
Algorithm Fully symbolic memory: naive implementation

Immutable objects:

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
 $equiv(e, \tilde{e}, \pi)$:= $(e \neq \tilde{e} \wedge \pi) == UNSAT$
 $disjoint(e, \tilde{e}, \pi)$:= $(e = \tilde{e} \wedge \pi) == UNSAT$
 $intersect(e, \tilde{e}, \pi)$:= $(e = \tilde{e} \wedge \pi) == SAT$

```
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:    $t \leftarrow t + 1$   
5:   INSERT( $((a, b), (e, v, t, true))$ )  
6: end function
```

```
1: function INSERT( $((a, b), (e, v, t, \delta))$ ):  
2:   for  $x \in SEARCH(a, b)$ : 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:  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  $\text{\_LOAD}(e)$ :
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:    $\text{ADD}((a, b), (e, v, 0, true))$ 
8:   for  $(\tilde{e}, \tilde{v}, \tilde{t}, \tilde{\delta}) \in P'$  do
9:      $v \leftarrow \text{ite}(e = \tilde{e} \wedge \tilde{\delta}, \tilde{v}, v)$ 
10:  end for
11:  return  $v$ 
12: end function

```

▷ implicit store

```

1: function MERGE( $(S_1, \delta_1), (S_2, \delta_2)$ ):
2:    $M_s \leftarrow \text{INTERVAL\_TREE}(\ )$ 
3:   for  $x \in S_1.M_s$  do
4:      $x(\delta) = x(\delta) \wedge \delta_1$ 
5:      $\text{ADD}((x(a), x(b)), x)$ 
6:   end for
7:   for  $x \in S_2.M_s$  do
8:      $x(\delta) = x(\delta) \wedge \delta_2$ 
9:      $\text{ADD}((x(a), x(b)), x)$ 
10:  end for
11: end function

```
