Algorithm Fully symbolic memory: naive implementation

Immutable objects:

M $:= \{(e, v)\}$
e := an expression over symbols and concrete values
v := an expression over symbols and concrete values
π := set of assumptions
$equiv(e, \widetilde{e}, \pi) := (e \neq \widetilde{e} \wedge \pi) == UNSAT$
$disjoint(e, \widetilde{e}, \pi) := (e = \widetilde{e} \wedge \pi) == UNSAT$
$intersect(e, \widetilde{e}, \pi) := (e = \widetilde{e} \wedge \pi) == SAT$

function _STORE(e, v):
  $M' \leftarrow M$
  for $(\widetilde{e}, \widetilde{v}) \in M$ **do**
    if $disjoint(\widetilde{e}, e, \pi)$ **then**
      continue
    else if $equiv(\widetilde{e}, e, \pi)$ **then**
      $M' \leftarrow M'|_{\widetilde{e} \mapsto v}$
      $flag = true$
    else
      $M' \leftarrow M'|_{\widetilde{e} \mapsto ite(\widetilde{e} = e \wedge \pi, v, \widetilde{v})}$
    end if
  end for
  if $\neg flag$ **then**
    $M' \leftarrow M'|_{e \mapsto v}$
  end if
  $M \leftarrow M'$
end function

function LOAD(e, $size$):
  $v = \perp$
  for $(\widetilde{e}, \widetilde{v}) \in M$ **do**
    if $intersect(\widetilde{e}, e, \pi)$ **then**
      $v = ite(\widetilde{e} = e \wedge \pi, \widetilde{v}, v)$
    end if
  end for
  return v
end function

function _LOAD(e):
  $v = \perp$
  for $(\widetilde{e}, \widetilde{v}) \in M$ **do**
    if $intersect(\widetilde{e}, e, \pi)$ **then**
      $v = ite(\widetilde{e} = e \wedge \pi, \widetilde{v}, v)$
    end if
  end for
  return v
end function