

## lib/memory-range/memory\_unittest.ath

```

1  ## Test Memory-theory.
2
3  load "memory"
4
5  #####
6  # Test proofs with standard operators:
7
8  define Mem-ops := no-renaming
9
10 assert (theory-axioms Memory.theory)
11
12 pick-any M:(Memory 'S) a:(Memory.Loc 'S) x:'S
13         b:(Memory.Loc 'S)
14   assume (a = b)
15         (!chain [(M Memory.\ a Memory.<- x) Memory.at b] = x
16                [(get-property Memory.assign.equal Mem-ops Memory.theory)]])
17
18 #!/map-method (method (n) (!property-test n Mem-ops Memory.theory))
19 # (property-names Memory-theorems) make-conjunction)
20
21 open Memory
22
23 pick-any M:(Memory 'S) a:(Memory.Loc 'S) x:'S
24         b:(Memory.Loc 'S)
25   assume (a = b)
26         (!chain [(M \ a <- x) at b] = x
27                [(get-property Memory.assign.equal Mem-ops theory)]])
28
29 (!prove-property Double-assign Mem-ops theory)
30 (!prove-property Direct-double-assign Mem-ops theory)
31 (!prove-property Self-assign Mem-ops theory)
32 (!prove-property Direct-self-assign Mem-ops theory)
33 (!prove-property Double-swap Mem-ops theory)
34 (!prove-property Direct-double-swap Mem-ops theory)
35
36 #####
37 # Test proofs with a different set of operators:
38
39 declare At: (S) [(Memory S) (Memory.Loc S)] -> S
40
41 declare Assign: (S) [(Memory.Loc S) S] -> (Memory.Change S)
42
43 declare Swap: (S) [(Memory.Loc S) (Memory.Loc S)] -> (Memory.Change S)
44
45 define Mem-ops := (renaming |{at := At, <- := Assign, swap := Swap}|)
46
47 assert (Mem-ops (theory-axioms theory))
48
49 (!prove-property Double-assign Mem-ops theory)
50 (!prove-property Direct-double-assign Mem-ops theory)
51 (!prove-property Self-assign Mem-ops theory)
52 (!prove-property Direct-self-assign Mem-ops theory)
53 (!prove-property Double-swap Mem-ops theory)
54 (!prove-property Direct-double-swap Mem-ops theory)

```