<AFFIRM>VCGEN..12 17-Jun-81 16:42:09

```
(FILECREATED "31-Mar-81 20:26:37" <AFFIRM>VCGEN..12 19179

     previous date: "28-Mar-81 17:13:35" <AFFIRM>VCGEN..11)


(PRETTYCOMPRINT VCGENCOMS)

(RPAQQ VCGENCOMS ((FNS * VCGENFNS)
        (RECORDS * VCGENRECORDS)
        (BLOCKS (VCGENBLOCK AltersList AltersListHelper AssignmentSubst AssignmentSubst1
                            ConcurrentAssignmentSubst GenNewSymbols GetLastAssertion GetLoopAssertion
                            MakeBaseName MakeCallRelationName MakeCaseTest MakeImp
                            MakeInitialValueSymbols ReportVC SETUPCALLVCGEN baseName vcs1
                            (ENTRIES SETUPCALLVCGEN)
                            (GLOBALVARS VCsTraceFlag StatementsAlreadySeen SymbolsUsedInProgramUnit
                                        stmtlst)
                            (NOLINKFNS . T)))
        (VARS (VCsTraceFlag NIL))))

(RPAQQ VCGENFNS (AltersList AltersListHelper AssignmentSubst AssignmentSubst1
                            ConcurrentAssignmentSubst GETDEFINITIONS GenNewSymbols GetLastAssertion
                            GetLoopAssertion MakeBaseName MakeCallRelationName MakeCaseTest MakeImp
                            MakeInitialValueSymbols ReportVC SETUPCALLVCGEN Someify TVCS baseName
                            vcs1))
(DEFINEQ

                                                                                                1


(AltersList
  [LAMBDA (stmt)
    (PROG ((altlst (AltersListHelper stmt)))
          (RETURN (INTERSECTION altlst altlst])

                                                                                                2


(AltersListHelper
  [LAMBDA (stmt)
    (SELECTQ stmt:SYNTACTICTYPE
             ((compoundStatement repeatStatement)
               (for s in stmt:statement join (AltersListHelper s)))
             ((procedureStatement concurrentAssignmentStatement)
               (for v in stmt:variable collect (baseName v)))
             (assignmentStatement < (baseName stmt:variable)
                                  >)
             (caseStatement < !! (for c in stmt:caseElementList
                                    join (for s in c:statement join (AltersListHelper s)))
                            !
                            (AltersListHelper stmt:statement)
                            >)
             (ifStatement < !! (AltersListHelper stmt:statement)
                          !
                          (AltersListHelper stmt:statement#)
                          >)
             (labelStatement (AltersListHelper stmt:simpleStatement))
             (whileStatement (AltersListHelper stmt:statement))
             NIL])

                                                                                                3


(AssignmentSubst
  [LAMBDA (x e r)
    (SUBLIS <(AssignmentSubst1 x e) > r])

                                                                                                4


(AssignmentSubst1
  [LAMBDA (x expression)
    (if (ATOM x)
        then <x ! expression>
      else <(baseName x) !(create Assign
                                  baseName ←(baseName x)
                                  qualifiers ←(baseNameSubst x '@)
                                  newValue ← expression)
             >])

                                                                                                5
```

```
(ConcurrentAssignmentSubst
  [LAMBDA (varlist explist expression)                    (* R.Bates "14-Dec-79 14:43")
    (PROG (subpairs)
          (subpairs←(for v in varlist as ex in explist collect (AssignmentSubst1 v ex)))
          [subpairs←(for sp on subpairs bind duplicate collect sp:1
                       unless (PROGN duplicate←(FASSOC sp:1:OLD sp::1)
                                     (if duplicate
                                         then (PUTASSOC sp:1:OLD
                                                        (create Assign
                                                                baseName ← duplicate:NEW
                                                           using sp:1:NEW)
                                                        sp::1]
          (RETURN (SUBLIS subpairs expression])
```

6

```
(GETDEFINITIONS
  [LAMBDA (programUnit)                                   (* Edited by R.Bates on 22-JUL-77;
                                                          from version 105)
    (if programUnit:SYNTACTICTYPE='program
        then (GETDEFINITIONS programUnit:procedureOrFunctionDeclaration)
      else                                                (* programUnit is a procedureOrFunctionDeclaration)
           <<programUnit:identifier ! programUnit> !(for dop in programUnit:block:declareopt
                                                        when
                                                     dop:declareType:procedureOrFunctionDeclaration
                                                        join (GETDEFINITIONS
                                                     dop:declareType:procedureOrFunctionDeclaration))
            >])
```

7

```
(GenNewSymbols
  [LAMBDA (variables)                                     (* R.Erickson "27-Feb-81 18:03")
    (for v in variables bind newvar nvi collect (for n from 1 bind (base ←(MakeBaseName v))
                                                          newvar
                                                 thereis [newvar←(MakeExtension
                                                                  (PACK* base n)
                                                                  (Extension (baseName v]
                                                         (if newvar ~MEMB SymbolsUsedInProgramUnit
                                                             then (push SymbolsUsedInProgramUnit
                                                                        newvar)
                                                                  nvi←(Shorten newvar)
                                                                  (declareVar
                                                                    <nvi>
                                                                    (TranslateLitAtom v):Type 'vc)
                                                                  T)
                                                 finally (RETURN newvar])
```

8

```
(GetLastAssertion
  [LAMBDA (stmtlst)
    (if stmtlst:1:SYNTACTICTYPE='assertStatement
        then stmtlst:1:assertion])
```

9

```
(GetLoopAssertion
  [LAMBDA (stmt)                                          (* Edited by R.Bates on 16-JAN-78;
                                                          from version 118)
    (PROG (temp)
          (RETURN (if stmt:assertion
                    elseif temp←(GetLastAssertion stmtlst::1)
                      then stmtlst←stmtlst::1
                           temp
                    else (PRINTLINES "Can not find loop assertion:***" T)
                         TRUE])
```

10

```
(MakeBaseName
  [LAMBDA (variable)                                      (* R.Bates "13-Dec-79 17:09")
    (if (ATOM variable)
        then (Shorten variable)
      else (SELECTQ (Shorten variable:SYNTACTICTYPE)
```

```
                (ArrayAccess (PACK* (MakeBaseName variable:Array)
                                    (MakeBaseName variable:Index)))
                (RecordAccess (PACK* (MakeBaseName variable:Record)
                                     (MakeBaseName variable:Field)))
                (HeapOrFileAccess (PACK* (MakeBaseName variable:HeapOrFile)
                                         (MakeBaseName variable:Pointer)))
                (GENSYM])
```

11

```
(MakeCallRelationName
  [LAMBDA (p)
    (MKATOM (CONCAT p "Call"])
```

12

```
(MakeCaseTest
  [LAMBDA (expression labelList)                    (* R.Bates "4-Dec-79 10:23")
    (if (EQLENGTH labelList 1)
        then <EQOP expression labelList:1>
      else (N2BINARY <OROP !(for z in labelList collect <EQOP expression z>)
                     >])
```

13

```
(MakeImp
  [LAMBDA (h c)                                     (* Edited by R.Bates on 16-JAN-78:
                                                    from version 116)

    (if (ATOM c)
        then (if c=TRUE
                 then TRUE
               elseif c=FALSE
                 then (MakeNot h)
               else <IMPOP h c>)
      elseif h=TRUE
        then c
      elseif c:1=IMPOP
        then <IMPOP (MakeAnd h c:2)
                    c:3>
      else <IMPOP h c>])
```

14

```
(MakeInitialValueSymbols
  [LAMBDA (identifiers char)                        (* D.Thompson "12-Aug-80 06:45")
    (for i in identifiers collect (MakeExtension (PACK* (Shorten i)
                                                        (if char
                                                          else SingleQuote))
                                                 (Extension i])
```

15

```
(ReportVC
  [LAMBDA (vc)
    (if VCsTraceFlag
        then (INFIX\PRINT vc))
    vc])
```

16

```
(SETUPCALLVCGEN
  [LAMBDA (programUnit)                             (* D.Musser "3-Dec-79 13:30")
                                                    (* Returns a list whose first element is the "Call Lemma"

                                                    for programUnit and whose remaining elements are the
                                                    verification conditions for programUnit)
    (PROG (formalsAndImports varFormalsAndImports precondition postcondition stmtlst
                             SymbolsUsedInProgramUnit)
          (formalsAndImports← < !!(formalParameters programUnit:formalParameterSection)
                              !(formalParameters programUnit:formalParameterSection#)
                              >)
          (varFormalsAndImports←programUnit:identifier##)
          (SymbolsUsedInProgramUnit←formalsAndImports)
          (precondition←(if programUnit:block:assertion
                         else TRUE))
```

```
        (postcondition←(if programUnit:block:assertion#
                           else TRUE))
        (stmtlst← <(create concurrentAssignmentStatement
                           variable ←(MakeInitialValueSymbols formalsAndImports)
                           expression ← formalsAndImports)
                    (create assumeStatement
                           assertion ← precondition)
                    >)
        (StatementsAlreadySeen←NIL)
        (RETURN < !!(vcs1 <(if programUnit:unitKind:LEXEME='FUNCTION
                               then (create assignmentStatement
                                           variable ←(if programUnit:identifier#
                                                         else programUnit:identifier)
                                           expression ←(<programUnit:identifier
                                                          ! formalsAndImports>))
                               else (create procedureStatement
                                           identifier ← programUnit:identifier
                                           expression ← formalsAndImports
                                           variable ← varFormalsAndImports))
                           ! stmtlst>
                          (if programUnit:unitKind:LEXEME='FUNCTION
                              then postcondition
                           else (Someify (for v in formalsAndImports
                                            unless (MEMB v varFormalsAndImports) collect v)
                                         postcondition)))
                    ! < !!(if programUnit:unitKind:LEXEME='FUNCTION
                              then <(ReportVC <SOMEOP (if programUnit:identifier#
                                                          else programUnit:identifier)
                                                      (MakeImp precondition postcondition)
                                                      >)
                                   >)
                    !(vcs1 <programUnit:block:compoundStatement ! stmtlst> postcondition)
                    >>])
```

17

```
(Someify
  [LAMBDA (vlist predicate)                                    (* D.Musser "25-Sep-79 07:15")
                                                               (* For each variable in vlist that occurs in predicate,
                                                               an existential quantifier is added to predicate.)
    (for v in vlist when (OccursIn v predicate) do predicate← <SOMEOP v predicate>)
    predicate])
```

18

```
(TVCS
  [LAMBDA (file)                                               (* D.Thompson "26-Feb-80 11:29")
    (if (ATOM file)
        then (PARSE file)
             file←!VALUE)
    (if file
        then (for def in LastProgramUnits do (SETUPCALLVCGEN def::1))
      else (ParsingError T])
```

19

```
(baseName
  [LAMBDA (v)                                                  (* R.Erickson " 3-Feb-81 16:55")
    (if (ATOM v)
        then v
      else (baseName (SELECTQ (Shorten v:SYNTACTICTYPE)
                              (ArrayAccess v:Array)
                              (RecordAccess v:Record)
                              (HeapOrFileAccess v:HeapOrFile)
                              (PROGN (Heading "Unrecognized variable:")
                                     (PrettyPrint v T)
                                     (AffirmError])
```

20

```
(vcs1
  [LAMBDA (stmtlst post)                                       (* R.Bates "30-Apr-79 16:07")
    (if stmtlst=NIL
        then <(ReportVC post) >
      else
       (PROG (stmt invariant)
             (stmt←stmtlst:1)
```

```
(RETURN
  (if stmt=NIL
      then (vcs1 stmtlst::1 post)
    else
     (SELECTQ
       stmt:SYNTACTICTYPE
       (assertStatement < (ReportVC (MakeImp stmt:assertion post))
                          !
                          (if (MEMBER stmtlst StatementsAlreadySeen) or stmtlst::1=NIL
                              then NIL
                            else StatementsAlreadySeen← <stmtlst
                                                          ! StatementsAlreadySeen>
                                 (vcs1 stmtlst::1 stmt:assertion))
                        >)
       (assignmentStatement (vcs1 stmtlst::1 (AssignmentSubst stmt:variable
                                                              stmt:expression post)))
       (assumeStatement (vcs1 stmtlst::1 (MakeImp stmt:assertion post)))
       (caseStatement < !! (for y in stmt:caseElementList
                              join (vcs1 <y:statement (create assumeStatement
                                                              assertion ←(MakeCaseTest
                                                                stmt:expression
                                                                y:caseLabel))
                                                    ! stmtlst::1>
                                                  post))
                      !
                      [if stmt:statement
                          then (vcs1
                                  <stmt:statement [create assumeStatement
                                                          assertion ←(MakeNot
                                                            (MakeCaseTest
                                                              stmt:expression
                                                              (for y
                                                                 in stmt:caseElementList
                                                                 join (COPY y:caseLabel]
                                                ! stmtlst::1>
                                               post)
                        elseif (MEMBER stmtlst StatementsAlreadySeen)
                          then NIL
                        else StatementsAlreadySeen← <stmtlst
                                                      ! StatementsAlreadySeen>
                             (vcs1 stmtlst::1 (MakeCaseTest stmt:expression
                                                            (for y
                                                               in stmt:caseElementList
                                                               join (COPY y:caseLabel]
                     >)
       (compoundStatement (vcs1 < !(REVERSE stmt:statement) ! stmtlst::1> post))
       (concurrentAssignmentStatement (vcs1 stmtlst::1 (ConcurrentAssignmentSubst
                                                          stmt:variable
                                                        stmt:expression
                                                                  post)))
       [forStatement
         (PROG (eqvWhile temp)
               (eqvWhile←(create compoundStatement
                                 statement ←(<<(create assignmentStatement
                                                       variable ← stmt:identifier
                                                       expression ← stmt:expression)
                                   (create whileStatement
                                           assertion ←(GetLoopAssertion stmt)
                                           expression ←(<<(if stmt:direction='TO
                                                              then LEOP
                                                            else GEOP)
                                             stmt:identifier stmt:expression#>)
                                           statement ←(create
                                             compoundStatement
                                             statement ←(<(create assumeStatement
                                                                  assertion ←(<<(if
                                                                      stmt:direction='TO
                                                                                then
                                                                                 LEOP
                                                                               else
                                                                                GEOP)
                                                                    stmt:expression
                                                                    stmt:identifier>))
                                               stmt:statement
                                               (create assignmentStatement
                                                       variable ← stmt:identifier
                                                       expression ←(<<(if
                                                                  stmt:direction='TO
                                                                       then ADDOP
                                                                       else DIFFOP)
                                                         stmt:identifier 1>))
```

```
                                        >))
                                      assertion# ← stmt:assertion#)
                                >)))
        (RETURN (vcs1 <eqvWhile ! stmtlst::1> post]
(goToStatement (vcs1 stmtlst::1 stmt:assertion))
(ifStatement < ! (vcs1 <stmt:statement (create assumeStatement
                                                assertion ← stmt:expression)
                          ! stmtlst::1>
                       post)
             !
             (vcs1 <stmt:statement# (create assumeStatement
                                             assertion ←(MakeNot stmt:expression))
                     ! stmtlst::1>
                  post)
             >)
(labelStatement (vcs1 <stmt:simpleStatement ! stmtlst::1> post))
[procedureStatement (PROG (newSymbols)
                          (newSymbols←(GenNewSymbols stmt:variable))
                          (RETURN (vcs1 stmtlst::1
                                        (MakeImp
                                          <'computes\ProcedureCall
                                            <stmt:identifier ! stmt:expression
                                              ! stmt:identifier#>
                                            <'result\ProcedureCall
                                              ! newSymbols>>
                                          (ConcurrentAssignmentSubst
                                                                stmt:variable
                                                                   newSymbols
                                                                     post]
(proveStatement < !! (if (MEMBER stmtlst StatementsAlreadySeen)
                         or stmtlst::1=NIL
                        then NIL
                       else StatementsAlreadySeen←
                            <stmtlst ! StatementsAlreadySeen>
                            (vcs1 stmtlst::1 stmt:assertion))
                !
                (vcs1 stmtlst::1 (MakeImp stmt:assertion post))
                >)
(repeatStatement
  (PROG (stmtlst1)
        (stmtlst1←(REVERSE stmt:statement))
        (invariant←(if (GetLastAssertion stmtlst1)
                   else (PRINTLINES
                          "Can not find Assertion for Repeat Statement:***"
                                T)
                        TRUE))
        (RETURN
          <(ReportVC (MakeImp (MakeAnd invariant stmt:expression)
                              post))
            !(if (MEMBER stmtlst StatementsAlreadySeen)
                 then NIL
              else StatementsAlreadySeen← <stmtlst
                                             ! StatementsAlreadySeen>
                   (vcs1
                     < !! stmtlst1::1
                       (create splitStatement
                               leftStatementList ← stmtlst::1
                               rightStatementList ←(<(create
                                   assumeStatement
                                   assertion ←(MakeAnd invariant
                                                       (MakeNot stmt:expression))
                                   )
                                 >))
                     >
                    invariant))
          >)))
(returnStatement (vcs1 stmtlst::1 stmt:assertion))
(splitStatement < !! (vcs1 stmt:leftStatementList post)
                !
                (vcs1 stmt:rightStatementList post)
                >)
(statement (vcs1 stmtlst::1 post))
(whileStatement
  invariant←
  (GetLoopAssertion stmt)
  (PROG (altlst newSymbols initvals)
        (altlst←(AltersList stmt:statement))
        (newSymbols←(GenNewSymbols altlst))
        (initvals←(MakeInitialValueSymbols altlst "Initial"))
        (RETURN
          <
```

```
                                !!(vcs1 stmtlst::1
                                      (ConcurrentAssignmentSubst < ! altlst ! initvals>
                                                                 < ! newSymbols ! altlst
                                                                 (MakeImp
                                                                   (MakeAnd
                                                                     invariant
                                                                     (if stmt:assertion#
                                                                         then
                                                                          (MakeAnd (MakeNot
                                                                                stmt:expression)
                                                                                   stmt:assertion#)
                                                                       else (MakeNot
                                                                                stmt:expression)))
                                                                   post)))
                                !(if (MEMBER stmtlst StatementsAlreadySeen)
                                     then NIL
                                   else StatementsAlreadySeen← <stmtlst
                                                               ! StatementsAlreadySeen>
                                        < !!(vcs1 stmtlst::1 (ConcurrentAssignmentSubst initvals
                                                                                        altlst
                                                                                        invariant))
                                          ! <
                                          !!(if stmt:assertion#
                                                then
                                                 <(ConcurrentAssignmentSubst
                                                     initvals altlst
                                                     (MakeImp (MakeAnd invariant
                                                                       (MakeNot stmt:expression))
                                                              stmt:assertion#))
                                                 >)
                                          !(vcs1 <stmt:statement (create assumeStatement
                                                                         assertion ←(MakeAnd
                                                                                      invariant
                                                                                    stmt:expression))
                                                   (create concurrentAssignmentStatement
                                                           variable ← initvals
                                                           expression ← altlst)
                                                   >
                                                  (if stmt:assertion#
                                                      then
                                                       [MakeAnd invariant
                                                                (MakeImp
                                                                  (ConcurrentAssignmentSubst
                                                                    < ! altlst ! initvals
                                                                    < ! newSymbols ! altlst
                                                                    (MakeAnd (MakeNot stmt:expression)
                                                                             (MakeAnd invariant
                                                                                      stmt:assertion#)))
                                                                  (ConcurrentAssignmentSubst
                                                                    altlst newSymbols
                                                                    (MakeAnd invariant stmt:assertion#]
                                                    else invariant))
                                          >>)
                                     >)))
                              (PROGN (PRINTLINES "Unrecognized statement:***" T stmt T)
                                     NIL])
)

(RPAQQ VCGENRECORDS (splitStatement ArrayWrite Assign Expression SubPair RecordWrite))
[DECLARE: EVAL@COMPILE

(TYPERECORD splitStatement (leftStatementList rightStatementList))

(TYPERECORD ArrayWrite (Array Index Value))

(TYPERECORD Assign (baseName qualifiers newValue))

(RECORD Expression (Operator . Arguments))

(RECORD SubPair (OLD . NEW))

(TYPERECORD RecordWrite (Record Field Value))
]
[DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY
(BLOCK: VCGENBLOCK AltersList AltersListHelper AssignmentSubst AssignmentSubst1
        ConcurrentAssignmentSubst GenNewSymbols GetLastAssertion GetLoopAssertion MakeBaseName
        MakeCallRelationName MakeCaseTest MakeImp MakeInitialValueSymbols ReportVC SETUPCALLVCGEN
        baseName vcs1 (ENTRIES SETUPCALLVCGEN)
        (GLOBALVARS VCsTraceFlag StatementsAlreadySeen SymbolsUsedInProgramUnit stmtlst
        (NOLINKFNS . T))
]
```

```
(RPAQ VCsTraceFlag NIL)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (1057 18216 (AltersList 1069 . 1201) (AltersListHelper 1205 . 2084) (AssignmentSubst
2088 . 2174) (AssignmentSubst1 2178 . 2444) (ConcurrentAssignmentSubst 2448 . 3075) (GETDEFINITIONS
3079 . 3766) (GenNewSymbols 3770 . 4466) (GetLastAssertion 4470 . 4599) (GetLoopAssertion 4603 . 5001)
 (MakeBaseName 5005 . 5580) (MakeCallRelationName 5584 . 5656) (MakeCaseTest 5660 . 5957) (MakeImp
5961 . 6401) (MakeInitialValueSymbols 6405 . 6689) (ReportVC 6693 . 6787) (SETUPCALLVCGEN 6791 . 9115)
 (Someify 9119 . 9533) (TVCS 9537 . 9849) (baseName 9853 . 10277) (vcs1 10281 . 18213)))))
STOP
```