The WISL Memory Model

14th Summer School on Formal Techniques

Menlo College

May, 2025

What is WISL?

- WISL: While with a Simplfied C memory model (block-offset)
  - Uses pointers and pointer arithmetic
  - Support for more language errors (e.g. Use After Free)
  - C-style deallocation
  - Used in Gillian

The Syntax of WISL

Values

$$p \in \mathsf{Ptr} \stackrel{\text{def}}{=} \mathcal{L} \times \mathbb{N}$$

$$v \in \mathsf{Val} \supseteq \mathbb{N} \cup \mathsf{Bool} \cup \{\texttt{null}\} \cup \mathsf{Ptr}$$

Expressions

$$E, E_1, E_2, \ldots \in \mathsf{Exp} \stackrel{\text{def}}{=} v \in \mathsf{Val} \mid \mathtt{x} \in \mathsf{PVar} \mid \ominus E \mid E \oplus E$$

Commands

$$\begin{array}{lll} C \in \mathcal{C}_W \stackrel{\text{def}}{=} & \mathtt{skip} \mid \mathtt{x} := E \mid \mathtt{x} := f(\vec{E}) & \text{Basic} \\ & \mid \mathtt{x} := [E] \mid [E] := E \mid \mathtt{x} := \mathtt{new}(E) \mid \mathtt{free}(E) & \text{Memory management} \\ & \mid C; C \mid \mathtt{if}\ (E)\ C\ \mathtt{else}\ C \mid \mathtt{while}\ (E)\ C & \text{Control flow} \end{array}$$

The Semantics of WISL: What's new?

Pointers

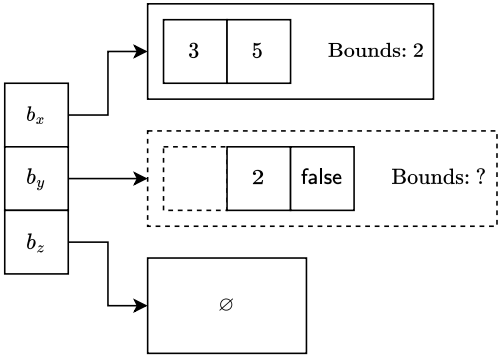
- Block-offset pairs: (l, n)
- Allocation creates a new block b of a given size n and returns the pair $(b, 0)$
- Deallocation can free entire blocks only

Partial Memory

$$h : \quad \mathcal{L} \rightharpoonup_{\text{fin}} ((\mathbb{N} \rightharpoonup_{\text{fin}} \mathcal{V}_W) \times \mathbb{N}^?)_\varnothing$$

The Memory Model Visualised

Partial Memory $h : \quad \mathcal{L} \rightharpoonup_{\mathsf{fin}} ((\mathbb{N} \rightharpoonup_{\mathsf{fin}} \mathcal{V}_W) \times \mathbb{N}^?)_\varnothing$



A Tractable Assertion Language for Tools

Pure: $P_\pi, Q_\pi \in \mathbb{A}_\pi \stackrel{\text{def}}{=} \mathsf{True} \mid \mathsf{False} \mid E = E \mid E < E \mid \ldots \mid \neg P_\pi \mid P_\pi \wedge P_\pi \mid P_\pi \vee P_\pi.$

Spatial: $P_\sigma, Q_\sigma \in \mathbb{A}_\sigma \stackrel{\text{def}}{=} \mathsf{emp} \mid P_\sigma \star P_\sigma \mid E \mapsto E \mid \mathsf{bound}(E, E) \mid E \mapsto \varnothing$

All: $P, Q \in \mathbb{A} \quad : \quad \mathbb{A}_\pi \times \mathbb{A}_\sigma$

What are the main differences?

- Conjunction and negation of pure formulae only
- No *explicit* existential quantification

WISL-Specific Assertions

WISL-Specific Assertions

$$E \mapsto E \mid \mathsf{bound}(E, E) \mid E \mapsto \varnothing$$

- $E_1 \mapsto E_2$ describes the single cell, as in While
- $\mathsf{bound}(E_1, E_2)$ states that E_1 is a block pointer, pointing to a block of length E_2
- $E \mapsto \varnothing$ states that E is a block pointer and its entire block has been freed

WISL-Specific Assertions

WISL-Specific Assertions

$$E \mapsto E \mid \mathsf{bound}(E, E) \mid E \mapsto \varnothing$$

- $E_1 \mapsto E_2$ describes the single cell, as in While
- $\mathsf{bound}(E_1, E_2)$ states that E_1 is a block pointer, pointing to a block of length E_2
- $E \mapsto \varnothing$ states that E is a block pointer and its entire block has been freed

Derived Predicates

- $E \mapsto E_1, \ldots, E_n$ describes n consecutive cells in a memory block, as usual
- $E \mapsto_b E_1, \ldots, E_n \stackrel{\text{def}}{=} E \mapsto E_1, \ldots, E_n \star \mathsf{bound}(E, n)$ describes a complete block of size n at E

WISL-Specific Assertions

WISL-Specific Assertions

$$E \mapsto E \mid \mathsf{bound}(E, E) \mid E \mapsto \varnothing$$

- $E_1 \mapsto E_2$ describes the single cell, as in While
- $\mathsf{bound}(E_1, E_2)$ states that E_1 is a block pointer, pointing to a block of length E_2
- $E \mapsto \varnothing$ states that E is a block pointer and its entire block has been freed

Derived Predicates

- $E \mapsto E_1, \ldots, E_n$ describes n consecutive cells in a memory block, as usual
- $E \mapsto_b E_1, \ldots, E_n \stackrel{\text{def}}{=} E \mapsto E_1, \ldots, E_n \star \mathsf{bound}(E, n)$ describes a complete block of size n at E

Some Interesting Properties

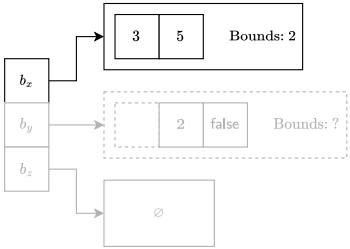
- $\vdash E + m \mapsto - \star \mathsf{bound}(E, E_1) \vdash m < E_1$
- $\vdash E \mapsto_b E_1, \ldots, E_n \star E + m \mapsto - \Rightarrow \mathsf{False}$

WISL-Specific Assertions

WISL-Specific Assertions

$$E \mapsto E \mid \mathsf{bound}(E, E) \mid E \mapsto \varnothing$$

$$b_x \mapsto_b 3, 5 \quad * \quad b_y + 1 \mapsto 2, \mathsf{false} \quad * \quad b_z \mapsto \varnothing$$

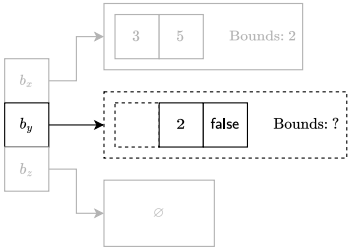


WISL-Specific Assertions

WISL-Specific Assertions

$$E \mapsto E \mid \mathsf{bound}(E, E) \mid E \mapsto \varnothing$$

$$b_x \mapsto_b 3, 5 \quad * \quad b_y + 1 \mapsto 2, \mathsf{false} \quad * \quad b_z \mapsto \varnothing$$

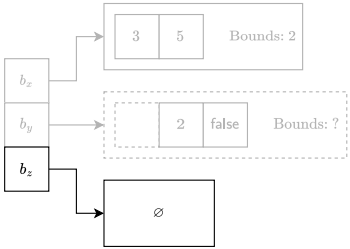


WISL-Specific Assertions

WISL-Specific Assertions

$$E \mapsto E \mid \mathsf{bound}(E, E) \mid E \mapsto \varnothing$$

$$b_x \mapsto_b 3, 5 \quad * \quad b_y + 1 \mapsto 2, \mathsf{false} \quad * \quad b_z \mapsto \varnothing$$



Predicate definitions

$$\mathsf{list}(x, \alpha) \stackrel{\text{def}}{=} (x = \texttt{null} \star \alpha = [\,]) \vee (\exists a, z, \beta.\ x \mapsto a, z \star \alpha = a{:}\beta \star \mathsf{list}(z, \beta))$$

$$\mathsf{lseg}(x, y, \alpha) \stackrel{\text{def}}{=} (x = y \star \alpha = \texttt{null}) \vee (\exists a, z, \beta.\ x \mapsto a, z \star \alpha = a{:}\beta \star \mathsf{lseg}(z, y, \beta)$$

Predicate definitions

$$\mathsf{list}(x, \alpha) \stackrel{\text{def}}{=} (x = \texttt{null} \star \alpha = [\,]) \vee (\exists a, z, \beta.\ x \mapsto_b a, z \star \alpha = a{:}\beta \star \mathsf{list}(z, \beta))$$

$$\mathsf{lseg}(x, y, \alpha) \stackrel{\text{def}}{=} (x = y \star \alpha = \texttt{null}) \vee (\exists a, z, \beta.\ x \mapsto_b a, z \star \alpha = a{:}\beta \star \mathsf{lseg}(z, y, \beta)$$

Predicate definitions

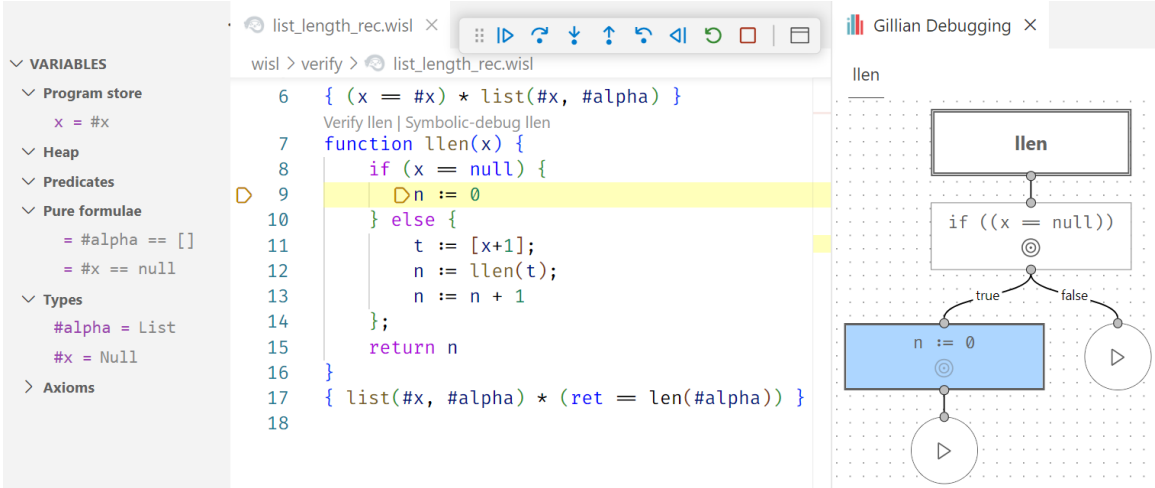
$$\mathsf{list}(x, \alpha) \stackrel{\text{def}}{=} (x = \texttt{null} \star \alpha = [\,]) \vee (\exists a, z, \beta.\ x \mapsto_b a, z \star \alpha = a{:}\beta \star \mathsf{list}(z, \beta))$$

$$\mathsf{lseg}(x, y, \alpha) \stackrel{\text{def}}{=} (x = y \star \alpha = \texttt{null}) \vee (\exists a, z, \beta.\ x \mapsto_b a, z \star \alpha = a{:}\beta \star \mathsf{lseg}(z, y, \beta))$$

```
predicate list(+x, alpha) {
  (x == null) * (alpha == []);
  (x -b> #a, #z) * (alpha == #a :: #beta) * list(#z, #beta)
}

predicate listseg(+x, y, alpha) {
  (x == y) * (alpha == []);
  (x -b> #a, #z) * (alpha == #a :: #beta) * listseg(#z, y, #beta)
}
```

Verifying WISL in Gillian



Stay tuned for the lab tomorrow!