## Algorithm Fully symbolic memory: naive implementation

```
Immutable objects:
     M_c
                              := \{(e_c, v)\}
                              := an expression over concrete values
      e_c
     M_s
                              := \{(e, v)\}
                             := an expression over symbols and concrete values
                              := a 1-byte expression over symbols and concrete values
      V
                             := ordered set of v
                              := 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
 1: function STORE(e, v, size):
 2:
          for k = 0 to size - 1 do
 3:
               \_STORE(e+k, v_k)
 4:
          end for
 5: end function
 1: function \_STORE(e, V):
 2:
          min_e = min(e)
 3:
          max_e = max(e)
 4:
          flag = false
 5:
          constant = false
          {M_c}' \leftarrow M_c
 6:
         {M_s}' \leftarrow M_s
 7:
         if min_e == max_e then
 8:
 9:
              constant=true \\
              {M_c}' \leftarrow {M_c}'|_{e \mapsto v}
10:
         end if
11:
12:
          for (\widetilde{e}, \widetilde{v}) \in M_s do
              if disjoint(\widetilde{e}, e, \pi) then
13:
14:
                   continue
15:
               else if equiv(\tilde{e}, e, \pi) then
                   if constant then
16:
                        {M_s}' \leftarrow {M_s}' - (\widetilde{e}, v)
17:
18:
                       {M_s}' \leftarrow {M_s}'|_{\widetilde{e} \mapsto v}
19:
20:
                   end if
                   flag = true
21:
22:
                   {M_s}' \leftarrow {M_s}'|_{\widetilde{e} \mapsto ite(\widetilde{e}=e,v,\widetilde{v})}
23:
24:
              end if
25:
          end for
          if \neg flag \land \neg constant then
26:
              {M_s}' \leftarrow {M_s}'|_{e \mapsto v}
27:
          end if
28:
          M_c \leftarrow {M_c}'
29:
          M_s \leftarrow M_s'
30:
31: end function
```

```
1: function LOAD(e, size):
 2:
         V = \langle \rangle
 3:
         for k = 0 to size - 1 do
              v_k = \bot \text{LOAD}(e+k)
 4:
              V = V \cdot v_k
 5:
 6:
         end for
 7:
         \mathbf{return}\ V
 8: end function
 1: function \bot OAD(e):
 2:
         v = \bot
         min_e = min(e)
 3:
 4:
         max_e = max(e)
         if min_e == max_e then
 5:
 6:
              if (min_e, \widetilde{v}) \in M_c then
                  v = \widetilde{v}
 7:
             end if
 8:
 9:
         else
10:
              for e_c = min_e to max_e do
                  if (e_c, \widetilde{v}) \in M_c then
11:
12:
                       v = ite(e_c = e, \widetilde{v}, v)
                  end if
13:
14:
              end for
         end if
15:
16:
         for (\widetilde{e}, \widetilde{v}) \in M_s do
              if intersect(\widetilde{e}, e, \pi) then
17:
18:
                  v = ite(\widetilde{e} = e, \widetilde{v}, v)
              end if
19:
20:
         end for
21:
         \mathbf{return}\ v
22: end function
```

```
1: 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
 3:
 4:
           for i \in \{1, n\} do
 5:
                for (e, v) \in M^i{}_c do
 6:
                      if (e, v') \in M_c then
 7:
                           M_c \leftarrow M_c|_{e \mapsto ite(\pi^i, v, v')}
 8:
 9:
                           M_c \leftarrow M_c|_{e \mapsto ite(\pi^i, v, \perp)}
10:
                      end if
11:
                 end for
12:
13:
14:
                 for (e, v) \in M^i{}_s do
                      flag = false
15:
                      for (\widetilde{e}, \widetilde{v}) \in M_s do
16:
                           if disjoint(\tilde{e}, e, \pi) then
17:
18:
                                 continue
                           else if equiv(\widetilde{e}, e, \pi) then
19:
20:
                                 M_s \leftarrow M_s|_{\widetilde{e} \mapsto ite(\pi^i, v, \widetilde{v})}
21:
                                 flag = true
22:
                           else
23:
                                 M_s \leftarrow M_s|_{\widetilde{e} \mapsto ite(\pi^i \land e = \widetilde{e}, v, \widetilde{v})}
24:
                                 flag = true
25:
                           end if
                      end for
26:
                      if \neg flag then
27:
                           M_s \leftarrow M_s|_{e \mapsto ite(\pi^i, v, \perp)}
28:
29:
                      end if
                 end for
30:
31:
           end for
32:
33:
           return (M_c, M_s, \pi)
34: end function
```