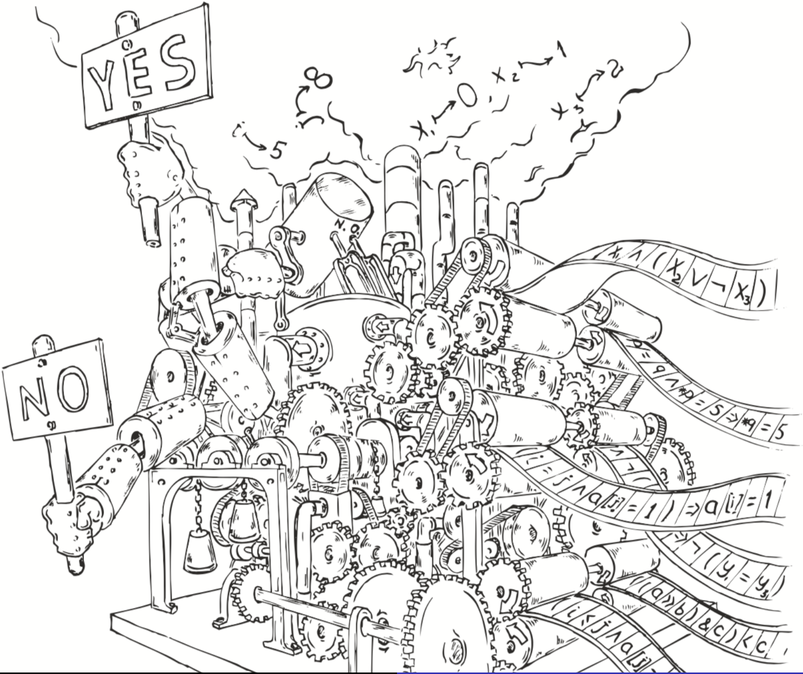

YES
NO

SAT/SMT solvers
7. Pointer Logic

Roman Kholin

Lalambda

Tbilisi, 2023

Definitions

Memory model

A memory model describes the assumptions that are made about the way memory cells are addressed. We assume that the architecture provides a continuous, uniform address space, i.e., the set of addresses A is a subinterval of the integers $\{0, \ldots, N-1\}$. Each address corresponds to a memory cell that is able to store one data word. The set of data words is denoted by D. A *memory valuation* $M : A \rightarrow D$ is a mapping from a set of addresses A into the domain D of data words

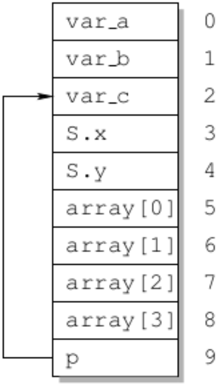
Memory layout

A memory layout $L : V \rightarrow A$ is a mapping from each variable $v \in V$ to an address $a \in A$. The address of v is also called the memory location of v

Example

```
int var_a, var_b, var_c;
struct { int x; int y; } S;
int array[4];
int *p = &var_c;

int main() {
  *p=100;
}
```



Analysis of programs with pointers

```
void f(int *sum) {
  *sum = 0;

  for(i=0; i<10; i++)
    *sum = *sum + array[i];
}
```

```
int *p, *q;

p = new int[10];
q = &p[3];
delete p;
*q = 2;
```

Definitions

Pointer logic

- $formula : formula \wedge formula \mid \neg formula \mid (formula) \mid atom$
- $pointer = pointer \mid term = term \mid pointer < pointer \mid term < term$
- $pointer : pointer\text{-}identifier \mid pointer + term \mid (pointer) \mid \&identifier \mid \& * pointer \mid pointer \mid \text{NULL}$
- $term : identifier \mid *pointer \mid term\ op\ term \mid (term) \mid integer\text{-}constant \mid identifier[term]$
- $op : + \mid -$

Example

$$
\begin{array}{ll}
*(p+i)=1, & p+i, \\
*(p+*p)=0, & p=i, \\
p=q \wedge *p=5, & *(p+q), \\
*****p=1, & *1=1, \\
p<q. & p<i.
\end{array}
$$

Semantics

$$
\begin{array}{rcll}
[\![f_1 \wedge f_2]\!] & \doteq & [\![f_1]\!] \wedge [\![f_2]\!] & \\
[\![\neg f]\!] & \doteq & \neg [\![f]\!] & \\
[\![p_1 = p_2]\!] & \doteq & [\![p_1]\!] = [\![p_2]\!] & \text{where } p_1,\, p_2 \text{ are pointer expressions} \\
[\![p_1 < p_2]\!] & \doteq & [\![p_1]\!] < [\![p_2]\!] & \text{where } p_1,\, p_2 \text{ are pointer expressions} \\
[\![t_1 = t_2]\!] & \doteq & [\![t_1]\!] = [\![t_2]\!] & \text{where } t_1,\, t_2 \text{ are terms} \\
[\![t_1 < t_2]\!] & \doteq & [\![t_1]\!] < [\![t_2]\!] & \text{where } t_1,\, t_2 \text{ are terms} \\
[\![p]\!] & \doteq & M[L[p]] & \text{where } p \text{ is a pointer identifier} \\
[\![p + t]\!] & \doteq & [\![p]\!] + [\![t]\!] & \text{where } p \text{ is a pointer expression, and } t \text{ is a term} \\
[\![\&v]\!] & \doteq & L[v] & \text{where } v \in V \text{ is a variable} \\
[\![\& * p]\!] & \doteq & [\![p]\!] & \text{where } p \text{ is a pointer expression} \\
[\![\mathrm{NULL}]\!] & \doteq & 0 & \\
[\![v]\!] & \doteq & M[L[v]] & \text{where } v \in V \text{ is a variable} \\
[\![*p]\!] & \doteq & M[[\![p]\!]] & \text{where } p \text{ is a pointer expression} \\
[\![t_1 \; op \; t_2]\!] & \doteq & [\![t_1]\!] \; op \; [\![t_2]\!] & \text{where } t_1,\, t_2 \text{ are terms} \\
[\![c]\!] & \doteq & c & \text{where } c \text{ is an integer constant} \\
[\![v[t]]\!] & \doteq & M[L[v] + [\![t]\!]] & \text{where } v \text{ is an array identifier, and } t \text{ is a term}
\end{array}
$$

Example

$$*((\&a)+1) = a[1]$$

Example

$*((\&a) + 1) = a[1]$

$$
\begin{aligned}
[\![*((\&a)+1) = a[1]]\!] &\iff [\![*((\&a)+1)]\!] = [\![a[1]]\!] \\
&\iff M[[\![(\&a)+1]\!]] = M[L[a] + [\![1]\!]] \\
&\iff M[[\![\&a]\!] + [\![1]\!]] = M[L[a] + 1] \\
&\iff M[L[a]+1] = M[L[a]+1]
\end{aligned}
$$

Memory Model Axiom 1 («No object has address 0»)

The fact «no object has address 0» is easily formalized:
$\forall v \in V . L[v] \neq 0$

Memory Model Axiom 2 («Objects have size at least one»)

The fact «an object has size at least one» is easily captured by
$\forall v \in V . \sigma(v) \geq 1$

Memory Model Axiom 3 («Objects do not overlap»)

Different objects do not share any addresses:
$\forall v_1, v_2 \in V . v_1 \neq v_2 \Rightarrow$
$\{L[v_1], \ldots, L[v_1] + \sigma(v1) - 1\} \cap \{L[v_2], \ldots, L[v_2] + \sigma(v_2) - 1\} = \emptyset$

Adding structure types

$$s.f \quad \doteq \quad *((\&s) + o(f))$$

$$*(p+0) = a \ \wedge$$
$$*(p+1) = b \ \wedge$$
$$*(p+2) = c \ \ldots$$

A decision procedure

- The logic of pointers is reduced to the logic of arrays
- The formulas generated by this semantic translation contain array read operators and linear arithmetic over the type that is used for the indices
- Problem - quantors

Example

$$p = \&x \land x = 1 \Rightarrow *p = 1$$

Example

$p = \&x \wedge x = 1 \Rightarrow *p = 1$

$$
\begin{aligned}
&[\![p = \&x \wedge x = 1 \implies *p = 1]\!] \\
&\quad \iff [\![p = \&x]\!] \wedge [\![x = 1]\!] \implies [\![*p = 1]\!] \\
&\quad \iff [\![p]\!] = [\![\&x]\!] \wedge [\![x]\!] = 1 \implies [\![*p]\!] = 1 \\
&\quad \iff M[L[p]] = L[x] \wedge M[L[x]] = 1 \implies M[M[L[p]]] = 1
\end{aligned}
$$

Example

$p \to x \Rightarrow p = \&x$

$$
\begin{aligned}
&[\![p \hookrightarrow x \implies p = \&x]\!] \\
&\quad \iff [\![p \hookrightarrow x]\!] \implies [\![p = \&x]\!] \\
&\quad \iff [\![*p = x]\!] \implies [\![p]\!] = [\![\&x]\!] \\
&\quad \iff [\![*p]\!] = [\![x]\!] \implies M[L[p]] = L[x] \\
&\quad \iff M[M[L[p]]] = M[L[x]] \implies M[L[p]] = L[x]
\end{aligned}
$$

Example

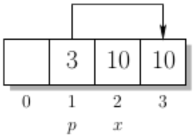
$p \rightarrow x \Rightarrow p = \&x$

$$
\begin{aligned}
&[\![p \hookrightarrow x \implies p = \&x]\!] \\
&\quad \iff [\![p \hookrightarrow x]\!] \implies [\![p = \&x]\!] \\
&\quad \iff [\![*p = x]\!] \implies [\![p]\!] = [\![\&x]\!] \\
&\quad \iff [\![*p]\!] = [\![x]\!] \implies M[L[p]] = L[x] \\
&\quad \iff M[M[L[p]]] = M[L[x]] \implies M[L[p]] = L[x]
\end{aligned}
$$

Example

$$
\begin{aligned}
&\llbracket p \hookrightarrow x \implies p = \&x \rrbracket \\
&\quad \iff \llbracket p \hookrightarrow x \rrbracket \implies \llbracket p = \&x \rrbracket \\
&\quad \iff \llbracket *p = x \rrbracket \implies \llbracket p \rrbracket = \llbracket \&x \rrbracket \\
&\quad \iff \llbracket *p \rrbracket = \llbracket x \rrbracket \implies M[L[p]] = L[x] \\
&\quad \iff M[M[L[p]]] = M[L[x]] \implies M[L[p]] = L[x]
\end{aligned}
$$

$$L[p] = 1,\ L[x] = 2,\ M[1] = 3,\ M[2] = 10,\ M[3] = 10$$



Applying the Memory model axioms

$$\sigma(x) = 2 \implies \&y \neq \&x + 1$$

Applying the Memory model axioms

$$\sigma(x) = 2 \implies \&y \neq \&x + 1$$

$$\sigma(x) = 2 \implies L[y] \neq L[x] + 1$$

Applying the Memory model axioms

$$\sigma(x) = 2 \implies \&y \neq \&x + 1$$

$$\sigma(x) = 2 \implies L[y] \neq L[x] + 1$$

$$\{L[x], \ldots, L[x] + \sigma(x) - 1\} \cap \{L[y], \ldots, L[y] + \sigma(y) - 1\} = \emptyset$$

Applying the Memory model axioms

$$\sigma(x) = 2 \implies \&y \neq \&x + 1$$

$$\sigma(x) = 2 \implies L[y] \neq L[x] + 1$$

$$\{L[x], \ldots, L[x] + \sigma(x) - 1\} \cap \{L[y], \ldots, L[y] + \sigma(y) - 1\} = \emptyset$$

$$(L[x] + \sigma(x) - 1 < L[y]) \vee (L[x] > L[y] + \sigma(y) - 1)$$

Applying the Memory model axioms

$$\sigma(x) = 2 \implies \&y \neq \&x + 1$$

$$\sigma(x) = 2 \implies L[y] \neq L[x] + 1$$

$$\{L[x], \ldots, L[x] + \sigma(x) - 1\} \cap \{L[y], \ldots, L[y] + \sigma(y) - 1\} = \emptyset$$

$$(L[x] + \sigma(x) - 1 < L[y]) \vee (L[x] > L[y] + \sigma(y) - 1)$$

$$(L[x] + 1 < L[y]) \vee (L[x] > L[y])$$

Pure variables

$$
\begin{aligned}
&[\![x = y \implies y = x]\!] \\
&\quad \iff [\![x = y]\!] \implies [\![y = x]\!] \\
&\quad \iff M[L[x]] = M[L[y]] \implies M[L[y]] = M[L[x]]
\end{aligned}
$$

Pure variables

$$
\begin{aligned}
&[\![x = y \implies y = x]\!] \\
&\quad \iff [\![x = y]\!] \implies [\![y = x]\!] \\
&\quad \iff M[L[x]] = M[L[y]] \implies M[L[y]] = M[L[x]]
\end{aligned}
$$

$$x = y \implies y = x$$

Pure variables

$\mathcal{P}(\&x = y)$

$$
\begin{array}{lll}
[\![v]\!]^{\mathcal{P}} & \doteq & \Upsilon_v \qquad \text{for } v \in \mathcal{P}(\varphi) \\
[\![v]\!]^{\mathcal{P}} & \doteq & M[L[v]] \quad \text{for } v \in V \setminus \mathcal{P}(\varphi)
\end{array}
$$

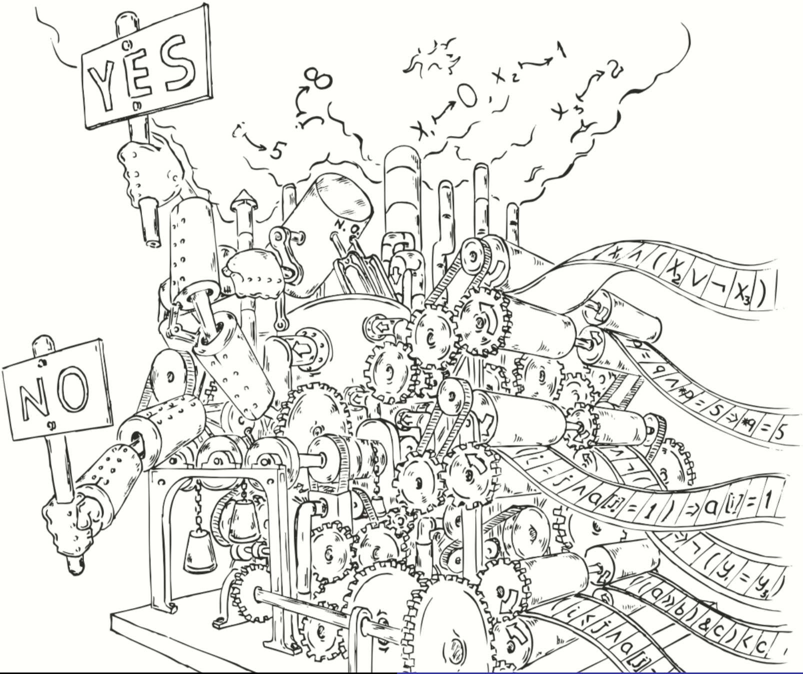
Pure variables

$\mathcal{P}(\&x = y)$

$$\begin{array}{lll} [\![v]\!]^{\mathcal{P}} & \doteq & \Upsilon_v & \text{for } v \in \mathcal{P}(\varphi) \\ [\![v]\!]^{\mathcal{P}} & \doteq & M[L[v]] & \text{for } v \in V \setminus \mathcal{P}(\varphi) \end{array}$$

Theorem:

$$[\![\varphi]\!]^{\mathcal{P}} \iff [\![\varphi]\!]$$


YES
NO