lib/main/group_unittest.ath

```
## Test group concepts and theorems

load "group"

module Test {

open Group

define Group0 := no-renaming

assert (Group0 (theory-axioms ’Group))

##############################################################################
#
#(set-debug-mode "rewriting")

(test-proofs [Group.left-inverse Group.double-negation
              Group.unique-negation Group.neg-plus] ’Group)

##############################################################################
## Create some dummy symbols to plug into Group theory

declare pluss: (T) [T T] -> T

declare negs: (T) [T] -> T

declare zeros: (T) [] -> T

# Show that the proofs work with this different set of symbols.

define Group1 := (renaming |{Group.+ := pluss, Group.U- := negs,  Group.<0> := zeros}|)

assert (Group1 (theory-axioms ’Group))

# This time, don’t prove ’Left-Inverse before testing ’Double-Negation;
# it should thus be proved on the fly.

(!prove-property double-negation Group1 Group.theory)

(!prove-property unique-negation Group1 Group.theory)

(!prove-property neg-plus Group1 Group.theory)

# Although it was proved during the proof of Double-Negation, it wasn’t
# left in the assumption base, so !property has to reprove it.
(!prove-property left-inverse Group1 Group.theory)

##############################################################################

declare plus’: (T) [T T] -> T

declare neg’: (T) [T] ->  T

declare zero’: (T) [] -> T

declare minus’: (T) [T T] -> T

define Abelian-Group0 := (renaming |{Group.+ := plus’, Group.U- := neg’,
                                     Group.<0> := zero’, Group.- := minus’}|)

assert (Abelian-Group0 (theory-axioms Abelian-Group.theory))

(!prove-property left-inverse Abelian-Group0 Abelian-Group.theory)

(!prove-property double-negation Abelian-Group0 Abelian-Group.theory)

(!prove-property unique-negation Abelian-Group0 Abelian-Group.theory)

(!prove-property neg-plus Abelian-Group0 Abelian-Group.theory)

(!prove-property Abelian-Group.neg-plus Abelian-Group0 Abelian-Group.theory)

################################################################################

define MG0 := no-renaming

assert (MG0 (theory-axioms MG.theory))

(!prove-property left-inverse MG0 MG.theory)

(!prove-property double-negation MG0 MG.theory)

(!prove-property unique-negation MG0 MG.theory)

(!prove-property neg-plus MG0 MG.theory)

################################################################################

declare times1: (T) [T T] -> T

declare one1: (T) [] -> T

declare inv1: (T) [T] -> T

declare div1: (T) [T T] -> T

define MG1 :=
   (renaming |{MSG.* := times1, MM.<1> := one1, MG.inv := inv1,  MG./ := div1}|)

assert (MG1 (theory-axioms MG.theory))

(!prove-property left-inverse MG1 MG.theory)

(!prove-property double-negation MG1 MG.theory)

(!prove-property unique-negation MG1 MG.theory)

(!prove-property neg-plus MG1 MG.theory)

} # Test
```