Algorithm Fully symbolic memory: naive implementation

Immutable objects:

M_c	$:= \{(e_c, v)\}$
e_c	:= an expression over concrete values
M_s	$:= \{(e, v)\}$
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):
    $min_e = min(e)$
    $max_e = max(e)$
    $flag = false$
    $constant = false$
    ${M_c}' \leftarrow M_c$
    ${M_s}' \leftarrow M_s$
    if $min_e == max_e$ **then**
        $constant = true$
        ${M_c}' \leftarrow {M_c}'|_{e \mapsto v}$
    end if
    for $(\widetilde{e}, \widetilde{v}) \in M_s$ **do**
        if $disjoint(\widetilde{e}, e, \pi)$ **then**
            continue
        else if $equiv(\widetilde{e}, e, \pi)$ **then**
            if $constant$ **then**
                ${M_s}' \leftarrow {M_s}' - (\widetilde{e}, v)$
            else
                ${M_s}' \leftarrow {M_s}'|_{\widetilde{e} \mapsto v}$
            end if
            $flag = true$
        else
            ${M_s}' \leftarrow {M_s}'|_{\widetilde{e} \mapsto ite(\widetilde{e}=e, v, \widetilde{v})}$
        end if
    end for
    if $\neg flag \wedge \neg constant$ **then**
        ${M_s}' \leftarrow {M_s}'|_{e \mapsto v}$
    end if
    $M_c \leftarrow {M_c}'$
    $M_s \leftarrow {M_s}'$
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($e$):
    $v = \perp$
    $min_e = min(e)$
    $max_e = max(e)$
    if $min_e == max_e$ then
        if $(min_e, \widetilde{v}) \in M_c$ then
            $v = \widetilde{v}$
        end if
    else
        for $e_c = min_e$ to $max_e$ do
            if $(e_c, \widetilde{v}) \in M_c$ then
                $v = ite(e_c = e, \widetilde{v}, v)$
            end if
        end for
    end if
    for $(\widetilde{e}, \widetilde{v}) \in M_s$ do
        if $intersect(\widetilde{e}, e, \pi)$ then
            $v = ite(\widetilde{e} = e, \widetilde{v}, v)$
        end if
    end for
    return $v$
end function
```

```
function MERGE((M^1_c, M^1_s, \pi^1), (M^2_c, M^2_s, \pi^2), ..., (M^n_s, M^n_s, \pi^n)):
    M_c \leftarrow \{\},  M_s \leftarrow \{\}
    \pi \leftarrow \bigvee_i pi^i

    for i \in \{1, n\} do
        for (e, v) \in M^i_c do
            if (e, v') \in M_c then
                M_c \leftarrow M_c|_{e \mapsto ite(\pi^i, v, v')}
            else
                M_c \leftarrow M_c|_{e \mapsto ite(\pi^i, v, \bot)}
            end if
        end for

        for (e, v) \in M^i_s do
            flag = false
            for (\widetilde{e}, \widetilde{v}) \in M_s do
                if disjoint(\widetilde{e}, e, \pi) then
                    continue
                else if equiv(\widetilde{e}, e, \pi) then
                    M_s \leftarrow M_s|_{\widetilde{e} \mapsto ite(\pi^i, v, \widetilde{v})}
                    flag = true
                else
                    M_s \leftarrow M_s|_{\widetilde{e} \mapsto ite(\pi^i \wedge e = \widetilde{e}, v, \widetilde{v})}
                    flag = true
                end if
            end for
            if \neg flag then
                M_s \leftarrow M_s|_{e \mapsto ite(\pi^i, v, \bot)}
            end if
        end for
    end for

    return (M_c, M_s, \pi)
end function
```