



# SAT/SMT solvers

## 7. Pointer Logic

Roman Kholin

Lambda

Tbilisi, 2023

## Memory model

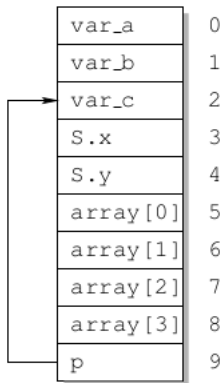
A memory model describes the assumptions that are made about the way memory cells are addressed. We assume that the architecture provides a continuous, uniform address space, i.e., the set of addresses  $A$  is a subinterval of the integers  $\{0, \dots, N - 1\}$ . Each address corresponds to a memory 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;
```

## Pointer logic

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

# Example

$*(p + i) = 1,$	$p + i,$
$*(p + *p) = 0,$	$p = i,$
$p = q \wedge *p = 5,$	$*(p + q),$
$*****p = 1,$	$*1 = 1,$
$p < q.$	$p < i.$

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



# Example

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

# Example

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

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

## 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:

$$\forall v_1, v_2 \in V. v_1 \neq v_2 \Rightarrow$$

$$\{L[v_1], \dots, L[v_1] + \sigma(v_1) - 1\} \cap \{L[v_2], \dots, L[v_2] + \sigma(v_2) - 1\} = \emptyset$$

# Adding structure types

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

$$*(p + 0) = a \wedge$$

$$*(p + 1) = b \wedge$$

$$*(p + 2) = c \dots$$

# 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

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

# Example

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

$$\llbracket p = \&x \wedge x = 1 \implies *p = 1 \rrbracket$$

$$\iff \llbracket p = \&x \rrbracket \wedge \llbracket x = 1 \rrbracket \implies \llbracket *p = 1 \rrbracket$$

$$\iff \llbracket p \rrbracket = \llbracket \&x \rrbracket \wedge \llbracket x \rrbracket = 1 \implies \llbracket *p \rrbracket = 1$$

$$\iff M[L[p]] = L[x] \wedge M[L[x]] = 1 \implies M[M[L[p]]] = 1$$

# Example

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

$$\llbracket p \hookrightarrow x \rrbracket \Longrightarrow p = \&x$$

$$\Longleftrightarrow \llbracket p \hookrightarrow x \rrbracket \Longrightarrow \llbracket p = \&x \rrbracket$$

$$\Longleftrightarrow \llbracket *p = x \rrbracket \Longrightarrow \llbracket p \rrbracket = \llbracket \&x \rrbracket$$

$$\Longleftrightarrow \llbracket *p \rrbracket = \llbracket x \rrbracket \Longrightarrow M[L[p]] = L[x]$$

$$\Longleftrightarrow M[M[L[p]]] = M[L[x]] \Longrightarrow M[L[p]] = L[x]$$



# Example

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

$$\llbracket p \hookrightarrow x \rrbracket \Longrightarrow p = \&x$$

$$\Longleftrightarrow \llbracket p \hookrightarrow x \rrbracket \Longrightarrow \llbracket p = \&x \rrbracket$$

$$\Longleftrightarrow \llbracket *p = x \rrbracket \Longrightarrow \llbracket p \rrbracket = \llbracket \&x \rrbracket$$

$$\Longleftrightarrow \llbracket *p \rrbracket = \llbracket x \rrbracket \Longrightarrow M[L[p]] = L[x]$$

$$\Longleftrightarrow M[M[L[p]]] = M[L[x]] \Longrightarrow M[L[p]] = L[x]$$

# Example

$$\llbracket p \hookrightarrow x \rrbracket \implies p = \&x$$

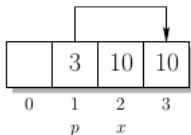
$$\iff \llbracket p \hookrightarrow x \rrbracket \implies \llbracket p = \&x \rrbracket$$

$$\iff \llbracket *p = x \rrbracket \implies \llbracket p \rrbracket = \llbracket \&x \rrbracket$$

$$\iff \llbracket *p \rrbracket = \llbracket x \rrbracket \implies M[L[p]] = L[x]$$

$$\iff M[M[L[p]]] = M[L[x]] \implies M[L[p]] = L[x]$$

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



# 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

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

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

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

# Applying the Memory model axioms

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

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

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

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

# Applying the Memory model axioms

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

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

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

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

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

# Pure variables

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



# Pure variables

$$\llbracket x = y \implies y = x \rrbracket$$

$$\iff \llbracket x = y \rrbracket \implies \llbracket y = x \rrbracket$$

$$\iff M[L[x]] = M[L[y]] \implies M[L[y]] = M[L[x]]$$

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

# Pure variables

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

$$\begin{aligned} \llbracket v \rrbracket^{\mathcal{P}} &\doteq \mathcal{I}_v && \text{for } v \in \mathcal{P}(\varphi) \\ \llbracket v \rrbracket^{\mathcal{P}} &\doteq M[L[v]] && \text{for } v \in V \setminus \mathcal{P}(\varphi) \end{aligned}$$

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

$$\begin{aligned} \llbracket v \rrbracket^{\mathcal{P}} &\doteq \mathcal{I}_v && \text{for } v \in \mathcal{P}(\varphi) \\ \llbracket v \rrbracket^{\mathcal{P}} &\doteq M[L[v]] && \text{for } v \in V \setminus \mathcal{P}(\varphi) \end{aligned}$$

Theorem:

$$\llbracket \varphi \rrbracket^{\mathcal{P}} \iff \llbracket \varphi \rrbracket$$

