/* This is a Nitpick counterpart to the specification used in chapter 4 of Diller's book, Z: an Introduction to Formal Methods */

[Person, Phone, Report]

PhoneDB = /* see Diller p. 46 */
[ members: set Person
  telephones: Person <-> Phone
  Not_a_member: Report
  Entry_already_exists: Report
  Okay: Report
  |
  dom (telephones) <= members ]

AddEntry (name: Person; newnumber: Phone) = /* see p. 53 */
[ PhoneDB
  | name in members
  not ({name -> newnumber} <= telephones) /* recast slightly */
  telephones' = telephones U {name -> newnumber}
  members' = members ]

NotMember (name: Person; rep: Report) =
[ const PhoneDB
  | name not in members
  rep = Not_a_member ]

EntryAlreadyExists (name: Person; newnum: Phone; rep: Report) =
[ const PhoneDB
  | {name -> newnum} <= telephones
  rep = Entry_already_exists ]

Success (rep: Report) =
[ const PhoneDB
  | rep = Okay ]

DoAddEntry (name: Person; newnum: Phone; rep: Report) =
[ PhoneDB
  | AddEntry (name, newnum) and Success (rep) or NotMember (name, rep) or EntryAlreadyExists (name, newnum, rep) ]

FindPhones (name: Person; numbers: set Phone) = /* see p. 57 */
[ const PhoneDB
  | name in dom (telephones)
  numbers = telephones.{name} ]
FindNames (number: Phone; names: set Person) = /* see p. 58 */
[ const PhoneDB
| number in ran(telephones)
| names = telephones~.{number}
]

RemoveEntry(oldnumber: Phone; name: Person) = /* see p. 59 */
[ PhoneDB
| {name -> oldnumber} <= telephones
| telephones' = telephones \ {name -> oldnumber}
| members' = members
]

AddMember(name: Person) = /* see p. 60 */
[ PhoneDB
| name not in members
| members' = members U {name}
| telephones' = telephones
]

RemoveMember(name: Person) = /* see p. 62 */
[ PhoneDB
| name in members
| members' = members \ {name}
| telephones' = {name} <= telephones
]

InitPhoneDB = /* see p. 66 */
[ PhoneDB
| members = {}
| telephones = {}
]

Claim1(n: Person; p: Phone) ::
/* FindPhones yields ALL phones in DB */
[ PhoneDB
| x: set Phone
| {n in members and {n -> p} <= telephones and FindPhones(n,x) => p in x}
]

Claim2(n: Person; p: Phone) ::
/* FindPhones yields ONLY phones in DB */
[ PhoneDB
| x: set Phone
| {n in members and FindPhones(n,x) and p in x} => {n -> p} <= telephones
]
Nitpick Checker 0.87 (Beta)

Parsing "Macintosh HD:Word 5.1:classes:22c181:diller2.np"...

Generating 3x3 isomorph free relations
Generated 36 3x3 relations
Checking schema "DoAddEntry"
Creating Executable ...
Finished creating executable
Elapsed time 0:00.11
Isomorph elimination on
Derived variable detection on
Short circuiting on
Restricting Phone to 3 elements \{ ph0, ph1, ph2 \}
Restricting Person to 3 elements \{ pe0, pe1, pe2 \}
Restricting Report to 3 elements \{ r0, r1, r2 \}

Schema "DoAddEntry" is executed by case:
Entry_already_exists: Report is r0
Entry_already_exists': Report is r0
members: Set Person is empty
members': Set Person is empty
name: Person is pe0
newnum: Phone is ph0
Not_a_member: Report is r0
Not_a_member': Report is r0
Okay: Report is r0
Okay': Report is r0
rep: Report is r0
telephones: Person <-> Phone is empty
telephones': Person <-> Phone is empty

Finished checking schema "DoAddEntry"
After checking 1 cases of 3.30226e+11 possible
1 unlabeled cases examined
0 unlabeled cases skipped due to short-circuiting
0.0% of the entire case space checked
1 example found
Executed 63 instructions checking schema
Elapsed time 0:00.26

Checking schema "RemoveEntry"
Creating Executable ...
Finished creating executable
Elapsed time 0:00.01
Isomorph elimination on
Derived variable detection on
Short circuiting on
Restricting Phone to 3 elements \{ ph0, ph1, ph2 \}
Restricting Person to 3 elements \{ pe0, pe1, pe2 \}
Restricting Report to 3 elements \{ r0, r1, r2 \}

Schema "RemoveEntry" is executed by case:
members: Set Person is \{ pe0 \}
members': Set Person is \{ pe0 \}
name: Person is pe0
oldnumber: Phone is ph2
telephones: Person <-> Phone is
{ pe0 -> {ph2 } }

telephones': Person <-> Phone is empty

Finished checking schema "RemoveEntry"
After checking 85 cases of 36864 possible
38 unlabeled cases examined
0 unlabeled cases skipped due to short-circuiting
26.4% of the entire case space checked
1 example found
Executed 22 instructions checking schema
Elapsed time 0:00.20

Checking claim "Claim1"
Creating Executable ...
Finished creating executable
Elapsed time 0:00.01
Isomorph elimination on
Derived variable detection on
Short circuiting on
Restricting Phone to 3 elements { ph0, ph1, ph2 }
Restricting Person to 3 elements { pe0, pe1, pe2 }
Restricting Report to 3 elements { r0, r1, r2 }
...
Completed checking entire case space

Finished checking claim "Claim1"
After checking 1887 cases of 7.96262e+06 possible
294 unlabeled cases examined
282 unlabeled cases skipped due to short-circuiting
No counter examples were found
Executed 50871 instructions checking claim
Elapsed time 0:00.26

Checking claim "Claim2"
Creating Executable ...
Finished creating executable
Elapsed time 0:00.01
Isomorph elimination on
Derived variable detection on
Short circuiting on
Restricting Phone to 3 elements { ph0, ph1, ph2 }
Restricting Person to 3 elements { pe0, pe1, pe2 }
Restricting Report to 3 elements { r0, r1, r2 }
...
Completed checking entire case space

Finished checking claim "Claim2"
After checking 1523 cases of 7.96262e+06 possible
276 unlabeled cases examined
300 unlabeled cases skipped due to short-circuiting
No counter examples were found
Executed 38441 instructions checking claim
Elapsed time 0:00.23