Retargeting an Abstract Interpreter for a New Language by Partial Evaluation

Jay Lee, **Joongwon Ahn**, and Kwangkeun Yi

Jan 13th, 2026

Seoul National University

Motivation

How can we build a static analyzer for T *using what we already have*?

- $\mathbf{I}_{\mathbf{S}}^{\mathsf{T}}$: concrete interpreter for T written in **S**
- $\mathsf{I}_{\mathsf{M}}^{\sharp\mathbf{S}}$: abstract interpreter for **S**

Key Idea: Partial Evaluation

Consider partial evaluation:

$$\mathsf{I}_{\mathsf{M}}^{\mathbf{S}} : \text{interpreter for } \mathbf{S}$$

$$\mathbf{I}_{\mathbf{S}}^{\mathsf{T}} : \text{interpreter for } \mathsf{T} \text{ written in } \mathbf{S}$$

$$\Longrightarrow \mathcal{PE}(\mathsf{I}_{\mathsf{M}}^{\mathbf{S}}, \mathbf{I}_{\mathbf{S}}^{\mathsf{T}}) : \text{interpreter for } \mathsf{T}$$

Key Idea: Partial Evaluation

Consider partial evaluation:

$$\mathsf{I}_{\mathsf{M}}^{\mathbf{S}} : \text{interpreter for } \mathbf{S}$$
$$\mathbf{I}_{\mathbf{S}}^{\mathsf{T}} : \text{interpreter for } \mathsf{T} \text{ written in } \mathbf{S}$$
$$\implies \mathcal{PE}(\mathsf{I}_{\mathsf{M}}^{\mathbf{S}}, \mathbf{I}_{\mathbf{S}}^{\mathsf{T}}) : \text{interpreter for } \mathsf{T}$$

Partial evaluation of an *abstract* interpreter?

$$\mathsf{I}_{\mathsf{M}}^{\#\mathbf{S}} : \textit{abstract} \text{ interpreter for } \mathbf{S}$$
$$\mathbf{I}_{\mathbf{S}}^{\mathsf{T}} : \textit{concrete} \text{ interpreter for } \mathsf{T} \text{ written in } \mathbf{S}$$
$$\implies \mathcal{PE}(\mathsf{I}_{\mathsf{M}}^{\#\mathbf{S}}, \mathbf{I}_{\mathbf{S}}^{\mathsf{T}}) : \textit{abstract} \text{ interpreter for } \mathsf{T}?$$

Retargeting via Partial Evaluation

Main Theorem

Partial evaluation of a sound **S***-abstract interpreter with respect to a* T*-concrete interpreter produces a sound* T*-abstract interpreter.*

$$\mathsf{I}_{\mathsf{M}}^{\#\mathsf{T}} \triangleq \mathcal{PE}(\mathsf{I}_{\mathsf{M}}^{\#\mathbf{S}}, \mathbf{I}_{\mathbf{S}}^{\mathsf{T}})$$

$$(\mathsf{p}, \sigma) \in \boldsymbol{\gamma}\boldsymbol{\sigma}^{\#} \implies [\![\mathsf{p}]\!]\sigma \in \boldsymbol{\gamma}([\![\mathsf{I}_{\mathsf{M}}^{\#\mathsf{T}}]\!]\boldsymbol{\sigma}^{\#})$$

Proof.

$$[\![\mathsf{p}]\!]\sigma = [\![\mathbf{I}_{\mathbf{S}}^{\mathsf{T}}]\!](\mathsf{p}, \sigma) \in \boldsymbol{\gamma}([\![\mathsf{I}_{\mathsf{M}}^{\#\mathbf{S}}]\!](\mathbf{I}_{\mathbf{S}}^{\mathsf{T}}, \boldsymbol{\sigma}^{\#})) = \boldsymbol{\gamma}([\![\mathcal{PE}(\mathsf{I}_{\mathsf{M}}^{\#\mathbf{S}}, \mathbf{I}_{\mathbf{S}}^{\mathsf{T}})]\!]\boldsymbol{\sigma}^{\#})$$

∎

Minimal Example

Languages

S: Imperative language with fixed-size memory

$$\mathbb{E}_S\ \mathbf{e} ::= \mathbf{n} \mid *\mathbf{e} \mid \mathbf{e} + \mathbf{e} \mid \mathbf{e} \times \mathbf{e} \mid \mathbf{e} == \mathbf{e}$$

$$\mathbb{P}_S\ \mathbf{p} ::= \mathbf{skip} \mid *\mathbf{e} = \mathbf{e} \mid \mathbf{p};\mathbf{p} \mid \mathbf{if}\ (\mathbf{e})\ \{\mathbf{p}\}\ \mathbf{else}\ \{\mathbf{p}\} \mid \mathbf{while}\ (\mathbf{e})\ \{\mathbf{p}\}$$

T: Assembly-like language with single register

$$\mathbb{P}_T\ \mathtt{p} ::= \mathtt{ADD\ n} \mid \mathtt{MUL\ n} \mid \mathtt{p;\ p}$$

T-Interpreter in S

$\mathbf{I}_S^T =$

```
while ( *PC ) {
  if ( *PC == ADD ) {
    R = R + *(PC + 1)
  } else if ( *PC == MUL ) {
    R = R × *(PC + 1)
  } else {
    skip
  };
  PC = PC + 2
}
```

where

$\mathbf{R} = *0, \mathbf{PC} = *1, \mathbf{ADD} = 1, \mathbf{MUL} = 2$

$p = \texttt{ADD 3; MUL 2}$

$\mathbf{R} = 42, \mathbf{PC} = 2$

$(\sigma = [42, 2, 1, 3, 2, 2, 0])$

$\hookrightarrow \quad \mathbf{R} = 45, \mathbf{PC} = 4$

$\hookrightarrow \quad \mathbf{R} = 90, \mathbf{PC} = 6$

$(\sigma = [90, 6, 1, 3, 2, 2, 0])$

S-Abstract Interpreter

let rec $\text{evale}^\#(\mathbf{e}, \boldsymbol{\sigma}^\#) =$
match **e** *with*
$\mid \mathbf{e_1 + e_2} \to \text{evale}^\#(\mathbf{e_1}, \boldsymbol{\sigma}^\#) +^\# \text{evale}^\#(\mathbf{e_2}, \boldsymbol{\sigma}^\#)$
$\mid \mathbf{e_1 \times e_2} \to \text{evale}^\#(\mathbf{e_1}, \boldsymbol{\sigma}^\#) \times^\# \text{evale}^\#(\mathbf{e_2}, \boldsymbol{\sigma}^\#)$
...
let $\mathrm{I}_M^{\#\mathbf{S}}\text{evalp}^\#(\mathbf{p}, \boldsymbol{\sigma}^\#) =$
match **p** *with*
$\mid$ **if (e) {p$_t$} else {p$_f$}** $\to$
$\ldots\ \text{evalp}^\#(\mathbf{p_t}, \mathcal{F}^\#_{\neq 0}(\mathbf{v}^\#, \boldsymbol{\sigma}^\#)) \sqcup^\#$
$\text{evalp}^\#(\mathbf{p_f}, \mathcal{F}^\#_{=0}(\mathbf{v}^\#, \boldsymbol{\sigma}^\#))$
$\mid$ **while (e) {p$_b$}** $\to$
$\ldots\ \text{evalp}^\#(\mathbf{p}, \text{evalp}^\#(\mathbf{p_b}, \mathcal{F}^\#_{\neq 0}(\mathbf{v}^\#, \boldsymbol{\sigma}^\#))) \sqcup^\#$
$\mathcal{F}^\#_{=0}(\mathbf{v}^\#, \boldsymbol{\sigma}^\#)$
...

Retargeted T-Abstract Interpreter

$$I_M^{\#T} = \mathcal{PE}(I_M^{\#S}, \mathbf{I_S^T})$$

$$
\begin{aligned}
&\textit{let } I_M^{\#T} \text{ eval}^{\#}\ \sigma^{\#} = \\
&\quad \ldots \\
&\quad \textit{let } \sigma_7^{\#} = \\
&\qquad \sigma_3^{\#}[0^{\#} \mapsto^{\#} \sigma_3^{\#}[0^{\#}] +^{\#} \sigma_3^{\#}[\sigma_3^{\#}[1^{\#}] +^{\#} 1^{\#}]] \sqcup^{\#} \\
&\qquad \sigma_5^{\#}[0^{\#} \mapsto^{\#} \sigma_5^{\#}[0^{\#}] \times^{\#} \sigma_5^{\#}[\sigma_5^{\#}[1^{\#}] +^{\#} 1^{\#}]] \sqcup^{\#} \sigma_6^{\#} \\
&\quad \textit{in} \\
&\quad \sigma_1^{\#} \sqcup^{\#} \text{eval}^{\#}\ (\sigma_7^{\#}[1^{\#} \mapsto^{\#} \sigma_7^{\#}[1^{\#}] +^{\#} 2^{\#}])
\end{aligned}
$$

$\mathbf{I_S^T} =$

```
while ( *PC ) {
  if ( *PC == ADD ) {
    R = R + *(PC + 1)
  } else if ( *PC == MUL ) {
    R = R × *(PC + 1)
  } else {
    skip
  };
  PC = PC + 2
}
```

Retargeted T-Abstract Interpreter

let $I^{\#T}_M$ eval$^\#$ $\sigma^\#$ =

...

let $\sigma^\#_7$ =

$\sigma^\#_3[0^\# \mapsto^\# \sigma^\#_3[0^\#] +^\# \sigma^\#_3[\sigma^\#_3[1^\#] +^\# 1^\#]] \sqcup^\#$

$\sigma^\#_5[0^\# \mapsto^\# \sigma^\#_5[0^\#] \times^\# \sigma^\#_5[\sigma^\#_5[1^\#] +^\# 1^\#]] \sqcup^\# \sigma^\#_6$

in

$\sigma^\#_1 \sqcup^\#$ eval$^\#$ $(\sigma^\#_7[1^\# \mapsto^\# \sigma^\#_7[1^\#] +^\# 2^\#])$

I^T_S =

```
while ( *PC ) {
  if ( *PC == ADD ) {
►   R = R + *(PC + 1)
  } else if ( *PC == MUL ) {
►   R = R × *(PC + 1)
  } else {
    skip
  };
  PC = PC + 2
}
```

Abstract operations are reused!

Retargeted T-Abstract Interpreter

$let\ I_M^{\#T}\ eval^{\#}\ \sigma^{\#} =$

$\quad \ldots$

$\quad let\ \sigma_7^{\#} =$

$\quad\quad \sigma_3^{\#}[0^{\#} \mapsto^{\#} \sigma_3^{\#}[0^{\#}] +^{\#} \sigma_3^{\#}[\sigma_3^{\#}[1^{\#}] +^{\#} 1^{\#}]] \sqcup^{\#}$

$\quad\quad \sigma_5^{\#}[0^{\#} \mapsto^{\#} \sigma_5^{\#}[0^{\#}] \times^{\#} \sigma_5^{\#}[\sigma_5^{\#}[1^{\#}] +^{\#} 1^{\#}]] \sqcup^{\#} \sigma_6^{\#}$

$\quad in$

$\quad \sigma_1^{\#} \sqcup^{\#} eval^{\#}\ (\sigma_7^{\#}[1^{\#} \mapsto^{\#} \sigma_7^{\#}[1^{\#}] +^{\#} 2^{\#}])$

$I_S^T =$

```
while ( *PC ) {
  if ( *PC == ADD ) {
    R = R + *(PC + 1)
  } else if ( *PC == MUL ) {
    R = R × *(PC + 1)
  } else {
    skip
  };
  PC = PC + 2
}
```

Interpreter structure is preserved!

Future Work

Future Work: More Case Studies

- More rich **S** language
  - functions, heap, algebraic data types, etc.
- More practical T languages
  - e. g. mini-JS, mini-Python
- Domain-specific T languages
  - e. g. React Hooks

Future Work: Delivery of Precision

- Relation between the precisions of **S**- and T-analyzers
  - e. g., $\mathbf{I}_{\mathbf{S}}^{\mathsf{T}}$ uses $\mathbf{PC} = *\mathbf{1}$ for program counter:

```
I_S^T =
  while ( *PC ) {
    if ( *PC == ADD ) {
      R = R + *(PC + 1)
    } else if ( *PC == MUL ) {
      R = R × *(PC + 1)
    } else { skip };
    PC = PC + 2
  }
```

  - $*\mathbf{1}$-sensitive $\mathrm{I}_{\mathrm{M}}^{\#\mathbf{S}}$ yields flow-sensitive $\mathrm{I}_{\mathrm{M}}^{\#\mathsf{T}}$
- What is the necessary **S**-sensitivities for a desired T-sensitivity?

Conclusion

Retarget abstract interpreters
using partial evaluation.