

---

**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  $v$   
 $\pi$  := set of assumptions  
 $equiv(e, \tilde{e}, \pi)$  :=  $(e \neq \tilde{e} \wedge \pi) == UNSAT$   
 $disjoint(e, \tilde{e}, \pi)$  :=  $(e = \tilde{e} \wedge \pi) == UNSAT$   
 $intersect(e, \tilde{e}, \pi)$  :=  $(e = \tilde{e} \wedge \pi) == SAT$

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

```
1: function INSERT( $((a, b), (e, v, t, \delta))$ ):  
2:   for  $x \in \text{SEARCH}(a, b)$ : do  
3:     if  $equiv\_sup(e, x(e))$  then  
4:        $x(v) \leftarrow v$   
5:        $x(t) \leftarrow t$   
6:        $x(\delta) \leftarrow \delta$   
7:       return  
8:     end if  
9:   end for  
10:   $M_s.ADD((a, b), (e, v, t, \delta))$   
11: end function
```

```
1: function SEARCH( $a, b$ ):  
2:   return  $\{x \in M_s \mid x(a, b) \cap [a, b] \neq \emptyset\}$   
3: end function
```

---

---

```

1: function LOAD( $e, size$ ):
2:    $V = \langle \rangle$ 
3:   for  $k = 0$  to  $size - 1$  do
4:      $v_k = \text{LOAD}(e + k)$ 
5:      $V = V \cdot v_k$ 
6:   end for
7:   return  $V$ 
8: end function

```

```

1: function _LOAD( $e$ ):
2:    $a = \min(e)$ 
3:    $b = \max(e)$ 
4:    $P \leftarrow \{(\tilde{e}, \tilde{v}, \tilde{t}, \tilde{\delta}) \mid (\tilde{e}, \tilde{v}, \tilde{t}, \tilde{\delta}) \in \text{SEARCH}(a, b)\}$ 
5:    $P' \leftarrow \text{SORT\_BY\_INCREASING\_TIMESTAMP}(P)$ 
6:    $v \leftarrow \perp$ 
7:    $t_{neg} \leftarrow t_{neg} - 1$ 
8:    $M_s.\text{ADD}((a, b), (e, v, t_{neg}, \text{true}))$ 
9:   for  $(\tilde{e}, \tilde{v}, \tilde{t}, \tilde{\delta}) \in P'$  do
10:     $v \leftarrow \text{ite}(e = \tilde{e} \wedge \tilde{\delta}, \tilde{v}, v)$ 
11:   end for
12:   return  $v$ 
13: end function

```

▷ implicit store

```

1: function MERGE( $(S_1, \delta_1), (S_2, \delta_2), S_{\text{ancestor}}$ ):
2:    $t_{pos}^{anc} = S_{\text{ancestor}}.t_{pos}$ 
3:    $t_{neg}^{anc} = S_{\text{ancestor}}.t_{neg}$ 
4:    $M_s \leftarrow S_{\text{ancestor}}.M_s$ 
5:   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
6:      $x(\delta) = x(\delta) \wedge \delta_1$ 
7:      $M_s.\text{ADD}((x(a), x(b)), x)$ 
8:   end for
9:   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
10:     $x(\delta) = x(\delta) \wedge \delta_2$ 
11:     $M_s.\text{ADD}((x(a), x(b)), x)$ 
12:   end for
13:    $t_{pos} = \max(S_1.t_{pos}, S_2.t_{pos})$ 
14:    $t_{neg} = \min(S_1.t_{neg}, S_2.t_{neg})$ 
15: end function

```

---

▷  $S_1 := self$