Algorithm Fully symbolic memory: naive implementation

Immutable objects:

t_{pos}	:= timestamp (initially, set to 0)
t_{neg}	:= timestamp (initially, set to 0)
e	:= an expression over symbols and concrete values
v	:= a 1-byte expression over symbols and concrete values
V	:= ordered set of v
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, $size$):
    for $k = 0$ to $size - 1$ **do**
        _STORE($e + k$, v_k)
    end for
end function

function _STORE(e, v):
    $a = min(e)$
    $b = max(e)$
    $t_{pos} \leftarrow t_{pos} + 1$
    INSERT($(a, b), (e, v, t_{pos}, true)$)
end function

function INSERT($(a, b), (e, v, t, \delta)$):
    for $x \in$ SEARCH(a, b): **do**
        if $equiv\_sup(e, x(e))$ **then**
            $x(v) \leftarrow v$
            $x(t) \leftarrow t$
            $x(\delta) \leftarrow \delta$
            return
        end if
    end for
    M_s.ADD($(a, b), (e, v, t, \delta)$)
end function

function SEARCH((a, b))):
    return $\{x \in M_s \mid x(a, b) \cap [a, b] \neq \varnothing\}$
end function

function LOAD(e, $size$):
  $V = \langle\rangle$
  for $k = 0$ to $size - 1$ **do**
    $v_k = \_\mathrm{LOAD}(e + k)$
    $V = V \cdot v_k$
  end for
  return V
end function

function _LOAD(e):
  $a = min(e)$
  $b = max(e)$
  $P \leftarrow \{(\widetilde{e}, \widetilde{v}, \widetilde{t}, \widetilde{\delta}) \mid (\widetilde{e}, \widetilde{v}, \widetilde{t}, \widetilde{\delta}) \in \mathrm{SEARCH}(a, b)\}$
  $P' \leftarrow \mathrm{SORT\_BY\_INCREASING\_TIMESTAMP}(P)$
  $v \leftarrow \perp$
  $t_{neg} \leftarrow t_{neg} - 1$
  $M_s.\mathrm{ADD}((a, b), (e, v, t_{neg}, true))$ ▷ implicit store
  for $(\widetilde{e}, \widetilde{v}, \widetilde{t}, \widetilde{\delta}) \in P'$ **do**
    $v \leftarrow ite(e = \widetilde{e} \ \wedge \ \widetilde{\delta}, \widetilde{v}, v)$
  end for
  return v
end function

function MERGE($(S_1, \delta_1), (S_2, \delta_2), S_{ancestor}$): ▷ $S_1 := self$
  $t_{pos}^{anc} = S_{ancestor}.t_{pos}$
  $t_{neg}^{anc} = S_{ancestor}.t_{neg}$
  $M_s \leftarrow S_{ancestor}.M_s$
  for $\{x \in S_1.M_s \mid (x(t) > 0 \wedge x(t) > t_{pos}^{anc}) \vee (x(t) < 0 \wedge x(t) < t_{neg}^{anc})\}$ **do**
    $x(\delta) = x(\delta) \wedge \delta_1$
    $M_s.\mathrm{ADD}((x(a), x(b)), x))$
  end for
  for $\{x \in S_2.M_s \mid (x(t) > 0 \wedge x(t) > t_{pos}^{anc}) \vee (x(t) < 0 \wedge x(t) < t_{neg}^{anc})\}$ **do**
    $x(\delta) = x(\delta) \wedge \delta_2$
    $M_s.\mathrm{ADD}((x(a), x(b)), x))$
  end for
  $t_{pos} = max(S_1.t_{pos}, S_2.t_{pos})$
  $t_{neg} = min(S_1.t_{neg}, S_2.t_{neg})$
end function