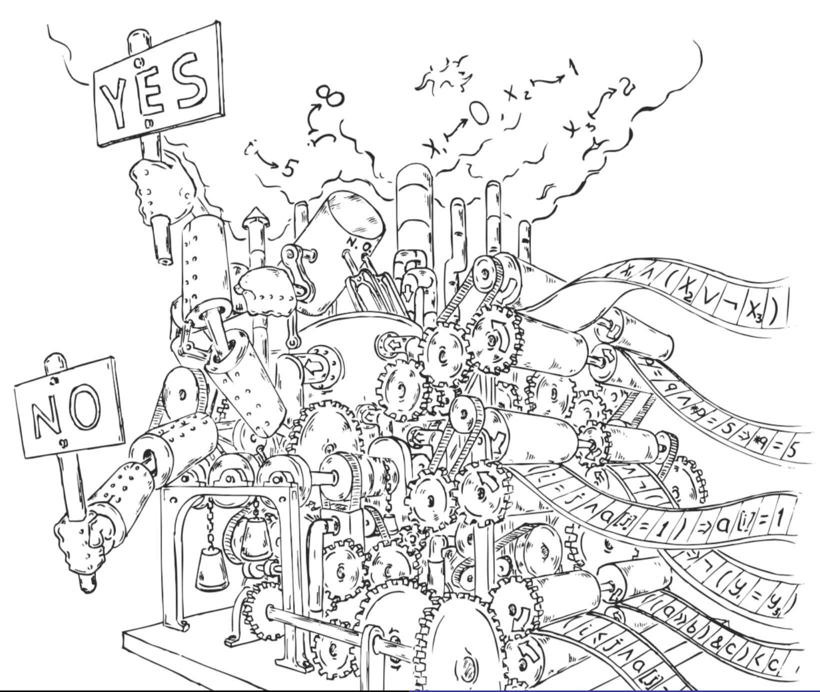
function INCREMENTAL-BV-FLATTENING(φ)
    $\mathcal{B} := e(\varphi)$; $\triangleright$ propositional skeleton of φ
    for each $t_{[l]} \in T(\varphi)$ **do**
        for each $i \in \{0, \ldots, l-1\}$ **do**
            set $e(t)_i$ to a new Boolean variable;
    while (TRUE) **do**
        $\alpha :=$ SAT-SOLVER($\mathcal{B}$);
        if $\alpha=$"Unsatisfiable" **then**
            return "Unsatisfiable";
        else
            Let $I \subseteq T(\varphi)$ be the set of terms that are inconsistent with the satisfying assignment;
            if $I = \emptyset$ **then**
                return "Satisfiable";
            else
                Select "easy" $F' \subseteq I$;
                for each $t_{[l]} \in F'$ **do**
                    $\mathcal{B} := \mathcal{B} \wedge$ BV-CONSTRAINT(e, t);

Fixed-Point Arithmetic

$$\underbrace{\overbrace{\boxed{b_{l-1}}\boxed{b_{l-2}} \cdots \boxed{b_{k+1}}\boxed{b_k}}^{j \text{ bits}} \,.\, \overbrace{\boxed{b_{k-1}}\boxed{b_{k-2}} \cdots \boxed{b_1}\boxed{b_0}}^{k \text{ bits}}}_{l \text{ bits}}$$

$$\langle\cdot\rangle : \{0,1\}^{m+f} \longrightarrow \mathbb{Q}\ ,$$
$$\langle M.F\rangle := \frac{\langle M \circ F\rangle_S}{2^f}\ .$$

$$\begin{aligned}
\langle 0.10\rangle &= 0.5\ ,\\
\langle 0.01\rangle &= 0.25\ ,\\
\langle 01.1\rangle &= 1.5\ ,\\
\langle 11111111.1\rangle &= -0.5\ .
\end{aligned}$$


YES
NO