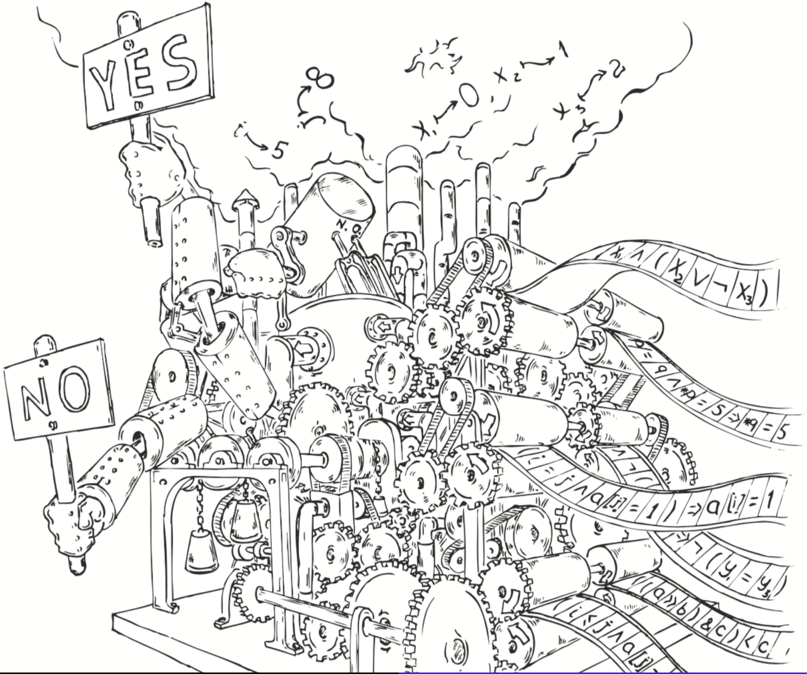

YES
NO
N.O.
$i \to 5$
8
$x_1 \to 0$
$x_2 \to 1$
$x_3 \to 2$
$x_1 \wedge (x_2 \vee \neg x_3)$
$p = q \wedge *p = 5 \Rightarrow *q = 5$
$(i = j \wedge a[j] = 1) \Rightarrow a[i] = 1$
$\Rightarrow \neg (y_1 = y_3)$
$((a \gg b) \& c) < c$
$(i < j \wedge a[j]$

SAT
SMT solvers
5. Linear arithmetic and Bit vectors

Roman Kholin

Lalambda

Tbilisi, 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

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

Bit vector

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

Definitions

Operator «|»

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

Binary encoding

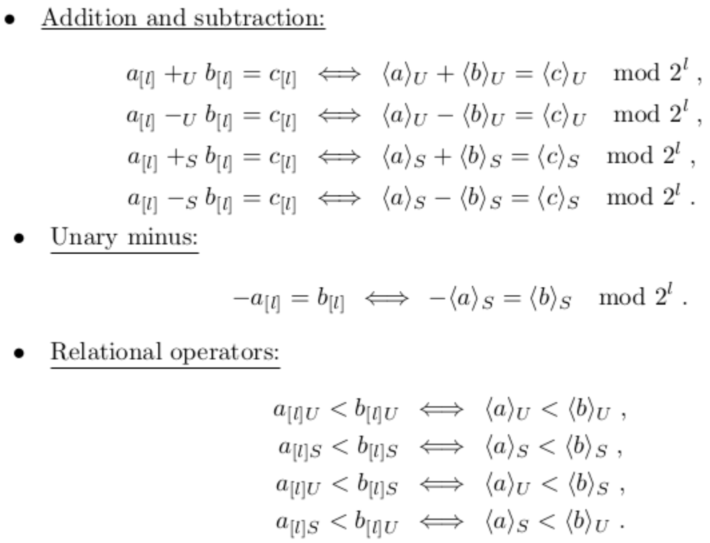
$x = \langle b \rangle_U$ - binary encoding, where
$\langle \cdot \rangle_U : bvec_l \rightarrow \{0, \ldots, 2l - 1\}$,
$\langle b \rangle_U ::= \Sigma_{i=0}^{l-1} b_i \cdot 2^i$

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:

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

Shifting:

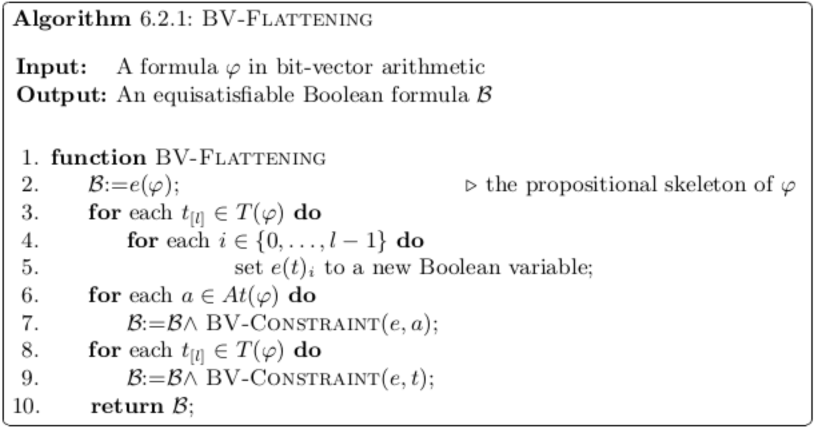
$$
a_{[l]} << b_U = \lambda i \in \{0, \ldots, l-1\}. \begin{cases} a_{i-\langle b\rangle_U} & : i \geq \langle b\rangle_U \\ 0 & : \text{otherwise} \end{cases}
$$

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

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

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 $\varphi$
    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:

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

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

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

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

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

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

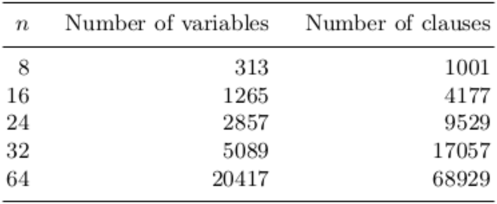
Multiplication and Division

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

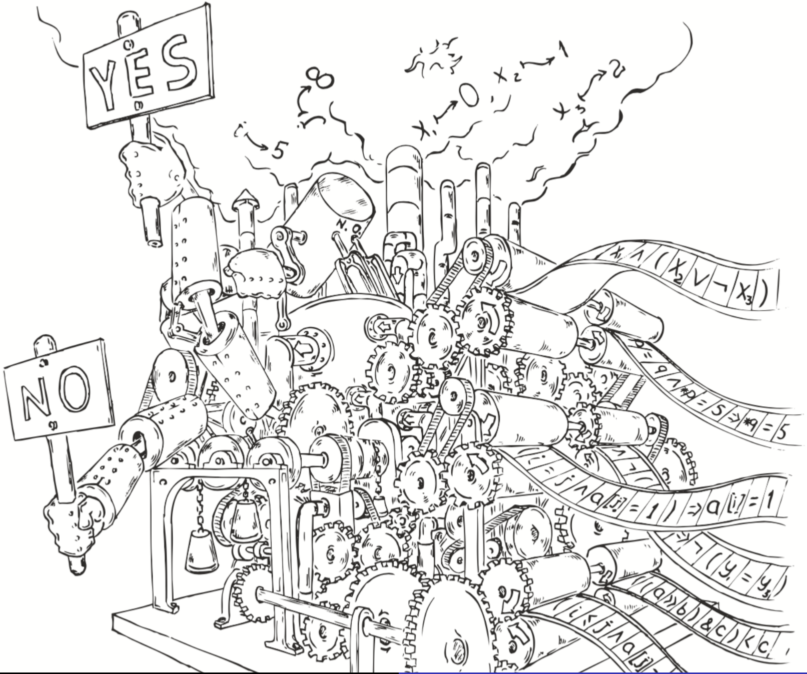
$$b \neq 0 \implies e(t) \cdot b + r = a$$

$$b \neq 0 \implies r < b\,.$$

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


YES
NO
N.O.
i → 5
j → 8
x₁ → 0
x₂ → 1
x₃ → 2
x₁ ∧ (x₂ ∨ ¬x₃)
p = q ∧ *p = 5 ⇒ *q = 5
(i = j ∧ a[j] = 1) ⇒ a[i] = 1
⇒ ¬(y₁ = y₃)
((a >> b) & c) < c
(i < j ∧ a[j]