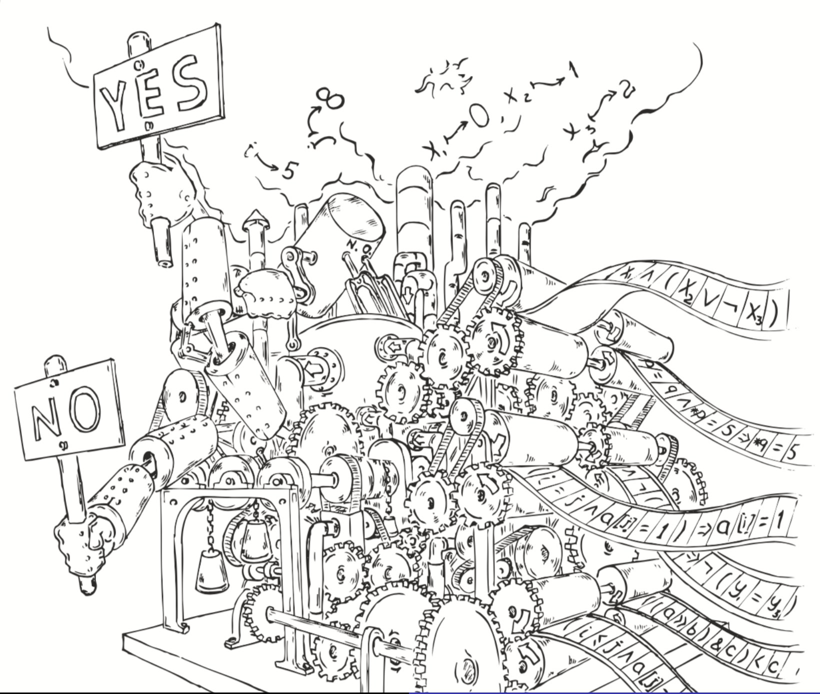

YES
NO

SAT/SMT solvers
1. Arrays

Roman Kholin

Lalambda

Tbilisi, 2023

Example

Verification conditions:

$$\begin{aligned} &(\forall x \in \mathbb{N}_0.\ x < i \implies \mathsf{a}[x] = 0) \\ &\wedge\ \mathsf{a}' = \mathsf{a}\{i \leftarrow 0\} \\ \implies &(\forall x \in \mathbb{N}_0.\ x \leq i \implies \mathsf{a}'[x] = 0) \end{aligned}$$

```
a: array 0..99 of integer;
i: integer;

for i:=0 to 99 do
        assert(∀x ∈ ℕ0. x < i ⟹ a[x] = 0);
        a[i]:=0;
        assert(∀x ∈ ℕ0. x ≤ i ⟹ a[x] = 0);
done;
assert(∀x ∈ ℕ0. x ≤ 99 ⟹ a[x] = 0);
```

$a[i]$ - array reading operator

$a\{i \leftarrow x\}$ - array update operator

$(\forall x \in \mathbb{N}_0 . x < i \implies a[x] = 0) \wedge a' = a\{i \leftarrow 0\} \implies$
$(\forall x \in \mathbb{N}_0 . x \leq i \implies a'[x] = 0)$

Definitions

- T_I - index theory
- T_E - element theory
- T_A - array theory, $T_I \to T_E$

Syntax

$termA$: *array-identifier* | $\text{term}_A\{\text{term}_I \leftarrow \text{term}_E\}$
$termE$: $\text{term}_A[\text{term}_I]$ | $(previousrules)$
$formula$: $\text{term}_A = \text{term}_A$ | $(previousrules)$

Definitions

Semantics

- $\forall a_1 \in T_A.\forall a_2 \in T_A.\forall i \in T_I.\forall j \in T_I.(a_1 = a_2 \wedge i = j) \implies a_1[i] = a_2[j]$
- $\forall a \in T_A.\forall e \in T_E.\forall i \in T_I.\forall j \in T_I.a\{i \leftarrow e\}[j] = (i = j)?e : a[j]$
- $\forall a_1 \in T_A.\forall a_2 \in T_A.(\forall i \in T_I.a_1[i] = a_2[i]) \implies a_1 = a_2$

Eliminating the array terms

We can therefore replace the array index operator by an uninterpreted function:

$(i = j \wedge a[j] =' z') \implies a[i] =' z'$

$(i = j \wedge F_a(i) =' z') \implies F_a(i) =' z'$

Write rule

Semantics

- $a'[i] = e$ for the value that is written,
- $\forall j \neq i . a'[j] = a[j]$ for the values that are unchanged.

$$a[0] = 10 \implies a\{1 \leftarrow 20\}[0] = 10$$

Write rule

Semantics

- $a'[i] = e$ for the value that is written,
- $\forall j \neq i . a'[j] = a[j]$ for the values that are unchanged.

$$a[0] = 10 \implies a\{1 \leftarrow 20\}[0] = 10$$
$$(a[0] = 10 \land a'[1] = 20 \land (\forall j \neq 1 . a'[j] = a[j])) \implies a_0[0] = 10$$
$$(F_a(0) = 10 \land F_{a'}(1) = 20 \land (\forall j \neq 1 . F_{a'}(j) = F_a(j))) \implies$$
$$F_{a'}(0) = 10$$

Array property

An array theory formula is called an array property if and only if it is of the form
$\forall i_1 \dots \forall i_k \in T_I.\phi I(i_1, \dots, i_k) \Rightarrow \phi_V(i_1, \dots, i_k)$
and satisfies the following conditions:

- The predicate ϕ , called the index guard, must follow the grammar
  $iguard : iguard \wedge iguard \mid iguard \vee iguard \mid iterm \leq iterm \mid iterm = iterm$
  $iterm : i_1 \mid \dots \mid i_k \mid term$
  $term : integer\text{-}constant \mid integer\text{-}constant * index\text{-}identifier \mid term + term$
  The *index-identifier* used in *term* must not be one of $i_1, \dots, i_k$
- The index variables $i_1, \dots, i_k$ can only be used in array read expressions of the form $a[i_j]$

The predicate ϕ_V is called the value constraint

Example

$$a' = a\{i \leftarrow 0\}$$

Example

$$a' = a\{i \leftarrow 0\}$$
$$\forall j \neq i . a'[j] = a[j]$$

Example

$$a' = a\{i \leftarrow 0\}$$
$$\forall j \neq i . a'[j] = a[j]$$
$$\forall j . (j \leq i - 1 \wedge i + 1 \leq j) \implies a'[j] = a[j]$$

$\iota(\phi)$

- All expressions used as an array index in ϕ that are not quantified variables
- All expressions used inside index guards in ϕ that are not quantified variables
- If ϕ contains none of the above, $\iota(\phi)$ is $\{0\}$ in order to obtain a nonempty set of index expressions

Array reduction

Input: An array property formula ϕ_A in NNF
Output: A formula ϕ_{UF} in the index and element theories with uninterpreted functions

1. Apply the write rule to remove all array updates from ϕ_A.
2. Replace all existential quantifications of the form $\exists i \in T_I.\ P(i)$ by $P(j)$, where j is a fresh variable.
3. Replace all universal quantifications of the form $\forall i \in T_I.\ P(i)$ by

$$\bigwedge_{i \in \mathcal{I}(\phi)} P(i)\ .$$

4. Replace the array read operators by uninterpreted functions and obtain ϕ_{UF};
5. **return** ϕ_{UF};

Example

$$(\forall x \in \mathbb{N}_0 . x < i \implies a[x] = 0) \land a' = a\{i \leftarrow 0\} \implies$$
$$(\forall x \in \mathbb{N}_0 . x \leq i \implies a'[x] = 0)$$

Example

$$(\forall x \in \mathbb{N}_0 . x < i \implies a[x] = 0) \land a' = a\{i \leftarrow 0\} \implies$$
$$(\exists x \in \mathbb{N}_0 . x \leq i \land a'[x] \neq 0)$$

Example

$$(\forall x \in \mathbb{N}_0 . x < i \implies a[x] = 0) \land a'[i] = 0 \land \forall j \neq i . a'[j] = a[j] \implies (\exists x \in \mathbb{N}_0 . x \leq i \land a'[x] \neq 0)$$

Example

$$(\forall x \in \mathbb{N}_0 . x < i \implies a[x] = 0) \wedge a'[i] = 0 \wedge \forall j \neq i . a'[j] = a[j] \implies (z \leq i \wedge a'[z] \neq 0)$$

Example

$\iota(\phi) = \{i, z\}$

Example

$$\iota(\phi) = \{i, z\}$$
$$(i < i \implies a[i] = 0) \wedge (z < i \implies a[z] = 0) \wedge$$
$$a'[i] = 0 \wedge \forall j \neq i . a'[j] = a[j] \implies$$
$$(z \leq i \wedge a'[z] \neq 0)$$

Example

$$\iota(\phi) = \{i, z\}$$
$$(i < i \implies a[i] = 0) \wedge (z < i \implies a[z] = 0) \wedge$$
$$a'[i] = 0 \wedge (i \neq i \implies a'[i] = a[i]) \wedge (z \neq i \implies a'[z] = a[z]) \implies$$
$$(z \leq i \wedge a'[z] \neq 0)$$

Example

$$(z < i \implies a[z] = 0) \wedge$$
$$a'[i] = 0 \wedge (z \neq i \implies a'[z] = a[z]) \implies$$
$$(z \leq i \wedge a'[z] \neq 0)$$

Example

$$(z < i \implies F_a(z) = 0) \wedge$$
$$F_{a'}(i) = 0 \wedge (z \neq i \implies F_{a'}(z) = F_a(z)) \implies$$
$$(z \leq i \wedge F_{a'}(z) \neq 0)$$

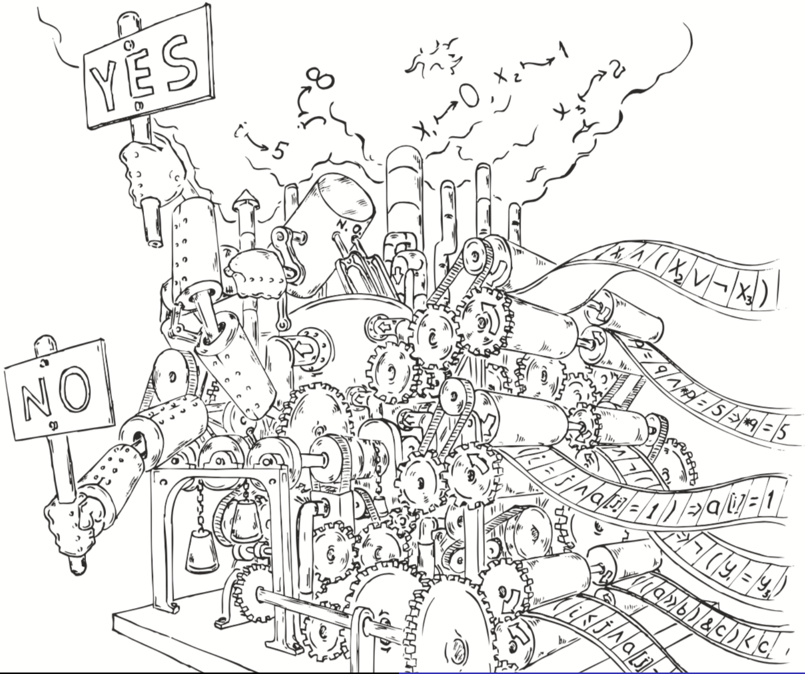
Example

$$
\begin{aligned}
&(z < i \implies F_a(z) = 0) \land \\
&F_{a'}(i) = 0 \land (z \neq i \implies F_{a'}(z) = F_a(z)) \implies \\
&(z \leq i \land F_{a'}(z) \neq 0)
\end{aligned}
$$

Example

$$(z < i \implies F_a(z) = 0) \land$$
$$F_{a'}(i) = 0 \land (z \neq i \implies F_{a'}(z) = F_a(z)) \implies$$
$$(z \leq i \land F_{a'}(z) \neq 0)$$

By distinguishing the three cases $z < i$, $z = i$, and $z > i$, it is easy to see that this formula is unsatisfiable


YES
NO