lib/basic/options.ath

```
datatype (Option T) := NONE | (SOME option-val:T)

(set-precedence SOME 110)

module Options {

define [NONE SOME] := [NONE SOME]

assert opt-axioms := (datatype-axioms "Option")

conclude option-lemma-1 :=
           (forall ?x ?y . ?x = SOME ?y ==> ?x =/= NONE)
  pick-any x y
    assume hyp := (x = SOME y)
      (!chain-last [true ==> (NONE =/= SOME y)  [opt-axioms]
                         ==> (NONE =/= x)       [hyp]
                         ==> (x =/= NONE)       [sym]])

conclude option-lemma-2 :=
           (forall ?x . ?x =/= NONE ==> exists ?y . ?x = SOME ?y)
  pick-any x
    assume hyp := (x =/= NONE)
      (!chain-last
       [true ==> (x = NONE | exists ?y . x = SOME ?y) [opt-axioms]
             ==> (exists ?y . x = SOME ?y)            [(dsyl with hyp)]])

conclude option-lemma-2-conv :=
           (forall ?x . (forall ?y . ?x =/= SOME ?y) ==> ?x = NONE)
  pick-any x
    assume hyp := (forall ?y . x =/= SOME ?y)
      (!chain-last
         [hyp ==> (~ exists ?y . x = SOME ?y)  [qn]
              ==> (x = NONE)                   [option-lemma-2]])


conclude option-lemma-3 :=
           (forall ?x . ?x = NONE ==> ~ exists ?y . ?x = SOME ?y)
  pick-any x
    assume hyp := (x = NONE)
      (!by-contradiction  (~ exists ?y . x = SOME ?y)
       assume hyp' := (exists ?y . x = SOME ?y)
         pick-witness y for hyp'
           (!absurd
            (!chain-last
             [(x = SOME y) ==> (NONE = SOME y) [hyp]])
            (!chain-last
             [true ==> (NONE =/= SOME y) [opt-axioms]])))

conclude option-lemma-4 :=
           (forall ?x ?y . ?x = NONE ==> ?x =/= SOME ?y)
  pick-any x y
    assume hyp := (x = NONE)
      (!by-contradiction (x =/= SOME y)
        assume (x = SOME y)
          (!absurd
           (!chain-last
            [(x = NONE) ==> (NONE = x) [sym]
                        ==> (NONE = SOME y) [(x = SOME y)]])
           (!uspec (forall ?y . NONE =/= SOME ?y) y)))


conclude option-lemma-5 :=
  (forall ?x ?y ?z . ?x = SOME ?y & ?y =/= ?z ==> ?x =/= SOME ?z)
 pick-any x y z
  assume h := (x = SOME y & y =/= z)
    (!chain-last [h ==> (y =/= z)           [right-and]
                 ==> (SOME y =/= SOME z) [opt-axioms]
```

```
68                     ==> (x =/= SOME z)       [h]])
69
70  define opt-lemmas :=
71    [option-lemma-1 option-lemma-2 option-lemma-2-conv option-lemma-3 option-lemma-4 option-lemma-5]
72
73  define option-results := (join opt-axioms opt-lemmas)
74
75  }
76
77  open Options
```