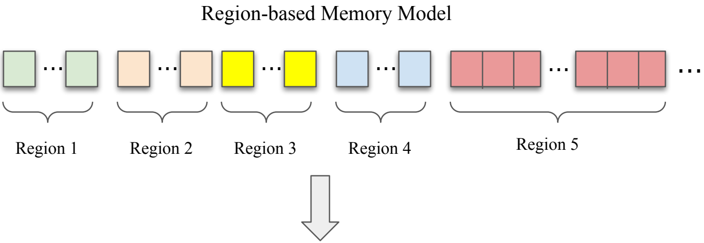
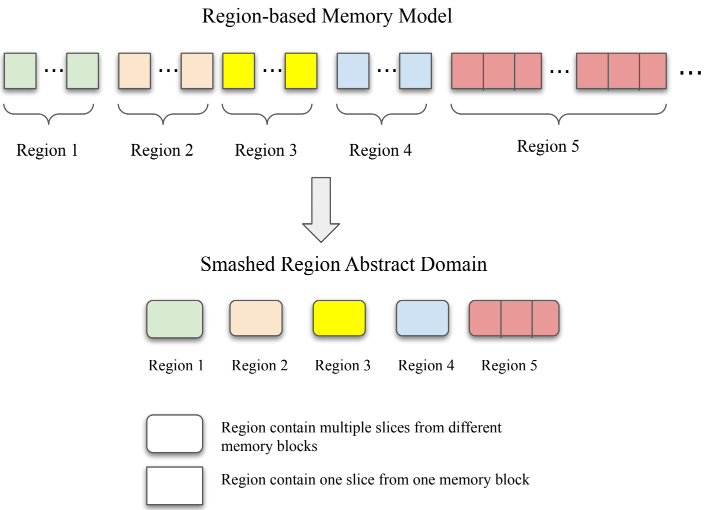
Region Abstract Domains



Region Abstract Domains



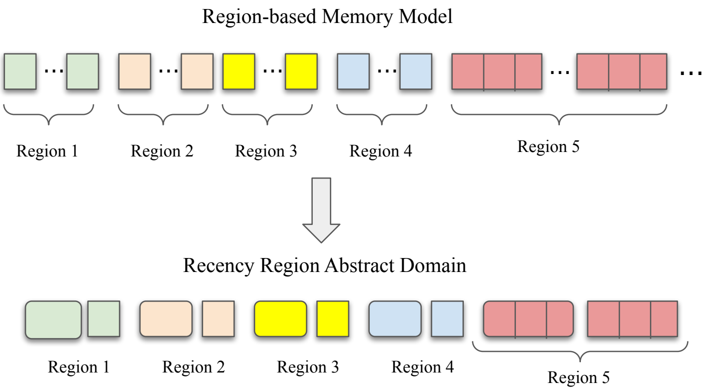
Region-based Memory Model

Smashed Region Abstract Domain

Region contain multiple slices from different memory blocks

Region contain one slice from one memory block

Region Abstract Domains



Smashed Region Abstract Domain

Abstract state $\sigma^{\#} = (\mathit{base}, \mathit{countAddr}, \mathit{init})$

1. *base* $\in$ Base: array domain parameterized by a numerical domain over integer and Boolean variables
2. *countAddr* $\in$ SmallRangeEnv : $v_{rgn} \mapsto [0|1|0{-}1|1^+|0^+]$
3. *init* $\in$ BoolEnv : $v_{rgn} \mapsto$ [True|False|Any]

- *base* abstracts *numEnv*, *refEnv*, and *rgnEnv*:
  - A reference is mapped to an integer variable
  - Each region content mapped to scalar/array variable
- *countAddr* abstracts *rgnAddrs*: how many addresses per region
  - if 0–1 then strong updates on the scalar or array
    - *base* can still decide a weak update if needed
  - if 1^+ or 0^+ then weak updates on the scalar or array
- *memObjs* is ignored
- *init*: whether a region has been written to

$[\![Stmt_{rgn}]\!]^{\#}(\sigma^{\#})$

$[\![\texttt{initrgn}(rgn)]\!]^{\#}(\sigma^{\#})$

```
match $\sigma^{\#}$ with $(base, countAddr, init)$ $\rightarrow$
  if $countAddr(rgn) \sqsubseteq_{\mathsf{SmallRange}} 1^{+}$ then $\bot$
  else
    let $init' = init[rgn \mapsto \mathsf{False}]$ in
    let $countAddr' = countAddr[rgn \mapsto 0]$ in
    $(base, countAddr', init')$
```

$[\![ref := \texttt{makeref}(rgn, n)]\!]^{\#}(\sigma^{\#})$

```
match $\sigma^{\#}$ with $(base, countAddr, init)$ $\rightarrow$
  let $countAddr' =$
    $countAddr[rgn \mapsto countAddr(rgn) +^{SmallRange} 1]$ in
  $(base, countAddr', init)$
```

$[\![Stmt_{rgn}]\!]^{\#}(\sigma^{\#})$

$$[\![(rgn_2, ref_2) := \texttt{gepref}(rgn_1, ref_1, n)]\!]^{\#}(\sigma^{\#})$$

```
match $\sigma^{\#}$ with $(base, countAddr, init)$ $\rightarrow$
    let $countAddr' =$
    if $rgn_1 \neq rgn_2$ or $[\![ref_2 \neq ref_1 + n]\!]^{\#_{\mathsf{Base}}}(base) \neq \bot_{\mathsf{Base}}$ then
        $countAddr[rgn_2 \mapsto countAddr(rgn_2) +^{SmallRange} 1]$
    else
        $countAddr$
    in
    let $base' = [\![ref_2 := ref_1 + n]\!]^{\#_{\mathsf{Base}}}(base)$ in
    $(base', countAddr', init)$
```

$[\![Stmt_{rgn}]\!]^{\#}(\sigma^{\#})$

$[\![lhs := \texttt{loadref}(rgn, ref)]\!]^{\#}(\sigma^{\#})$

```
match $\sigma^{\#}$ with $(base, countAddr, init) \rightarrow$
   if $[\![ ref \neq 0 ]\!]^{\#_{\mathsf{Base}}}(base) = \bot_{\mathsf{Base}}$ or
      $init(rgn) = \mathsf{False}$ then $\bot$
   else
      let $base' =$
         if $countAddr(rgn) \sqsubseteq_{\mathsf{SmallRange}} 0{-}1$ then
            $[\![ lhs := rgn ]\!]^{\#_{\mathsf{Base}}}(base)$
         else
            let $\langle base'', rgn_{copy} \rangle = base.\mathrm{expand}(rgn)$
            $[\![ lhs := rgn_{copy} ]\!]^{\#_{\mathsf{Base}}}(base'')$
      in
      $(base', countAddr, init)$
```

$[\![Stmt_{rgn}]\!]^{\#}(\sigma^{\#})$

```
$[\![\texttt{storeref}(rgn, ref, val)]\!]^{\#}(\sigma^{\#})$

match $\sigma^{\#}$ with $(base, countAddr, init) \rightarrow$
  if $[\![ref \neq 0]\!]^{\#_{\text{Base}}}(base) = \bot_{\text{Base}}$ then $\bot$
  else
    let $base' =$
    if $init(rng) = \text{False or } countAddr(rgn) \sqsubseteq_{\text{SmallRange}} 0\text{--}1$ then
      $[\![rgn := val]\!]^{\#_{\text{Base}}}(base)$
    else
      $base \sqcup_{base} [\![rgn := val]\!]^{\#_{\text{Base}}}(base)$
    in
    let $init' = init[rgn \mapsto \text{Any}]$ in
    $(base', countAddr, init')$
```

$[\![Stmt_{rgn}]\!]^{\#}(\sigma^{\#})$

$[\![x := \texttt{reftoint}(rgn, ref)]\!]^{\#}(\sigma^{\#})$

$$\begin{array}{l}\texttt{match } \sigma^{\#} \texttt{ with } (base, countAddr, init) \ \rightarrow \\ \quad ([\![x := ref]\!]^{\#_{\text{Base}}}(base), countAddr, init)\end{array}$$

$[\![ref := \texttt{inttoref}(rgn, x)]\!]^{\#}(\sigma^{\#})$

$$\begin{array}{l}\texttt{match } \sigma^{\#} \texttt{ with } (base, countAddr, init) \ \rightarrow \\ \quad ([\![ref := x]\!]^{\#_{\text{Base}}}(base), \\ \quad\ \ countAddr[rgn \mapsto countAddr(rgn) +^{SmallRange} 1], \\ \quad\ \ init)\end{array}$$

$[\![Stmt_{rgn}]\!]^{\#}(\sigma^{\#})$

$[\![rgn' := \texttt{copyrgn}(rgn)]\!]^{\#}(\sigma^{\#})$

```
match $\sigma^{\#}$ with $(base, countAddr, init)$ $\rightarrow$
    let $\langle base'', rgn_{copy} \rangle := base.\mathrm{expand}(rgn)$ in
    let $base' = [\![ rgn' := rgn_{copy} ]\!]^{\#_{\mathrm{Base}}}(base'')$ in
    let $countAddr' = countAddr[rgn' \mapsto countAddr(rgn)]$ in
    let $init' = init[rgn' \mapsto init(rgn)]$ in
    $(base', countAddr', init')$
```