```
(FILECREATED "19-Aug-80 11:15:50" <AFFIRM>SUFFICIENT..14 29849

     changes to:  SUFFICIENTCOMS

     previous date: "14-Mar-80 15:58:16" <AFFIRM>SUFFICIENT..13)


(PRETTYCOMPRINT SUFFICIENTCOMS)

(RPAQQ SUFFICIENTCOMS [(FNS * SUFFICIENTFNS)
        (BLOCKS (SUFFICIENTBLOCK AllLevelMember AllSafe ArgMemberCheck BSafe BSafeII BuildMissingLHS CSafe
                                 Dok DokII ErrorHeading FindFunction FindInterfaceOperation FindOperation
                                 FiveSafe FourSafe GetInterfaces LeftHandSidesCheck MaxList
                                 Missing\ExtraAxiomsMessage Nest Partition SafeExtensionAxioms
                                 SafeOutputAxioms SufficientlyComplete (ENTRIES SufficientlyComplete)
                                 (GLOBALVARS AxiomList Cset CsetI Eset EsetI IOperations Operations Oset
                                         OsetI TOI lhs])

(RPAQQ SUFFICIENTFNS (AllLevelMember AllSafe ArgMemberCheck BSafe BSafeII BuildMissingLHS CSafe Dok DokII
                      ErrorHeading FindFunction FindInterfaceOperation FindOperation FiveSafe
                      FourSafe GetInterfaces LeftHandSidesCheck MaxList
                      Missing\ExtraAxiomsMessage Nest Partition SafeExtensionAxioms
                      SafeOutputAxioms SufficientlyComplete))
(DEFINEQ
                                                                                        1

(AllLevelMember
  [LAMBDA (atm lst)                                          (* R.Bates " 8-NOV-78 14:07")
    (EDITFINDP lst atm])
                                                                                        2

(AllSafe
  [LAMBDA (lst)                                              (* R.Bates "23-Jan-80 12:58")

        (* * (This function returns True if all members of list lst are True, otherwise it returns NIL.))


    (NIL ~MEMB lst])
                                                                                        3

(ArgMemberCheck
  [LAMBDA (arg lst)                                          (* D.Baker " 7-Sep-79 20:15")

        (* (This function determines if all free variables in arg are members of lst.) (Calls : AllLevelMember,
        ArgMemberCheck) (Uses : TOI))


    (if (ATOM arg)
        then (if (NUMBERP arg) or (STRINGP arg) or (GETPROP arg 'IsConstant)
                 then T
               else (AllLevelMember arg lst) and (FNTYP arg)=NIL)
      elseif arg:Arguments=NIL
        then (Extension arg)
             ~=TOI
      else (for x in arg:Arguments always (ArgMemberCheck x lst])
                                                                                        4

(BSafe
  [LAMBDA (axiom origaxiom)                                  (* D.Baker "15-Oct-79 19:45")

        (* * (This function returns T if the axiom is BSafe, as defined in J.V. Guttag's thesis, otherwise, it returns
        NIL. The argument origaxiom is needed for the error message in the case that a theorem derived from the axiom
        is in error.) (Uses:Oset,Cset) (Calls : Nest, BSafeII, ErrorHeading))
    (PROG (lhs rhs nestcondition axiomlistcondition safeflag)
          (lhs←axiom:LHS)
          (rhs←axiom:RHS)
          (nestcondition←((Nest rhs < ! Oset ! Cset>)
             lt
             (Nest lhs < ! Oset ! Cset>)))
          (axiomlistcondition←(BSafeII rhs rhs < ! Oset ! Cset>))
          (safeflag←(nestcondition or axiomlistcondition))
          (if ~safeflag
              then (ErrorHeading axiom origaxiom)
                   (printout NIL "  (1)  Nest(" # (PrettyPrint rhs T)
                             ") < Nest(" # (PrettyPrint lhs T)
                             ")" T "OR" T "  (2)  There exists axioms" , # (PrettyPrint rhs T)
                             " =z1," T "          z1=z2 ... zn=zn+1 such that" T "        Nest(" #
                             (PrettyPrint lhs T)
                             ") < Nest(zn+1)" T))
          (RETURN safeflag])
```

5

```
(BSafeII
  [LAMBDA (z0 zn lst)                                          (* D.Baker " 7-Sep-79 20:17")

          (* (This function determines if there exists axioms z0=z1,z1=z2,..., zn=zn+1 where Nest
          (z0) <Nest (zn+1).) (Uses : AxiomList) (Calls : Nest, BSafeII))


    (PROG (zn1)
          (if (MEMBER zn (for x in AxiomList collect x:LHS))
              then zn1←NIL
                   (for x in AxiomList do (if zn=x:LHS
                                              then zn1←x:RHS))
                   (if (Nest zn1 lst) lt (Nest z0 lst)
                       then (RETURN T)
                     else (RETURN (BSafeII z0 zn1)))
            else (RETURN NIL])
```

6

```
(BuildMissingLHS
  [LAMBDA (function constructor)                               (* D.Baker " 9-NOV-78 16:26")

          (* (This function builds the left hand side of any axiom missing in the axiomatization for use in error
          messages.) (Calls : Rules, FindInterfaceOperation) (Uses : TOI))


    (PROG (constructorterm lhs params flag)
          (constructorterm← <constructor ! (fetch Arguments
                                              of (fetch Expression
                                                    of (fetch RHS
                                                          of (CAR (Rules (FindInterfaceOperation constructor)
                                                                         <'interface >]
                           >)
          [params←(fetch Arguments of (fetch LHS of (CAR (Rules (FindInterfaceOperation function)
                                                                <'interface >]
          (flag←T)
          [params←(for x in old params join (if x
                                                then (if x:Type=TOI and flag
                                                         then flag←NIL
                                                              <constructorterm>
                                                       else <x:Arg1>]
          (RETURN <function ! params>])
```

7

```
(CSafe
  [LAMBDA (axiom origaxiom)                                    (* D.Baker "27-Sep-79 14:56")

          (* * (This function returns T if axiom is CSafe, as defined in J.V. Guttag's thesis, otherwise, it returns
        NIL. The argument origaxiom is needed for the error message in the case that a theorem derived from the axiom
        is in error.) (Calls: Nest, Rules, FindInterfaceOperation, BSafe) (Uses: Oset, Cset))


    (PROG (lhs boolean thenclause elseclause equalop safeflag rule)
          (if axiom:RHS:Operator=IFOP
              then lhs←axiom:LHS
                   boolean←axiom:RHS:Test
                   thenclause←axiom:RHS:ThenPart
                   elseclause←axiom:RHS:ElsePart
                   equalop←axiom:Operator
                   rule←(Rules (FindInterfaceOperation axiom:LHS:Operator)
                               <'interface >)
                   safeflag←((BSafe <equalop lhs thenclause> origaxiom)
                      and (BSafe <equalop lhs elseclause> origaxiom)
                      and ((Nest boolean < ! Oset ! Cset>)
                            !!
                            (Nest lhs < ! Oset ! Cset>) or (BSafe <equalop lhs boolean> origaxiom)
                             and (EQUAL rule:1:RHS:Type 'Boolean)))
            else safeflag←(BSafe axiom origaxiom))
          (RETURN safeflag])
```

8

```
(Dok
  [LAMBDA (axiom origaxiom)                                             (* R.Bates "23-Jan-80 13:40")

        (* (This function returns T if axiom is Dok, as defined in J.V. Guttag's thesis, otherwise it returns NIL.
        The argument origaxiom is needed for the error message in the case that a theorem derived from the axiom is in
        error.) (Calls: AllLevelMember, FindFunction, Occurences, DokII, ErrorHeading))


    (PROG (lhs rhs safeflag place innerop arglist numberargs)
          (lhs←axiom:LHS)
          (rhs←axiom:RHS)
          (place←(FindFunction lhs:Arguments))
          (innerop←(FNTH lhs:Arguments place):1:1)
          (safeflag←(~(AllLevelMember lhs:Operator rhs) or (FNTYP innerop) and (ARGLIST innerop)
                                                                              :1~='TooManyArguments
               and (DokII lhs rhs place)))
          (if ~safeflag
              then (ErrorHeading axiom origaxiom)
                   (printout NIL T "  (1)  There are no occurences of ")
                   (PrettyPrint lhs:Operator T)
                   (printout NIL " in ")
                   (PrettyPrint rhs T)
                   (printout NIL T "OR " T "  (2)  ")
                   (PrettyPrint innerop T)
                   (printout NIL " is not a constant, " T "       and for every subterm w of ")
                   (PrettyPrint rhs T)
                   (printout NIL T "       if w is of the form ")
                   (PrettyPrint lhs:Operator T)
                   arglist←(COPY lhs:Arguments)
                   (FNTH arglist place):1←NIL
                   numberargs←(FLENGTH (ARGLIST lhs:Operator))
                   (printout NIL # (MAPRINT (for x from 1 to numberargs collect x)
                                            NIL '%(v '%) ", v")
                             T "       then" ,)
                   (PRIN1 "v")
                   (PRIN1 place)
                   (printout NIL , "is a free variable contained in" , #
                             (MAPRINT (for x in (FNTH lhs:Arguments place):1:Arguments
                                         collect (Shorten x))
                                      NIL '[ '] ", "))
                   (printout NIL T "       and" , #
                             (MAPRINT (for x from 1 to numberargs collect x unless x=place)
                                      NIL "v" NIL ", v")
                             , "are free variables contained in" , # (MAPRINT (for x in arglist
                                                                                  collect (Shorten x)
                                                                                  unless (EQUAL x NIL))
                                                                               NIL '[ '] ", ")))
        (RETURN safeflag])
```

9

```
(DokII
  [LAMBDA (lhs rhs place)                                    (* R.Bates "23-Jan-80 13:40")

        (* (For lhs=f(g(x*),y*) and for all subterms w of rhs. If w is of the form f(u,v*) then this function
        determines whether or not u is a free variable in x* and all members of v* are free variables in y*.)
        (Calls : AllLevelMember, AllSafe, ArgMemberCheck, DokII))


    (PROG (safeflag lhsarglist rhsarglist)
        (safeflag←T)
        (lhsarglist←(COPY lhs:Arguments))
        ((FNTH lhsarglist place):1←NIL)
        (lhsarglist←(for y in lhsarglist unless (EQUAL y NIL) collect y))
        [if lhs:Operator=rhs:Operator
            then rhsarglist←(COPY rhs:Arguments)
                 (FNTH rhsarglist place):1←NIL
                 safeflag←((AllLevelMember (FNTH rhs:Arguments place):1 (FNTH lhs:Arguments place)
                                           :1:Arguments)
                   and (GETPROP (FNTH rhs:Arguments place):1 'IsConstant)=NIL
                   and (for x in rhsarglist unless (EQUAL x NIL) always (ArgMemberCheck x lhsarglist]
        [safeflag←(safeflag and (AllSafe (for x in rhs:Arguments collect (if (ATOM x)
                                                                              then T
                                                                            else (DokII lhs x place]
        (RETURN safeflag])
```

10

```
(ErrorHeading
  [LAMBDA (axiom origaxiom)                                  (* D.Thompson " 9-Mar-80 18:31")
    (if ~(EQUAL (RemoveIfs origaxiom)
                axiom)
        then (printout NIL T "The following theorem is in error")
             (PrettyPrint axiom)
             (printout NIL T "The axiom from which it was derived follows")
             (PrettyPrint origaxiom)
      else (printout NIL T "The following axiom is in error")
           (PrettyPrint axiom))
    (Heading

"One of the following conditions must be satisfied for the axiomatization
to be recognized to be sufficiently complete."])
```

11

```
(FindFunction
  [LAMBDA (expr)                                             (* D.Baker "18-JAN-79 18:03")
    (if (ATOM expr)
        then 0
      else 1+(for x in expr count T until (LISTP x) and (MEMBER x:Operator Operations])
```

12

```
(FindInterfaceOperation
  [LAMBDA (oper)                                             (* D.Baker " 7-Sep-79 20:19")

        (* (Given the extended name of an operation of the form oper\TOI this function returns the extended name of
        the form oper\TOI\INTERFACE.)(Calls : Shorten) (Uses : IOperations))


    (for (x (shortoper ←(Shorten oper))) in IOperations thereis (Shorten x)=shortoper])
```

13

```
(FindOperation
  [LAMBDA (ioper)                                              (* R.Bates " 8-NOV-78 15:37")

        (* (Given the extended name of the form oper\TOI\INTERFACE of an operation this function returns the extended
        name of the form oper\TOI.) (Uses : Operations) (Calls : Shorten))


    (for (x (shortioper ←(Shorten ioper))) in Operations thereis (Shorten x)=shortioper])
```

14

```
(FiveSafe
  [LAMBDA (axiom origaxiom)                                    (* D.Baker "27-Sep-79 15:07")

        (* (This function returns T if axiom is FiveSafe, as defined in J.V. Guttag's thesis, otherwise it returns
        NIL. The argument origaxiom is needed for the error message in the case that a theorem derived from the axiom
        is in error.) (Calls : FiveSafe, Dok))


    (PROG (lhs boolean thenclause elseclause equalop safeflag)
          (if axiom:RHS:Operator=IFOP
              then lhs←axiom:LHS
                   boolean←axiom:RHS:Test
                   thenclause←axiom:RHS:ThenPart
                   elseclause←axiom:RHS:ElsePart
                   equalop←axiom:Operator
                   safeflag←((FiveSafe <equalop lhs thenclause> origaxiom)
                      and (FiveSafe <equalop lhs elseclause> origaxiom))
            else safeflag←(Dok axiom origaxiom))
          (RETURN safeflag])
```

15

```
(FourSafe
  [LAMBDA (axiom origaxiom)                                    (* D.Baker "27-Sep-79 15:11")

        (* * (This function returns T if axiom is FourSafe, as defined in J.V. Guttag's thesis, otherwise it returns
        NIL. The argument origaxiom is needed for the error message in the case that a theorem derived from the axiom
        is in error.) (Uses:Cset,Oset) (Calls : Nest, CSafe, FourSafe, Rules, FindInterfaceOperation,))


    (PROG (lhs boolean thenclause elseclause equalop safeflag rule)
          (if axiom:RHS:Operator=IFOP
              then lhs←axiom:LHS
                   boolean←axiom:RHS:Test
                   thenclause←axiom:RHS:ThenPart
                   elseclause←axiom:RHS:ElsePart
                   equalop←axiom:Operator
                   rule←(Rules (FindInterfaceOperation axiom:LHS:Operator)
                               <'interface >)
                   safeflag←((FourSafe <equalop lhs thenclause> origaxiom)
                      and (FourSafe <equalop lhs elseclause> origaxiom)
                      and ((Nest boolean < ! Oset ! Cset>)
                           it
                           (Nest lhs < ! Oset ! Cset>) or (FourSafe <equalop lhs boolean> origaxiom)
                            and (EQUAL rule:1:RHS:Type 'Boolean)))
            else safeflag←(CSafe axiom origaxiom))
          (RETURN safeflag])
```

16

```
(GetInterfaces
  [LAMBDA (AxiomList)                                          (* D.Baker "11-Nov-79 18:01")

        (* (This function assigns a value to Operations which is a list of the operations of the TOI that should be
        specified in the axiomatization. Members of Operations have the form name\TOI. IOperations has the same
        members, but they are of the form name\TOI\INTERFACE.) (sets: Operations, IOperations))
  (PROG (Ops IOps definesAndSchemas definesAndSchemasOps axiomOps)
        (Operations←(Functions TOI))
        (IOperations←(Functions TOI T))
        (Ops←(COPY Operations))
        (IOps←(COPY IOperations))
        (for x on Ops do (x:1←(Shorten x:1)))
        (for x on IOps do (x:1←(Shorten x:1)))
        (Ops←(INTERSECTION Ops IOps))
        (for x in Operations do (if ~(MEMBER (Shorten x)
                                             Ops)
                                    then Operations←(REMOVE x Operations)))
                                                  (* An operation that is specified via a
                                                  Define or a Schema command does not belong in
                                                  Operations.)
        (definesAndSchemas←(for x in Operations join (Rules x <'defn. 'schema >)))
        (definesAndSchemasOps←(for x in definesAndSchemas collect x:LHS:Operator))
        (axiomOps←(for x in AxiomList join <x:LHS:Operator (for y in x:LHS:Arguments
                                                              join (if (LISTP y)
                                                                       then y:Operator))
                                            >))
        (axiomOps←(INTERSECTION axiomOps axiomOps))
        (definesAndSchemasOps←(LDIFFERENCE definesAndSchemasOps (INTERSECTION axiomOps definesAndSchemasOps)
                                          ))
        (Operations←(LDIFFERENCE Operations definesAndSchemasOps))
        [IOps←(COPY (for x in Operations collect (Shorten x]
        (Ops←(INTERSECTION Ops IOps))
        (for x in Operations do (if ~(MEMBER (Shorten x)
                                             Ops)
                                    then Operations←(REMOVE x Operations)))
        (for x in IOperations do (if ~(MEMBER (Shorten x)
                                              Ops)
                                     then IOperations←(REMOVE x IOperations])
```

17

```
(LeftHandSidesCheck
  [LAMBDA NIL                                                    (* R.Bates "14-Dec-79 14:43")
    (PROG (axiomlhs missinglhs extensions outputs specialfns zparams lhslist)

          (* (This function verifies that each output function composed with each constructor, and each extension
          function composed with each constructor is the left hand side of an axiom. It also allows as left hand sides
          extension functions not composed with constructors.) (Uses : AxiomList, TOI, Cset, Eset, Oset)
          (Calls: Heading, Missing\ExtraAxiomsMessage, Functions, Rules, FindInterfaceOperation BuildMissingLHS))


          (axiomlhs←(for x in AxiomList collect x:LHS))          (* Build lists of the compositions.)
          (extensions←(for x in Eset join (for y in Cset collect <x y>)))
          (outputs←(for x in Oset join (for y in Cset collect <x y>)))
                                                                (* Determine if each extension function is
                                                                composed with each constructor function)
          (for x in extensions do (for z in axiomlhs do (if x:Operator=z:Operator
                                                            and (for y in z thereis y:Operator=x:Arg1
                                                                                     or y=x:Arg1)
                                                            then axiomlhs←(REMOVE z axiomlhs)
                                                                 extensions←(REMOVE x extensions]
                                                                (* Determine if each output function is
                                                                composed with each constructor function)
          (for x in outputs do (for z in axiomlhs do (if x:Operator=z:Operator
                                                         and (for y in z thereis y:Operator=x:Arg1
                                                                                  or y=x:Arg1)
                                                         then axiomlhs←(REMOVE z axiomlhs)
                                                              outputs←(REMOVE x outputs]

          (* In some cases, the constructor will not be the first argument of the output or extension function.
          Check for this case.)


          (for x in < ! outputs ! extensions> do (lhslist←(for y in AxiomList collect y:LHS
                                                           when y:LHS:Operator=x:Operator
                                                                and (for z in y:LHS
                                                                       thereis z:Operator=x:Arg1
```

```
                                                                                                       or z=x:Arg1)))
                                                            (if (FLENGTH lhslist)
                                                                  ~=0
                                                                  then (if (MEMBER x:Operator Eset)
                                                                            then extensions←(REMOVE x extensions)
                                                                         else outputs←(REMOVE x outputs]
             (for z in axiomlhs do (if (MEMBER z:Operator Oset) and (for (y (functions ←(Functions TOI)))
                                                                                   in z:ARGS
                                                                                   always ~(MEMBER y:Operator functions))
                                             then axiomlhs←(REMOVE z axiomlhs)
                                                   (for x in outputs do (if x:Operator=z:Operator
                                                                                  then outputs←(REMOVE x outputs]
             (specialfns←NIL)
             (for z in axiomlhs do [zparams←(fetch Arguments
                                                       of (fetch LHS of (CAR (Rules (FindInterfaceOperation z:Operator)
                                                                                                  <'interface >]
                                        (if (for y in zparams always y:Arg2~=TOI)
                                             then axiomlhs←(REMOVE z axiomlhs)
                                                   specialfns← < ! specialfns z:Operator>))
             (missinglhs←(for z in < ! extensions ! outputs> collect (BuildMissingLHS z:Operator z:Arg1)
                               unless z:Operator MEMB specialfns))
             (Missing\ExtraAxiomsMessage missinglhs NIL)
             (RETURN missinglhs=NIL])
```

18

```
(MaxList
  [LAMBDA (lst)                                                                  (* D.Baker "11-Sep-79 17:32")

             (* * (This function returns the maximum of the numbers in lst.))


    (APPLY (FUNCTION IMAX)
              lst])
```

19

```
(Missing\ExtraAxiomsMessage
  [LAMBDA (missinglhs extralhs)                                                  (* D.Baker "11-Nov-79 18:05")
    (if missinglhs
        then (printout NIL T "Each of the following should be a left hand side of an axiom, but is not" T)
               (for x in missinglhs do (PrettyPrint x)))
    (if extralhs
        then (printout NIL T "Each of the following is an excess axiom" T)
               (for x in extralhs do (for y in AxiomList do (if y:LHS=x
                                                                                then (PrettyPrint y])
```

20

```
(Nest
  [LAMBDA (exp set)                                                              (* R.Bates "14-Dec-79 14:42")

             (* * (This function returns the maximum level of nesting in exp of members of set.) (Calls: MaxList,Nest))


    (if (NLISTP exp)
        then (if exp MEMB set
                      then 1
                   else 0)
      elseif exp:Operator ~MEMB set
        then (if exp:Arguments
                      then (MaxList (for x in exp:Arguments collect (Nest x set)))
                   else 0)
      elseif exp:Arguments
        then 1+(MaxList (for x in exp:Arguments collect (Nest x set)))
      else 1])
```

21

```
(Partition
  [LAMBDA (IOperations)                                    (* D.Thompson " 5-Feb-80 19:06")
    (PROG (yorigname xorigname)

          (* (This function partitions the members of IOperations into 3 subsets : OsetI (Output Functions), CsetI
          (Constructor Functions), EsetI (Extension Functions).) (Uses : TOI, AxiomList) (Calls : Shorten, Rules)
          (Updates : CsetI,EsetI,OsetI))


          (CsetI←NIL)
          (EsetI←NIL)
          (OsetI←NIL)
          (for x in IOperations do (if (Rules x <'interface >):1:RHS:Type=TOI
                                       then xorigname←(Shorten x)
                                            (for y in AxiomList do yorigname←(Shorten y:LHS:Operator)
                                               until yorigname=xorigname)
                                            (if yorigname=xorigname
                                                then EsetI← <x ! EsetI>
                                              else CsetI← <x ! CsetI>)
                                     else OsetI← <x ! OsetI>))
          (AFFIRMMAPRINT "The constructors are" (for x in CsetI collect (Shorten x))
                         T T)
          (AFFIRMMAPRINT "The extension functions are" (for x in EsetI collect (Shorten x))
                         T T)
          (AFFIRMMAPRINT "The output functions are" (for x in OsetI collect (Shorten x))
                         T T])
```

22

```
(SafeExtensionAxioms
  [LAMBDA (Eterms)                                         (* R.Bates "14-Dec-79 14:42")

          (* (This function returns T if all axioms whose left hand sides are members of Eterms are FiveSafe.)
          (Calls AllSafe,FiveSafe) (Uses: AxiomList, Eset))


    (AllSafe (for x in AxiomList collect (if x:LHS:Operator MEMB Eset
                                              then (FiveSafe x x)
                                            else T])
```

23

```
(SafeOutputAxioms
  [LAMBDA (Oterms)                                         (* R.Bates "23-Jan-80 12:58")

          (* (This function returns T if all axioms whose left hand sides are members of Oterms are FourSafe.)
          (Uses: Oset, TOI, AxiomList) Calls : FourSafe, AllSafe, AllLevelMember)


    (AllSafe (for x in AxiomList collect (if x:LHS:Operator MEMB Oset
                                              then (if [AllSafe (for y in x:LHS:Arguments
                                                                   collect (if (ATOM y)
                                                                               then y ~MEMB (Functions TOI]
                                                       then ~(AllLevelMember x:LHS:Operator x:RHS)
                                                     else (FourSafe x x))
                                            else T])
```

24

```
(SufficientlyComplete
  [LAMBDA (TOI)                                            (* D.Baker "12-Nov-79 13:32")

          (* (This is the function which determines if the axiomatization of TOI is sufficiently complete.
          For a definition of sufficient completeness, see J.V. Guttag's thesis) (Calls : Axioms, Functions, Shorten,
          Partition, FindOperation, LeftHandSidesCheck, SafeOutputAxioms, SafeExtensionAxioms))


    (PROG (Cset CsetI Eset EsetI Oset OsetI OkToContinue Operations IOperations AxiomList)
          (printout NIL T "The type of interest is" , TOI "." T T)
          (OkToContinue←(if (Axioms TOI)=NIL
                            then NIL
                          else T))
          (if OkToContinue
              then

          (* Before checking for sufficient completeness, some information about the type of interest, TOI, must be
          gathered. Operations will be a list of the functions defined for TOI; each element of Operations will be of
          the form name\TOI. IOperations will also be a list of the functions defined for TOI; each element will be of
          the form name\TOI\INTERFACE.)


                    AxiomList←(Axioms TOI)

          (* * The function GetInterfaces set Operations and Ioperations.)


                    (GetInterfaces AxiomList)

          (* Partition the functions of TOI into disjoint subsets of constructor, extension, and output functions,
          CsetI, EsetI, and OsetI respectively. The function Partition sets CSetI, EsetI and OSetI)


                    (Partition IOperations)

          (* Strip the \INTERFACE off of each of the elements in CsetI, EsetI, and OsetI to get Cset, Eset, and Oset.)


                    Cset←(for x in CsetI collect (FindOperation x))
                    Eset←(for x in EsetI collect (FindOperation x))
                    Oset←(for x in OsetI collect (FindOperation x))    (* Determine if all necessary axioms are
                                                                        present by checking the left hand side of
                                                                        each axiom.)
                    OkToContinue←(LeftHandSidesCheck)                    (* Determine if each axiom is constructed in
                                                                        such a way that the axiomatizaton is
                                                                        sufficiently complete.)
                    (if OkToContinue
                        then OkToContinue←(SafeOutputAxioms))
                    (if OkToContinue
                        then OkToContinue←(SafeExtensionAxioms))
                    (if OkToContinue
                        then (printout NIL T T "This axiomatization is sufficiently complete." T)
                      else (printout NIL T T
                                 "The system cannot determine if this axiomatization is sufficiently complete."
                                     T))
            else (printout NIL "There is no axiomatization for type" , TOI "." T))
          (RETURN OkToContinue])
)
(DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY
(BLOCK: SUFFICIENTBLOCK AllLevelMember AllSafe ArgMemberCheck BSafe BSafeII BuildMissingLHS CSafe Dok DokII
        ErrorHeading FindFunction FindInterfaceOperation FindOperation FiveSafe FourSafe GetInterfaces
        LeftHandSidesCheck MaxList Missing\ExtraAxiomsMessage Nost Partition SafeExtensionAxioms
        SafeOutputAxioms SufficientlyComplete (ENTRIES SufficientlyComplete)
        (GLOBALVARS AxiomList Cset CsetI Eset EsetI IOperations Operations Oset OsetI TOI lhs))
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (1148 28512 (AllLevelMember 1160 . 1320) (AllSafe 1324 . 1588) (ArgMemberCheck 1592 . 2311) (
BSafe 2315 . 3611) (BSafeII 3615 . 4363) (BuildMissingLHS 4367 . 5482) (CSafe 5486 . 6779) (Dok 6783 . 9263) (
DokII 9267 . 10712) (ErrorHeading 10716 . 11385) (FindFunction 11389 . 11694) (FindInterfaceOperation 11698 .
12165) (FindOperation 12169 . 12628) (FiveSafe 12632 . 13558) (FourSafe 13562 . 14880) (GetInterfaces 14884 .
17397) (LeftHandSidesCheck 17401 . 21384) (MaxList 21388 . 21568) (Missing\ExtraAxiomsMessage 21572 . 22183) (
Nost 22187 . 22913) (Partition 22917 . 24286) (SafeExtensionAxioms 24290 . 24781) (SafeOutputAxioms 24785 .
25558) (SufficientlyComplete 25562 . 28508)))))
STOP
```