VERIFICATION OF TCP-LIKE DATA TRANSPORT FUNCTIONS

B. L. DiVito

Technical Report 27 August 1982

Institute for Computing Science
The University of Texas at Austin
Austin, Texas 78712

TABLE OF CONTENTS

Introduction

This report contains the transcripts of a mechanical verification of the "Nano TCP protocol," which is an abstract model of the data transfer functions of the TCP protocol [Postel 80]. A description of the Nano TCP protocol, as well as complete documentation on the verification methods, can be found in a separate report [DiVito 82]. Reference to this report is necessary, since the following material is not self-contained.

There are two parts to the actual transcripts. First, there is a log of the VC reduction program [DiVito 82] run on the VCs of the sender and receiver process. The result is the production of a transformed and simplified set of VCs. Second, the transcripts of the proofs of these transformed VCs are displayed. The proofs were carried out using the Boyer-Moore theorem prover [Boyer 79].

APPENDIX A
VC REDUCTION

```
VC reduction log for: 'SENDER'                          23-Jun-82 10:11:49

Assertion function:

   (SENDER.INT SOURCE ACK.IN RECEIPTS PKT.OUT UNACK NEXT EDGE QUEUE
               TIMING TO.TIME)
      =
   (AND (PMAPP QUEUE)
        (NUMBERP NEXT)
        (ALL (FAPPLY (QUOTE MPAIRP)
                     (FLATTEN PKT.OUT)))
        (ALL (FAPPLY (QUOTE CONTIG)
                     PKT.OUT))
        (CONTIG QUEUE)
        (CONSISTENT (QUOTE DOM)
                    (QUOTE EQUAL)
                    (FLATTEN PKT.OUT))
        (FOLLOWS QUEUE (FLATTEN PKT.OUT))
        (EQUAL (FLATTEN SOURCE)
               (RANGE (REDUCE (QUOTE WITH)
                              (NULL)
                              (FLATTEN PKT.OUT))))
        (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
               (SUB1 NEXT))
        (IF (EQUAL NEXT 0)
            (AND (EQUAL (FLATTEN PKT.OUT)
                        (NULL))
                 (EQUAL QUEUE (NULL)))
            (AND (SEQP (FLATTEN PKT.OUT))
                 (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                          (NULL)
                                          (FLATTEN PKT.OUT))))
                        (SUB1 NEXT))
                 (IF (SEQP QUEUE)
                     (EQUAL (DOM (LST QUEUE))
                            (SUB1 NEXT))
                     T))))

----------------------------------------

Path conditions for VC 'SENDER#3':
   (AND (NOT (LESSP EDGE NEXT))
        (NOT TIMING)
        (SEQP MESS)
        (PSEQP MESS))
```

Variable substitution:

```
   SOURCE         :=  (APR SOURCE MESS)
   ACK.IN         :=  ACK.IN
   RECEIPTS       :=  RECEIPTS
   PKT.OUT        :=  (APR PKT.OUT (ENMAP MESS NEXT))
   UNACK          :=  UNACK
   NEXT           :=  (PLUS NEXT (SIZE MESS))
   EDGE           :=  EDGE
   QUEUE          :=  (UNION QUEUE (ENMAP MESS NEXT))
   TIMING         :=  T
   TO.TIME        :=  (PLUS TO.TIME DELTA.T)
```

Instantiated assertion:

```
   (AND
     (PMAPP (UNION QUEUE (ENMAP MESS NEXT)))
     (NUMBERP (PLUS NEXT (SIZE MESS)))
     (ALL (FAPPLY (QUOTE MPAIRP)
                  (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))
     (ALL (FAPPLY (QUOTE CONTIG)
                  (APR PKT.OUT (ENMAP MESS NEXT))))
     (CONTIG (UNION QUEUE (ENMAP MESS NEXT)))
     (CONSISTENT (QUOTE DOM)
                 (QUOTE EQUAL)
                 (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
     (FOLLOWS (UNION QUEUE (ENMAP MESS NEXT))
              (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
     (EQUAL (FLATTEN (APR SOURCE MESS))
            (RANGE (REDUCE (QUOTE WITH)
                           (NULL)
                           (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))
                           )))
     (EQUAL (HIGHEST (DOMAIN (FLATTEN (APR PKT.OUT
                                          (ENMAP MESS NEXT)))))
            (SUB1 (PLUS NEXT (SIZE MESS))))
     (IF
       (EQUAL (PLUS NEXT (SIZE MESS))
              0)
       (AND (EQUAL (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))
                   (NULL))
            (EQUAL (UNION QUEUE (ENMAP MESS NEXT))
                   (NULL)))
       (AND (SEQP (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
            (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                     (NULL)
                                     (FLATTEN (APR PKT.OUT
                                                   (ENMAP MESS NEXT)
                                                   )))))
                   (SUB1 (PLUS NEXT (SIZE MESS)))))
            (IF (SEQP (UNION QUEUE (ENMAP MESS NEXT)))
                (EQUAL (DOM (LST (UNION QUEUE (ENMAP MESS NEXT))))
                       (SUB1 (PLUS NEXT (SIZE MESS))))
                T))))
```

```
--------------------

Generating new VC 'SENDER#3.0.1.1':

   (IMPLIES (EQUAL QUEUE (NULL))
            (EQUAL (UNION QUEUE (ENMAP MESS NEXT))
                   (JOIN QUEUE (ENMAP MESS NEXT))))


VC not proved, being written.

--------------------

Generating new VC 'SENDER#3.0.1.2':

   (IMPLIES (AND (PMAPP QUEUE)
                 (NUMBERP NEXT)
                 (NOT (EQUAL NEXT 0))
                 (IF (SEQP QUEUE)
                     (EQUAL (DOM (LST QUEUE))
                            (SUB1 NEXT))
                     T)
                 (SEQP MESS))
            (EQUAL (UNION QUEUE (ENMAP MESS NEXT))
                   (JOIN QUEUE (ENMAP MESS NEXT))))
```

```
VC not proved, being written.

Equality substitution performed.

------------------

Generating new VC 'SENDER#3.0.2.1':

   (IMPLIES (EQUAL (FLATTEN PKT.OUT)
                   (NULL))
            (EQUAL (REDUCE (QUOTE WITH)
                           (NULL)
                           (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))
                           )
                   (JOIN (REDUCE (QUOTE WITH)
                                 (NULL)
                                 (FLATTEN PKT.OUT))
                         (ENMAP MESS NEXT))))


VC not proved, being written.

------------------

Generating new VC 'SENDER#3.0.2.2':

   (IMPLIES (AND (NUMBERP NEXT)
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.OUT)))
                 (NOT (EQUAL NEXT 0))
                 (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                          (NULL)
                                          (FLATTEN PKT.OUT))))
                        (SUB1 NEXT))
                 (SEQP MESS))
            (EQUAL (REDUCE (QUOTE WITH)
                           (NULL)
                           (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))
                           )
                   (JOIN (REDUCE (QUOTE WITH)
                                 (NULL)
                                 (FLATTEN PKT.OUT))
                         (ENMAP MESS NEXT))))
```

```
VC not proved, being written.

Equality substitution performed.

-------------------

Generating new VC 'SENDER#3.1.1':

   (IMPLIES (EQUAL QUEUE (NULL))
            (PMAPP (JOIN QUEUE (ENMAP MESS NEXT))))


VC not proved, being written.

-------------------

Generating new VC 'SENDER#3.1.2':

   (IMPLIES (AND (PMAPP QUEUE)
                 (NUMBERP NEXT)
                 (NOT (EQUAL NEXT 0))
                 (IF (SEQP QUEUE)
                     (EQUAL (DOM (LST QUEUE))
                            (SUB1 NEXT))
                     T)
                 (SEQP MESS))
            (PMAPP (JOIN QUEUE (ENMAP MESS NEXT))))


VC not proved, being written.

To be proved with lemmas (PMAPP.JOIN.2)


-------------------

Generating new VC 'SENDER#3.2':

   (NUMBERP (PLUS NEXT (SIZE MESS)))

This conjecture simplifies, trivially, to:

      T.
```

```
Q.E.D.

VC has been proved.

Compute time:    3.737 sec,   GC time:    0.000 sec.

------------------

Generating new VC 'SENDER#3.3':

   (IMPLIES (ALL (FAPPLY (QUOTE MPAIRP)
                         (FLATTEN PKT.OUT)))
            (ALL (FAPPLY (QUOTE MPAIRP)
                         (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))
            )


VC not proved, being written.

------------------

Generating new VC 'SENDER#3.4':

   (IMPLIES (ALL (FAPPLY (QUOTE CONTIG)
                         PKT.OUT))
            (ALL (FAPPLY (QUOTE CONTIG)
                         (APR PKT.OUT (ENMAP MESS NEXT)))))


VC not proved, being written.

------------------

Generating new VC 'SENDER#3.5.1':

   (IMPLIES (EQUAL QUEUE (NULL))
            (CONTIG (JOIN QUEUE (ENMAP MESS NEXT))))


VC not proved, being written.

------------------
```

```
Generating new VC 'SENDER#3.5.2':

   (IMPLIES (AND (CONTIG QUEUE)
                 (PMAPP QUEUE)
                 (NUMBERP NEXT)
                 (NOT (EQUAL NEXT 0))
                 (IF (SEQP QUEUE)
                     (EQUAL (DOM (LST QUEUE))
                            (SUB1 NEXT))
                     T)
                 (SEQP MESS))
            (CONTIG (JOIN QUEUE (ENMAP MESS NEXT))))


VC not proved, being written.

-------------------

Generating new VC 'SENDER#3.6.1':

   (IMPLIES (EQUAL (FLATTEN PKT.OUT)
                   (NULL))
            (CONSISTENT (QUOTE DOM)
                        (QUOTE EQUAL)
                        (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))


VC not proved, being written.

-------------------

Generating new VC 'SENDER#3.6.2':

   (IMPLIES (AND (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (FLATTEN PKT.OUT))
                 (NUMBERP NEXT)
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.OUT)))
                 (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                        (SUB1 NEXT))
                 (NOT (EQUAL NEXT 0))
                 (SEQP MESS))
            (CONSISTENT (QUOTE DOM)
                              (QUOTE EQUAL)
                              (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))


VC not proved, being written.

------------------

Generating new VC 'SENDER#3.7':

   (IMPLIES (FOLLOWS QUEUE (FLATTEN PKT.OUT))
            (FOLLOWS (JOIN QUEUE (ENMAP MESS NEXT))
                     (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))


VC not proved, being written.

------------------

Generating new VC 'SENDER#3.8':

   (IMPLIES (AND (EQUAL (FLATTEN SOURCE)
                        (RANGE (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (FLATTEN PKT.OUT))))
                 (PSEQP MESS))
            (EQUAL (FLATTEN (APR SOURCE MESS))
                   (RANGE (JOIN (REDUCE (QUOTE WITH)
                                        (NULL)
                                        (FLATTEN PKT.OUT))
                                (ENMAP MESS NEXT)))))


VC not proved, being written.

------------------

Generating new VC 'SENDER#3.9.1':

   (IMPLIES (AND (EQUAL NEXT 0)
                 (EQUAL (FLATTEN PKT.OUT)
                        (NULL))
                 (SEQP MESS))
            (EQUAL (HIGHEST (DOMAIN (FLATTEN (APR PKT.OUT
                                                            (ENMAP MESS NEXT))
                                                       )))
                       (SUB1 (PLUS NEXT (SIZE MESS)))))


VC not proved, being written.

-------------------

Generating new VC 'SENDER#3.9.2':

   (IMPLIES (AND (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                        (SUB1 NEXT))
                 (NUMBERP NEXT)
                 (NOT (EQUAL NEXT 0))
                 (SEQP MESS))
            (EQUAL (HIGHEST (DOMAIN (FLATTEN (APR PKT.OUT
                                                            (ENMAP MESS NEXT))
                                                       )))
                       (SUB1 (PLUS NEXT (SIZE MESS)))))


VC not proved, being written.

-------------------

Generating new VC 'SENDER#3.10.1.1':

   (IMPLIES (AND (EQUAL (PLUS NEXT (SIZE MESS))
                        0)
                 (EQUAL NEXT 0)
                 (SEQP MESS))
            F)


VC not proved, being written.

-------------------

VC 'SENDER#3.10.1.1' subsumes VC 'SENDER#3.10.1.2':

   (IMPLIES (AND (EQUAL (FLATTEN PKT.OUT)
                        (NULL))
                 (EQUAL (PLUS NEXT (SIZE MESS))
                         0)
                  (EQUAL NEXT 0)
                  (PMAPP QUEUE)
                  (NUMBERP NEXT)
                  (ALL (FAPPLY (QUOTE MPAIRP)
                               (FLATTEN PKT.OUT)))
                  (ALL (FAPPLY (QUOTE CONTIG)
                               PKT.OUT))
                  (CONTIG QUEUE)
                  (CONSISTENT (QUOTE DOM)
                              (QUOTE EQUAL)
                              (FLATTEN PKT.OUT))
                  (FOLLOWS QUEUE (FLATTEN PKT.OUT))
                  (EQUAL (FLATTEN SOURCE)
                         (RANGE (REDUCE (QUOTE WITH)
                                        (NULL)
                                        (FLATTEN PKT.OUT))))
                  (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                         (SUB1 NEXT))
                  (NOT (LESSP EDGE NEXT))
                  (SEQP MESS)
                  (PSEQP MESS)
                  (EQUAL QUEUE (NULL)))
             (EQUAL (JOIN QUEUE (ENMAP MESS NEXT))
                    (NULL)))
```

Generating new VC 'SENDER#3.10.2':

```
   (IMPLIES (AND (EQUAL (PLUS NEXT (SIZE MESS))
                        0)
                 (NUMBERP NEXT))
            (EQUAL NEXT 0))
```

VC not proved, being written.

Generating new VC 'SENDER#3.10.3.1':

```
   (IMPLIES (AND (NOT (SEQP (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))
                                     )))
```

```
                (SEQP MESS))
            F)


VC not proved, being written.

-------------------

Generating new VC 'SENDER#3.10.3.2':

   (IMPLIES
     (AND
       (NOT (EQUAL (DOM (LST (JOIN (REDUCE (QUOTE WITH)
                                           (NULL)
                                           (FLATTEN PKT.OUT))
                                   (ENMAP MESS NEXT))))
                        (SUB1 (PLUS NEXT (SIZE MESS)))))
       (SEQP MESS))
     F)


VC not proved, being written.

-------------------

Generating new VC 'SENDER#3.10.3.3':

   (IMPLIES (AND (SEQP (JOIN QUEUE (ENMAP MESS NEXT)))
                 (NOT (EQUAL (DOM (LST (JOIN QUEUE
                                                (ENMAP MESS NEXT))))
                             (SUB1 (PLUS NEXT (SIZE MESS)))))
                 (SEQP MESS))
            F)


VC not proved, being written.

-------------------

VC 'SENDER#3.10.3.4' is trivially true:

   (IMPLIES (AND (EQUAL NEXT 0)
                 F
                 (SEQP MESS))
```

```
          (EQUAL (PLUS NEXT (SIZE MESS))
                 0))
```

VC 'SENDER#3.10.3.1' subsumes VC 'SENDER#3.10.4.1':

```
   (IMPLIES (AND (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                          (NULL)
                                          (FLATTEN PKT.OUT))))
                        (SUB1 NEXT))
                 (IF (SEQP QUEUE)
                     (EQUAL (DOM (LST QUEUE))
                            (SUB1 NEXT))
                     T)
                 (NOT (EQUAL (PLUS NEXT (SIZE MESS))
                             0))
                 (NOT (EQUAL NEXT 0))
                 (PMAPP QUEUE)
                 (NUMBERP NEXT)
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.OUT)))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              PKT.OUT))
                 (CONTIG QUEUE)
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (FLATTEN PKT.OUT))
                 (FOLLOWS QUEUE (FLATTEN PKT.OUT))
                 (EQUAL (FLATTEN SOURCE)
                        (RANGE (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (FLATTEN PKT.OUT))))
                 (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                        (SUB1 NEXT))
                 (NOT (LESSP EDGE NEXT))
                 (SEQP MESS)
                 (PSEQP MESS)
                 (SEQP (FLATTEN PKT.OUT)))
          (SEQP (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))
```

VC 'SENDER#3.10.3.2' subsumes VC 'SENDER#3.10.4.2':

```
   (IMPLIES (AND (SEQP (FLATTEN PKT.OUT))
                 (IF (SEQP QUEUE)
                     (EQUAL (DOM (LST QUEUE))
                            (SUB1 NEXT))
                     T)
                 (NOT (EQUAL (PLUS NEXT (SIZE MESS))
                             0))
                 (NOT (EQUAL NEXT 0))
                 (PMAPP QUEUE)
                 (NUMBERP NEXT)
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.OUT)))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              PKT.OUT))
                 (CONTIG QUEUE)
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (FLATTEN PKT.OUT))
                 (FOLLOWS QUEUE (FLATTEN PKT.OUT))
                 (EQUAL (FLATTEN SOURCE)
                        (RANGE (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (FLATTEN PKT.OUT))))
                 (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                        (SUB1 NEXT))
                 (NOT (LESSP EDGE NEXT))
                 (SEQP MESS)
                 (PSEQP MESS)
                 (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                          (NULL)
                                          (FLATTEN PKT.OUT))))
                        (SUB1 NEXT)))
            (EQUAL (DOM (LST (JOIN (REDUCE (QUOTE WITH)
                                           (NULL)
                                           (FLATTEN PKT.OUT))
                                   (ENMAP MESS NEXT))))
                   (SUB1 (PLUS NEXT (SIZE MESS)))))
```

VC 'SENDER#3.10.3.3' subsumes VC 'SENDER#3.10.4.3.1':

```
   (IMPLIES (AND (SEQP (JOIN QUEUE (ENMAP MESS NEXT)))
                (SEQP QUEUE)
                (SEQP (FLATTEN PKT.OUT))
                (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                         (NULL)
                                         (FLATTEN PKT.OUT))))
                       (SUB1 NEXT))
                (NOT (EQUAL (PLUS NEXT (SIZE MESS))
                            0))
                (NOT (EQUAL NEXT 0))
                (PMAPP QUEUE)
                (NUMBERP NEXT)
                (ALL (FAPPLY (QUOTE MPAIRP)
                             (FLATTEN PKT.OUT)))
                (ALL (FAPPLY (QUOTE CONTIG)
                             PKT.OUT))
                (CONTIG QUEUE)
                (CONSISTENT (QUOTE DOM)
                            (QUOTE EQUAL)
                            (FLATTEN PKT.OUT))
                (FOLLOWS QUEUE (FLATTEN PKT.OUT))
                (EQUAL (FLATTEN SOURCE)
                       (RANGE (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (FLATTEN PKT.OUT))))
                (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                       (SUB1 NEXT))
                (NOT (LESSP EDGE NEXT))
                (SEQP MESS)
                (PSEQP MESS)
                (EQUAL (DOM (LST QUEUE))
                       (SUB1 NEXT)))
           (EQUAL (DOM (LST (JOIN QUEUE (ENMAP MESS NEXT))))
                  (SUB1 (PLUS NEXT (SIZE MESS)))))
```

VC 'SENDER#3.10.3.3' subsumes VC 'SENDER#3.10.4.3.2':

```
   (IMPLIES (AND (NOT (EQUAL (DOM (LST (JOIN QUEUE
                                             (ENMAP MESS NEXT))))
                             (SUB1 (PLUS NEXT (SIZE MESS)))))
                 T
                 (SEQP (FLATTEN PKT.OUT))
                 (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                                 (NULL)
                                                 (FLATTEN PKT.OUT))))
                         (SUB1 NEXT))
                  (NOT (EQUAL (PLUS NEXT (SIZE MESS))
                              0))
                  (NOT (EQUAL NEXT 0))
                  (PMAPP QUEUE)
                  (NUMBERP NEXT)
                  (ALL (FAPPLY (QUOTE MPAIRP)
                               (FLATTEN PKT.OUT)))
                  (ALL (FAPPLY (QUOTE CONTIG)
                               PKT.OUT))
                  (CONTIG QUEUE)
                  (CONSISTENT (QUOTE DOM)
                              (QUOTE EQUAL)
                              (FLATTEN PKT.OUT))
                  (FOLLOWS QUEUE (FLATTEN PKT.OUT))
                  (EQUAL (FLATTEN SOURCE)
                         (RANGE (REDUCE (QUOTE WITH)
                                        (NULL)
                                        (FLATTEN PKT.OUT))))
                  (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                         (SUB1 NEXT))
                  (NOT (LESSP EDGE NEXT))
                  (SEQP MESS)
                  (PSEQP MESS)
                  (SEQP (JOIN QUEUE (ENMAP MESS NEXT))))
             (SEQP QUEUE))

--------------------

VC 'SENDER#3.10.4.3.3' is trivially true:

   (IMPLIES F (SEQP (JOIN QUEUE (ENMAP MESS NEXT))))

--------------------

VC 'SENDER#3.10.4.3.4' is trivially true:

   (IMPLIES T T)


----------------------------------------
```

Path conditions for VC 'SENDER#4':

```
    (AND (NOT (LESSP EDGE NEXT))
         TIMING
         (SEQP MESS)
         (PSEQP MESS))
```

Variable substitution:

```
    SOURCE        :=  (APR SOURCE MESS)
    ACK.IN        :=  ACK.IN
    RECEIPTS      :=  RECEIPTS
    PKT.OUT       :=  (APR PKT.OUT (ENMAP MESS NEXT))
    UNACK         :=  UNACK
    NEXT          :=  (PLUS NEXT (SIZE MESS))
    EDGE          :=  EDGE
    QUEUE         :=  (UNION QUEUE (ENMAP MESS NEXT))
    TIMING        :=  TIMING
    TO.TIME       :=  TO.TIME
```

Instantiated assertion:

```
    (AND
      (PMAPP (UNION QUEUE (ENMAP MESS NEXT)))
      (NUMBERP (PLUS NEXT (SIZE MESS)))
      (ALL (FAPPLY (QUOTE MPAIRP)
                   (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))
      (ALL (FAPPLY (QUOTE CONTIG)
                   (APR PKT.OUT (ENMAP MESS NEXT))))
      (CONTIG (UNION QUEUE (ENMAP MESS NEXT)))
      (CONSISTENT (QUOTE DOM)
                  (QUOTE EQUAL)
                  (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
      (FOLLOWS (UNION QUEUE (ENMAP MESS NEXT))
               (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
      (EQUAL (FLATTEN (APR SOURCE MESS))
             (RANGE (REDUCE (QUOTE WITH)
                            (NULL)
                            (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))
                            )))
      (EQUAL (HIGHEST (DOMAIN (FLATTEN (APR PKT.OUT
                                               (ENMAP MESS NEXT)))))
             (SUB1 (PLUS NEXT (SIZE MESS))))
```

```
  (IF
    (EQUAL (PLUS NEXT (SIZE MESS))
           0)
    (AND (EQUAL (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))
                (NULL))
         (EQUAL (UNION QUEUE (ENMAP MESS NEXT))
                (NULL)))
    (AND (SEQP (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
         (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                  (NULL)
                                  (FLATTEN (APR PKT.OUT
                                                (ENMAP MESS NEXT)
                                                )))))
                (SUB1 (PLUS NEXT (SIZE MESS))))
         (IF (SEQP (UNION QUEUE (ENMAP MESS NEXT)))
             (EQUAL (DOM (LST (UNION QUEUE (ENMAP MESS NEXT))))
                    (SUB1 (PLUS NEXT (SIZE MESS))))
             T))))
```

VC 'SENDER#3' subsumes VC 'SENDER#4':

```
 (IMPLIES
   (AND
     (NOT (LESSP EDGE NEXT))
     TIMING
     (SEQP MESS)
     (PSEQP MESS)
     (PMAPP QUEUE)
     (NUMBERP NEXT)
     (ALL (FAPPLY (QUOTE MPAIRP)
                  (FLATTEN PKT.OUT)))
     (ALL (FAPPLY (QUOTE CONTIG)
                  PKT.OUT))
     (CONTIG QUEUE)
     (CONSISTENT (QUOTE DOM)
                 (QUOTE EQUAL)
                 (FLATTEN PKT.OUT))
     (FOLLOWS QUEUE (FLATTEN PKT.OUT))
     (EQUAL (FLATTEN SOURCE)
            (RANGE (REDUCE (QUOTE WITH)
                           (NULL)
                           (FLATTEN PKT.OUT))))
     (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
            (SUB1 NEXT))
```

```
  (IF (EQUAL NEXT 0)
      (AND (EQUAL (FLATTEN PKT.OUT)
                  (NULL))
           (EQUAL QUEUE (NULL)))
      (AND (SEQP (FLATTEN PKT.OUT))
           (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                    (NULL)
                                    (FLATTEN PKT.OUT))))
                  (SUB1 NEXT))
           (IF (SEQP QUEUE)
               (EQUAL (DOM (LST QUEUE))
                      (SUB1 NEXT))
               T))))
(AND
  (PMAPP (UNION QUEUE (ENMAP MESS NEXT)))
  (NUMBERP (PLUS NEXT (SIZE MESS)))
  (ALL (FAPPLY (QUOTE MPAIRP)
               (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))
  (ALL (FAPPLY (QUOTE CONTIG)
               (APR PKT.OUT (ENMAP MESS NEXT))))
  (CONTIG (UNION QUEUE (ENMAP MESS NEXT)))
  (CONSISTENT (QUOTE DOM)
              (QUOTE EQUAL)
              (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
  (FOLLOWS (UNION QUEUE (ENMAP MESS NEXT))
           (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
  (EQUAL (FLATTEN (APR SOURCE MESS))
         (RANGE (REDUCE (QUOTE WITH)
                        (NULL)
                        (FLATTEN (APR PKT.OUT
                                      (ENMAP MESS NEXT))))))
  (EQUAL (HIGHEST (DOMAIN (FLATTEN (APR PKT.OUT
                                        (ENMAP MESS NEXT)))))
         (SUB1 (PLUS NEXT (SIZE MESS))))
  (IF
    (EQUAL (PLUS NEXT (SIZE MESS))
           0)
    (AND (EQUAL (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))
                (NULL))
         (EQUAL (UNION QUEUE (ENMAP MESS NEXT))
                (NULL)))
    (AND
      (SEQP (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
      (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                    (NULL)
                                    (FLATTEN (APR PKT.OUT
                                                  (ENMAP MESS NEXT))
                                             ))))
                 (SUB1 (PLUS NEXT (SIZE MESS))))
          (IF (SEQP (UNION QUEUE (ENMAP MESS NEXT)))
              (EQUAL (DOM (LST (UNION QUEUE (ENMAP MESS NEXT))))
                     (SUB1 (PLUS NEXT (SIZE MESS))))
              T)))))
```

Path conditions for VC 'SENDER#5':

```
   (AND (NOT (LESSP EDGE NEXT))
        (EQUAL MESS (NULL)))
```

Variable substitution:

```
   SOURCE       :=  (APR SOURCE MESS)
   ACK.IN       :=  ACK.IN
   RECEIPTS     :=  RECEIPTS
   PKT.OUT      :=  PKT.OUT
   UNACK        :=  UNACK
   NEXT         :=  NEXT
   EDGE         :=  EDGE
   QUEUE        :=  QUEUE
   TIMING       :=  TIMING
   TO.TIME      :=  TO.TIME
```

Instantiated assertion:

```
   (AND (PMAPP QUEUE)
        (NUMBERP NEXT)
        (ALL (FAPPLY (QUOTE MPAIRP)
                     (FLATTEN PKT.OUT)))
        (ALL (FAPPLY (QUOTE CONTIG)
                     PKT.OUT))
        (CONTIG QUEUE)
        (CONSISTENT (QUOTE DOM)
                    (QUOTE EQUAL)
                    (FLATTEN PKT.OUT))
        (FOLLOWS QUEUE (FLATTEN PKT.OUT))
        (EQUAL (FLATTEN (APR SOURCE MESS))
               (RANGE (REDUCE (QUOTE WITH)
                              (NULL)
                              (FLATTEN PKT.OUT))))
        (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
               (SUB1 NEXT))
        (IF (EQUAL NEXT 0)
            (AND (EQUAL (FLATTEN PKT.OUT)
                        (NULL))
                 (EQUAL QUEUE (NULL)))
            (AND (SEQP (FLATTEN PKT.OUT))
                 (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                          (NULL)
                                          (FLATTEN PKT.OUT))))
                        (SUB1 NEXT))
                 (IF (SEQP QUEUE)
                     (EQUAL (DOM (LST QUEUE))
                            (SUB1 NEXT))
                     T))))

--------------------

Generating new VC 'SENDER#5.0.1':

   (IMPLIES (EQUAL MESS (NULL))
            (EQUAL (FLATTEN (APR SOURCE MESS))
                   (FLATTEN SOURCE)))


VC not proved, being written.

Equality substitution performed.

--------------------

VC 'SENDER#5.1' is trivially true:

   (IMPLIES (PMAPP QUEUE)
            (PMAPP QUEUE))

--------------------

VC 'SENDER#5.2' is trivially true:
   (IMPLIES (NUMBERP NEXT)
            (NUMBERP NEXT))

-------------------

VC 'SENDER#5.3' is trivially true:

   (IMPLIES (ALL (FAPPLY (QUOTE MPAIRP)
                         (FLATTEN PKT.OUT)))
            (ALL (FAPPLY (QUOTE MPAIRP)
                         (FLATTEN PKT.OUT))))

-------------------

VC 'SENDER#5.4' is trivially true:

   (IMPLIES (ALL (FAPPLY (QUOTE CONTIG)
                         PKT.OUT))
            (ALL (FAPPLY (QUOTE CONTIG)
                         PKT.OUT)))

-------------------

VC 'SENDER#5.5' is trivially true:

   (IMPLIES (CONTIG QUEUE)
            (CONTIG QUEUE))

-------------------

VC 'SENDER#5.6' is trivially true:

   (IMPLIES (CONSISTENT (QUOTE DOM)
                        (QUOTE EQUAL)
                        (FLATTEN PKT.OUT))
            (CONSISTENT (QUOTE DOM)
                        (QUOTE EQUAL)
                        (FLATTEN PKT.OUT)))

-------------------

VC 'SENDER#5.7' is trivially true:
    (IMPLIES (FOLLOWS QUEUE (FLATTEN PKT.OUT))
             (FOLLOWS QUEUE (FLATTEN PKT.OUT)))
```

VC 'SENDER#5.8' is trivially true:

```
    (IMPLIES (EQUAL (FLATTEN SOURCE)
                    (RANGE (REDUCE (QUOTE WITH)
                                   (NULL)
                                   (FLATTEN PKT.OUT))))
             (EQUAL (FLATTEN SOURCE)
                    (RANGE (REDUCE (QUOTE WITH)
                                   (NULL)
                                   (FLATTEN PKT.OUT)))))
```

VC 'SENDER#5.9' is trivially true:

```
    (IMPLIES (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                    (SUB1 NEXT))
             (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                    (SUB1 NEXT)))
```

VC 'SENDER#5.10.1.1' is trivially true:

```
    (IMPLIES (EQUAL (FLATTEN PKT.OUT)
                    (NULL))
             (EQUAL (FLATTEN PKT.OUT)
                    (NULL)))
```

VC 'SENDER#5.10.1.2' is trivially true:

```
    (IMPLIES (EQUAL QUEUE (NULL))
             (EQUAL QUEUE (NULL)))
```

VC 'SENDER#5.10.2' is trivially true:

```
   (IMPLIES (EQUAL NEXT 0)
            (EQUAL NEXT 0))

-------------------

VC 'SENDER#5.10.3' is trivially true:

   (IMPLIES (EQUAL NEXT 0)
            (EQUAL NEXT 0))

-------------------

VC 'SENDER#5.10.4.1' is trivially true:

   (IMPLIES (SEQP (FLATTEN PKT.OUT))
            (SEQP (FLATTEN PKT.OUT)))

-------------------

VC 'SENDER#5.10.4.2' is trivially true:

   (IMPLIES (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                     (NULL)
                                     (FLATTEN PKT.OUT))))
                   (SUB1 NEXT))
            (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                     (NULL)
                                     (FLATTEN PKT.OUT))))
                   (SUB1 NEXT)))

-------------------

VC 'SENDER#5.10.4.3.1' is trivially true:

   (IMPLIES (EQUAL (DOM (LST QUEUE))
                   (SUB1 NEXT))
            (EQUAL (DOM (LST QUEUE))
                   (SUB1 NEXT)))

-------------------

VC 'SENDER#5.10.4.3.2' is trivially true:
   (IMPLIES (SEQP QUEUE)
            (SEQP QUEUE))

--------------------

VC 'SENDER#5.10.4.3.3' is trivially true:

   (IMPLIES (SEQP QUEUE)
            (SEQP QUEUE))

--------------------

VC 'SENDER#5.10.4.3.4' is trivially true:

   (IMPLIES T T)


----------------------------------------


Path conditions for VC 'SENDER#6':

   (AND (NOT TIMING)
        (ACKNOWLP ACK))

Variable substitution:

   SOURCE       :=  SOURCE
   ACK.IN       :=  (APR ACK.IN ACK)
   RECEIPTS     :=  RECEIPTS
   PKT.OUT      :=  PKT.OUT
   UNACK        :=  UNACK
   NEXT         :=  NEXT
   EDGE         :=  (MAX EDGE (EDGE ACK))
   QUEUE        :=  QUEUE
   TIMING       :=  TIMING
   TO.TIME      :=  TO.TIME

VC 'SENDER#6' is trivially true:

   (IMPLIES
     (AND
       (NOT TIMING)
       (ACKNOWLP ACK)
   (PMAPP QUEUE)
   (NUMBERP NEXT)
   (ALL (FAPPLY (QUOTE MPAIRP)
                (FLATTEN PKT.OUT)))
   (ALL (FAPPLY (QUOTE CONTIG)
                PKT.OUT))
   (CONTIG QUEUE)
   (CONSISTENT (QUOTE DOM)
               (QUOTE EQUAL)
               (FLATTEN PKT.OUT))
   (FOLLOWS QUEUE (FLATTEN PKT.OUT))
   (EQUAL (FLATTEN SOURCE)
          (RANGE (REDUCE (QUOTE WITH)
                         (NULL)
                         (FLATTEN PKT.OUT))))
   (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
          (SUB1 NEXT))
   (IF (EQUAL NEXT 0)
       (AND (EQUAL (FLATTEN PKT.OUT)
                   (NULL))
            (EQUAL QUEUE (NULL)))
       (AND (SEQP (FLATTEN PKT.OUT))
            (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                     (NULL)
                                     (FLATTEN PKT.OUT))))
                   (SUB1 NEXT))
            (IF (SEQP QUEUE)
                (EQUAL (DOM (LST QUEUE))
                       (SUB1 NEXT))
                T))))
 (AND
   (PMAPP QUEUE)
   (NUMBERP NEXT)
   (ALL (FAPPLY (QUOTE MPAIRP)
                (FLATTEN PKT.OUT)))
   (ALL (FAPPLY (QUOTE CONTIG)
                PKT.OUT))
   (CONTIG QUEUE)
   (CONSISTENT (QUOTE DOM)
               (QUOTE EQUAL)
               (FLATTEN PKT.OUT))
   (FOLLOWS QUEUE (FLATTEN PKT.OUT))
   (EQUAL (FLATTEN SOURCE)
          (RANGE (REDUCE (QUOTE WITH)
                             (NULL)
                             (FLATTEN PKT.OUT))))
      (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
             (SUB1 NEXT))
      (IF (EQUAL NEXT 0)
          (AND (EQUAL (FLATTEN PKT.OUT)
                      (NULL))
               (EQUAL QUEUE (NULL)))
          (AND (SEQP (FLATTEN PKT.OUT))
               (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                        (NULL)
                                        (FLATTEN PKT.OUT))))
                      (SUB1 NEXT))
               (IF (SEQP QUEUE)
                   (EQUAL (DOM (LST QUEUE))
                          (SUB1 NEXT))
                   T)))))
```

```
----------------------------------------

Path conditions for VC 'SENDER#7':

   (AND (NOT (LESSP UNACK (ACKNO ACK)))
        TIMING
        (ACKNOWLP ACK))

Variable substitution:

   SOURCE       :=  SOURCE
   ACK.IN       :=  (APR ACK.IN ACK)
   RECEIPTS     :=  RECEIPTS
   PKT.OUT      :=  PKT.OUT
   UNACK        :=  UNACK
   NEXT         :=  NEXT
   EDGE         :=  (MAX EDGE (EDGE ACK))
   QUEUE        :=  QUEUE
   TIMING       :=  TIMING
   TO.TIME      :=  TO.TIME

VC 'SENDER#7' is trivially true:

   (IMPLIES
(AND
  (NOT (LESSP UNACK (ACKNO ACK)))
  TIMING
  (ACKNOWLP ACK)
  (PMAPP QUEUE)
  (NUMBERP NEXT)
  (ALL (FAPPLY (QUOTE MPAIRP)
               (FLATTEN PKT.OUT)))
  (ALL (FAPPLY (QUOTE CONTIG)
               PKT.OUT))
  (CONTIG QUEUE)
  (CONSISTENT (QUOTE DOM)
              (QUOTE EQUAL)
              (FLATTEN PKT.OUT))
  (FOLLOWS QUEUE (FLATTEN PKT.OUT))
  (EQUAL (FLATTEN SOURCE)
         (RANGE (REDUCE (QUOTE WITH)
                        (NULL)
                        (FLATTEN PKT.OUT))))
  (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
         (SUB1 NEXT))
  (IF (EQUAL NEXT 0)
      (AND (EQUAL (FLATTEN PKT.OUT)
                  (NULL))
           (EQUAL QUEUE (NULL)))
      (AND (SEQP (FLATTEN PKT.OUT))
           (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                    (NULL)
                                    (FLATTEN PKT.OUT))))
                  (SUB1 NEXT))
           (IF (SEQP QUEUE)
               (EQUAL (DOM (LST QUEUE))
                      (SUB1 NEXT))
               T))))
(AND
  (PMAPP QUEUE)
  (NUMBERP NEXT)
  (ALL (FAPPLY (QUOTE MPAIRP)
               (FLATTEN PKT.OUT)))
  (ALL (FAPPLY (QUOTE CONTIG)
               PKT.OUT))
  (CONTIG QUEUE)
  (CONSISTENT (QUOTE DOM)
              (QUOTE EQUAL)
```

```
                        (FLATTEN PKT.OUT))
            (FOLLOWS QUEUE (FLATTEN PKT.OUT))
            (EQUAL (FLATTEN SOURCE)
                   (RANGE (REDUCE (QUOTE WITH)
                                  (NULL)
                                  (FLATTEN PKT.OUT))))
            (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                   (SUB1 NEXT))
            (IF (EQUAL NEXT 0)
                (AND (EQUAL (FLATTEN PKT.OUT)
                            (NULL))
                     (EQUAL QUEUE (NULL)))
                (AND (SEQP (FLATTEN PKT.OUT))
                     (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                              (NULL)
                                              (FLATTEN PKT.OUT))))
                            (SUB1 NEXT))
                     (IF (SEQP QUEUE)
                         (EQUAL (DOM (LST QUEUE))
                                (SUB1 NEXT))
                         T)))))
```

```
Path conditions for VC 'SENDER#8':

   (AND (LESSP UNACK (ACKNO ACK))
        (NOT (EQUAL NEXT (ACKNO ACK)))
        TIMING
        (ACKNOWLP ACK))

Variable substitution:

   SOURCE        :=  SOURCE
   ACK.IN        :=  (APR ACK.IN ACK)
   RECEIPTS      :=  (APR RECEIPTS (DIFFERENCE (ACKNO ACK)
                                               UNACK))
   PKT.OUT       :=  PKT.OUT
   UNACK         :=  (ACKNO ACK)
   NEXT          :=  NEXT
   EDGE          :=  (MAX EDGE (EDGE ACK))
   QUEUE         :=  (UPPER QUEUE (ACKNO ACK))
   TIMING        :=  TIMING
   TO.TIME       :=  TO.TIME

Instantiated assertion:

   (AND
     (PMAPP (UPPER QUEUE (ACKNO ACK)))
     (NUMBERP NEXT)
     (ALL (FAPPLY (QUOTE MPAIRP)
                  (FLATTEN PKT.OUT)))
     (ALL (FAPPLY (QUOTE CONTIG)
                  PKT.OUT))
     (CONTIG (UPPER QUEUE (ACKNO ACK)))
     (CONSISTENT (QUOTE DOM)
                 (QUOTE EQUAL)
                 (FLATTEN PKT.OUT))
     (FOLLOWS (UPPER QUEUE (ACKNO ACK))
              (FLATTEN PKT.OUT))
     (EQUAL (FLATTEN SOURCE)
            (RANGE (REDUCE (QUOTE WITH)
                           (NULL)
                           (FLATTEN PKT.OUT))))
     (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
            (SUB1 NEXT))
     (IF (EQUAL NEXT 0)
         (AND (EQUAL (FLATTEN PKT.OUT)
                     (NULL))
              (EQUAL (UPPER QUEUE (ACKNO ACK))
                     (NULL)))
         (AND (SEQP (FLATTEN PKT.OUT))
              (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (FLATTEN PKT.OUT))))
                     (SUB1 NEXT))
              (IF (SEQP (UPPER QUEUE (ACKNO ACK)))
                  (EQUAL (DOM (LST (UPPER QUEUE (ACKNO ACK))))
                         (SUB1 NEXT))
                  T))))

--------------------

Generating new VC 'SENDER#8.1':

   (IMPLIES (PMAPP QUEUE)
            (PMAPP (UPPER QUEUE (ACKNO ACK))))

VC not proved, being written.

-------------------

VC 'SENDER#8.2' is trivially true:

   (IMPLIES (NUMBERP NEXT)
            (NUMBERP NEXT))

-------------------

VC 'SENDER#8.3' is trivially true:

   (IMPLIES (ALL (FAPPLY (QUOTE MPAIRP)
                         (FLATTEN PKT.OUT)))
            (ALL (FAPPLY (QUOTE MPAIRP)
                         (FLATTEN PKT.OUT))))

-------------------

VC 'SENDER#8.4' is trivially true:

   (IMPLIES (ALL (FAPPLY (QUOTE CONTIG)
                         PKT.OUT))
            (ALL (FAPPLY (QUOTE CONTIG)
                         PKT.OUT)))

-------------------

Generating new VC 'SENDER#8.5':

   (IMPLIES (AND (CONTIG QUEUE)
                 (PMAPP QUEUE))
            (CONTIG (UPPER QUEUE (ACKNO ACK))))


VC not proved, being written.

-------------------

VC 'SENDER#8.6' is trivially true:
   (IMPLIES (CONSISTENT (QUOTE DOM)
                        (QUOTE EQUAL)
                        (FLATTEN PKT.OUT))
            (CONSISTENT (QUOTE DOM)
                        (QUOTE EQUAL)
                        (FLATTEN PKT.OUT)))

-------------------

Generating new VC 'SENDER#8.7':

   (IMPLIES (FOLLOWS QUEUE (FLATTEN PKT.OUT))
            (FOLLOWS (UPPER QUEUE (ACKNO ACK))
                     (FLATTEN PKT.OUT)))


VC not proved, being written.

-------------------

VC 'SENDER#8.8' is trivially true:

   (IMPLIES (EQUAL (FLATTEN SOURCE)
                   (RANGE (REDUCE (QUOTE WITH)
                                  (NULL)
                                  (FLATTEN PKT.OUT))))
            (EQUAL (FLATTEN SOURCE)
                   (RANGE (REDUCE (QUOTE WITH)
                                  (NULL)
                                  (FLATTEN PKT.OUT)))))

-------------------

VC 'SENDER#8.9' is trivially true:

   (IMPLIES (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                   (SUB1 NEXT))
            (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                   (SUB1 NEXT)))

-------------------

VC 'SENDER#8.10.1.1' is trivially true:
   (IMPLIES (EQUAL (FLATTEN PKT.OUT)
                   (NULL))
            (EQUAL (FLATTEN PKT.OUT)
                   (NULL)))

--------------------

Generating new VC 'SENDER#8.10.1.2':

   (IMPLIES (EQUAL QUEUE (NULL))
            (EQUAL (UPPER QUEUE (ACKNO ACK))
                   (NULL)))


VC not proved, being written.

--------------------

VC 'SENDER#8.10.2' is trivially true:

   (IMPLIES (EQUAL NEXT 0)
            (EQUAL NEXT 0))

--------------------

VC 'SENDER#8.10.3' is trivially true:

   (IMPLIES (EQUAL NEXT 0)
            (EQUAL NEXT 0))

--------------------

VC 'SENDER#8.10.4.1' is trivially true:

   (IMPLIES (SEQP (FLATTEN PKT.OUT))
            (SEQP (FLATTEN PKT.OUT)))

--------------------

VC 'SENDER#8.10.4.2' is trivially true:

   (IMPLIES (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                     (NULL)
```

```
                                    (FLATTEN PKT.OUT))))
                  (SUB1 NEXT))
           (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                    (NULL)
                                    (FLATTEN PKT.OUT))))
                  (SUB1 NEXT)))

-------------------

Generating new VC 'SENDER#8.10.4.3.1':

   (IMPLIES (AND (EQUAL (DOM (LST QUEUE))
                        (SUB1 NEXT))
                 (SEQP (UPPER QUEUE (ACKNO ACK)))
                 (PMAPP QUEUE))
            (EQUAL (DOM (LST (UPPER QUEUE (ACKNO ACK))))
                   (SUB1 NEXT)))


VC not proved, being written.

-------------------

Generating new VC 'SENDER#8.10.4.3.2':

   (IMPLIES (SEQP (UPPER QUEUE (ACKNO ACK)))
            (SEQP QUEUE))


VC not proved, being written.

-------------------

VC 'SENDER#8.10.4.3.3' is trivially true:

   (IMPLIES F (SEQP (UPPER QUEUE (ACKNO ACK))))

-------------------

VC 'SENDER#8.10.4.3.4' is trivially true:

   (IMPLIES T T)
```

```
----------------------------------------

Path conditions for VC 'SENDER#9':

   (AND (LESSP UNACK (ACKNO ACK))
        (EQUAL NEXT (ACKNO ACK))
        TIMING
        (ACKNOWLP ACK))

Variable substitution:

   SOURCE          :=  SOURCE
   ACK.IN          :=  (APR ACK.IN ACK)
   RECEIPTS        :=  (APR RECEIPTS (DIFFERENCE (ACKNO ACK)
                                                  UNACK))
   PKT.OUT         :=  PKT.OUT
   UNACK           :=  (ACKNO ACK)
   NEXT            :=  NEXT
   EDGE            :=  (MAX EDGE (EDGE ACK))
   QUEUE           :=  (NULL)
   TIMING          :=  F
   TO.TIME         :=  TO.TIME

Instantiated assertion:

   (AND (PMAPP (NULL))
        (NUMBERP NEXT)
        (ALL (FAPPLY (QUOTE MPAIRP)
                     (FLATTEN PKT.OUT)))
        (ALL (FAPPLY (QUOTE CONTIG)
                     PKT.OUT))
        (CONTIG (NULL))
        (CONSISTENT (QUOTE DOM)
                    (QUOTE EQUAL)
                    (FLATTEN PKT.OUT))
        (FOLLOWS (NULL)
                 (FLATTEN PKT.OUT))
        (EQUAL (FLATTEN SOURCE)
               (RANGE (REDUCE (QUOTE WITH)
                              (NULL)
                              (FLATTEN PKT.OUT))))
        (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
               (SUB1 NEXT))
        (IF (EQUAL NEXT 0)
            (AND (EQUAL (FLATTEN PKT.OUT)
                        (NULL))
                 (EQUAL (NULL)
                        (NULL)))
            (AND (SEQP (FLATTEN PKT.OUT))
                 (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                          (NULL)
                                          (FLATTEN PKT.OUT))))
                        (SUB1 NEXT))
                 (IF (SEQP (NULL))
                     (EQUAL (DOM (LST (NULL)))
                            (SUB1 NEXT))
                     T))))

--------------------

Generating new VC 'SENDER#9.1':

   (PMAPP (NULL))

This conjecture simplifies, expanding the definition of PMAPP, to:

      T.

Q.E.D.

VC has been proved.

Compute time:    5.388 sec,   GC time:    0.000 sec.

--------------------

VC 'SENDER#9.2' is trivially true:

   (IMPLIES (NUMBERP NEXT)
            (NUMBERP NEXT))

--------------------

VC 'SENDER#9.3' is trivially true:

   (IMPLIES (ALL (FAPPLY (QUOTE MPAIRP)
                         (FLATTEN PKT.OUT)))
```

```
          (ALL (FAPPLY (QUOTE MPAIRP)
                       (FLATTEN PKT.OUT))))

------------------

VC 'SENDER#9.4' is trivially true:

   (IMPLIES (ALL (FAPPLY (QUOTE CONTIG)
                         PKT.OUT))
            (ALL (FAPPLY (QUOTE CONTIG)
                         PKT.OUT)))

------------------

Generating new VC 'SENDER#9.5':

   (CONTIG (NULL))

This formula can be simplified, using the abbreviation CONTIG, to:

      (CONSECP (DOMAIN (QUOTE (1QUOTE NULL)))),

which simplifies, expanding the functions DOMAIN and CONSECP, to:

      T.

Q.E.D.

VC has been proved.

Compute time:    8.849 sec,   GC time:    0.000 sec.

------------------

VC 'SENDER#9.6' is trivially true:

   (IMPLIES (CONSISTENT (QUOTE DOM)
                        (QUOTE EQUAL)
                        (FLATTEN PKT.OUT))
            (CONSISTENT (QUOTE DOM)
                        (QUOTE EQUAL)
                        (FLATTEN PKT.OUT)))

------------------
```

```
Generating new VC 'SENDER#9.7':

   (FOLLOWS (NULL)
            (FLATTEN PKT.OUT))

This conjecture simplifies, expanding the functions SEQP and
FOLLOWS, to:

      T.

Q.E.D.

VC has been proved.

Compute time:   12.077 sec,   GC time:    0.000 sec.

-------------------

VC 'SENDER#9.8' is trivially true:

   (IMPLIES (EQUAL (FLATTEN SOURCE)
                   (RANGE (REDUCE (QUOTE WITH)
                                  (NULL)
                                  (FLATTEN PKT.OUT))))
            (EQUAL (FLATTEN SOURCE)
                   (RANGE (REDUCE (QUOTE WITH)
                                  (NULL)
                                  (FLATTEN PKT.OUT)))))

-------------------

VC 'SENDER#9.9' is trivially true:

   (IMPLIES (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                   (SUB1 NEXT))
            (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                   (SUB1 NEXT)))

-------------------

VC 'SENDER#9.10.1.1' is trivially true:

   (IMPLIES (EQUAL (FLATTEN PKT.OUT)
                    (NULL))
              (EQUAL (FLATTEN PKT.OUT)
                     (NULL)))

--------------------

VC 'SENDER#9.10.1.2' is trivially true:

   (IMPLIES (EQUAL QUEUE (NULL))
            T)

--------------------

VC 'SENDER#9.10.2' is trivially true:

   (IMPLIES (EQUAL NEXT 0)
            (EQUAL NEXT 0))

--------------------

VC 'SENDER#9.10.3' is trivially true:

   (IMPLIES (EQUAL NEXT 0)
            (EQUAL NEXT 0))

--------------------

VC 'SENDER#9.10.4.1' is trivially true:

   (IMPLIES (SEQP (FLATTEN PKT.OUT))
            (SEQP (FLATTEN PKT.OUT)))

--------------------

VC 'SENDER#9.10.4.2' is trivially true:

   (IMPLIES (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                     (NULL)
                                     (FLATTEN PKT.OUT))))
                   (SUB1 NEXT))
            (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                     (NULL)
                                     (FLATTEN PKT.OUT))))
                   (SUB1 NEXT)))
```

```
-------------------

Generating new VC 'SENDER#9.10.4.3.1':

   (IMPLIES (SEQP (NULL))
            F)

This formula can be simplified, using the abbreviation NOT, to:

      (NOT (SEQP (QUOTE (1QUOTE NULL)))),

which we simplify, clearly, to:

      T.

Q.E.D.

VC has been proved.

Compute time:    6.035 sec,   GC time:    0.000 sec.

-------------------

VC 'SENDER#9.10.4.3.1' subsumes VC 'SENDER#9.10.4.3.2':

   (IMPLIES (AND (NOT (EQUAL (DOM (LST (NULL)))
                             (SUB1 NEXT)))
                 T
                 (SEQP (FLATTEN PKT.OUT))
                 (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                          (NULL)
                                          (FLATTEN PKT.OUT))))
                        (SUB1 NEXT))
                 (NOT (EQUAL NEXT 0))
                 (NOT (EQUAL NEXT 0))
                 (PMAPP QUEUE)
                 (NUMBERP NEXT)
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.OUT)))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              PKT.OUT))
                 (CONTIG QUEUE)
                 (CONSISTENT (QUOTE DOM)
                                    (QUOTE EQUAL)
                                    (FLATTEN PKT.OUT))
                        (FOLLOWS QUEUE (FLATTEN PKT.OUT))
                        (EQUAL (FLATTEN SOURCE)
                               (RANGE (REDUCE (QUOTE WITH)
                                              (NULL)
                                              (FLATTEN PKT.OUT))))
                        (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                               (SUB1 NEXT))
                        (LESSP UNACK (ACKNO ACK))
                        (EQUAL NEXT (ACKNO ACK))
                        (ACKNOWLP ACK)
                        (SEQP (NULL)))
                   (SEQP QUEUE))

--------------------

VC 'SENDER#9.10.4.3.3' is trivially true:

   (IMPLIES F (SEQP (NULL)))

--------------------

VC 'SENDER#9.10.4.3.4' is trivially true:

   (IMPLIES T T)


----------------------------------------


Path conditions for VC 'SENDER#10':

   (AND F (ACKNOWLP ACK))

Variable substitution:

   SOURCE        :=  SOURCE
   ACK.IN        :=  (APR ACK.IN ACK)
   RECEIPTS      :=  RECEIPTS
   PKT.OUT       :=  PKT.OUT
   UNACK         :=  UNACK
   NEXT          :=  NEXT
   EDGE          :=  EDGE
   QUEUE          :=  QUEUE
   TIMING         :=  TIMING
   TO.TIME        :=  TO.TIME

VC 'SENDER#10' is trivially true:

   (IMPLIES
     (AND
       F
       (ACKNOWLP ACK)
       (PMAPP QUEUE)
       (NUMBERP NEXT)
       (ALL (FAPPLY (QUOTE MPAIRP)
                    (FLATTEN PKT.OUT)))
       (ALL (FAPPLY (QUOTE CONTIG)
                    PKT.OUT))
       (CONTIG QUEUE)
       (CONSISTENT (QUOTE DOM)
                   (QUOTE EQUAL)
                   (FLATTEN PKT.OUT))
       (FOLLOWS QUEUE (FLATTEN PKT.OUT))
       (EQUAL (FLATTEN SOURCE)
              (RANGE (REDUCE (QUOTE WITH)
                             (NULL)
                             (FLATTEN PKT.OUT))))
       (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
              (SUB1 NEXT))
       (IF (EQUAL NEXT 0)
           (AND (EQUAL (FLATTEN PKT.OUT)
                       (NULL))
                (EQUAL QUEUE (NULL)))
           (AND (SEQP (FLATTEN PKT.OUT))
                (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                         (NULL)
                                         (FLATTEN PKT.OUT))))
                       (SUB1 NEXT))
                (IF (SEQP QUEUE)
                    (EQUAL (DOM (LST QUEUE))
                           (SUB1 NEXT))
                    T))))
     (AND
       (PMAPP QUEUE)
       (NUMBERP NEXT)
       (ALL (FAPPLY (QUOTE MPAIRP)
                  (FLATTEN PKT.OUT)))
      (ALL (FAPPLY (QUOTE CONTIG)
                   PKT.OUT))
      (CONTIG QUEUE)
      (CONSISTENT (QUOTE DOM)
                  (QUOTE EQUAL)
                  (FLATTEN PKT.OUT))
      (FOLLOWS QUEUE (FLATTEN PKT.OUT))
      (EQUAL (FLATTEN SOURCE)
             (RANGE (REDUCE (QUOTE WITH)
                            (NULL)
                            (FLATTEN PKT.OUT))))
      (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
             (SUB1 NEXT))
      (IF (EQUAL NEXT 0)
          (AND (EQUAL (FLATTEN PKT.OUT)
                      (NULL))
               (EQUAL QUEUE (NULL)))
          (AND (SEQP (FLATTEN PKT.OUT))
               (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                        (NULL)
                                        (FLATTEN PKT.OUT))))
                      (SUB1 NEXT))
               (IF (SEQP QUEUE)
                   (EQUAL (DOM (LST QUEUE))
                          (SUB1 NEXT))
                   T)))))
```

```
Path conditions for VC 'SENDER#11':

   TIMING

Variable substitution:

   SOURCE         :=   SOURCE
   ACK.IN         :=   ACK.IN
   RECEIPTS       :=   RECEIPTS
   PKT.OUT        :=   (APR PKT.OUT QUEUE)
   UNACK          :=   UNACK
   NEXT           :=   NEXT
   EDGE            :=  EDGE
   QUEUE           :=  QUEUE
   TIMING          :=  TIMING
   TO.TIME         :=  (PLUS TIME DELTA.T)
```

Instantiated assertion:

```
   (AND
     (PMAPP QUEUE)
     (NUMBERP NEXT)
     (ALL (FAPPLY (QUOTE MPAIRP)
                  (FLATTEN (APR PKT.OUT QUEUE))))
     (ALL (FAPPLY (QUOTE CONTIG)
                  (APR PKT.OUT QUEUE)))
     (CONTIG QUEUE)
     (CONSISTENT (QUOTE DOM)
                 (QUOTE EQUAL)
                 (FLATTEN (APR PKT.OUT QUEUE)))
     (FOLLOWS QUEUE (FLATTEN (APR PKT.OUT QUEUE)))
     (EQUAL (FLATTEN SOURCE)
            (RANGE (REDUCE (QUOTE WITH)
                           (NULL)
                           (FLATTEN (APR PKT.OUT QUEUE)))))
     (EQUAL (HIGHEST (DOMAIN (FLATTEN (APR PKT.OUT QUEUE))))
            (SUB1 NEXT))
     (IF (EQUAL NEXT 0)
         (AND (EQUAL (FLATTEN (APR PKT.OUT QUEUE))
                     (NULL))
              (EQUAL QUEUE (NULL)))
         (AND (SEQP (FLATTEN (APR PKT.OUT QUEUE)))
              (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (FLATTEN (APR PKT.OUT QUEUE))
                                       )))
                     (SUB1 NEXT))
              (IF (SEQP QUEUE)
                  (EQUAL (DOM (LST QUEUE))
                         (SUB1 NEXT))
                  T))))
```

Generating new VC 'SENDER#11.0.1':

```
   (IMPLIES (AND (PMAPP QUEUE)
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.OUT)))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (FLATTEN PKT.OUT))
                 (FOLLOWS QUEUE (FLATTEN PKT.OUT)))
            (EQUAL (REDUCE (QUOTE WITH)
                           (NULL)
                           (FLATTEN (APR PKT.OUT QUEUE)))
                   (REDUCE (QUOTE WITH)
                           (NULL)
                           (FLATTEN PKT.OUT))))


VC not proved, being written.

Equality substitution performed.

------------------

VC 'SENDER#11.1' is trivially true:

   (IMPLIES (PMAPP QUEUE)
            (PMAPP QUEUE))

------------------

VC 'SENDER#11.2' is trivially true:

   (IMPLIES (NUMBERP NEXT)
            (NUMBERP NEXT))

------------------

Generating new VC 'SENDER#11.3':

   (IMPLIES (AND (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.OUT)))
                 (PMAPP QUEUE))
            (ALL (FAPPLY (QUOTE MPAIRP)
                         (FLATTEN (APR PKT.OUT QUEUE)))))
```

```
VC not proved, being written.

------------------

Generating new VC 'SENDER#11.4':

   (IMPLIES (AND (ALL (FAPPLY (QUOTE CONTIG)
                              PKT.OUT))
                 (CONTIG QUEUE))
            (ALL (FAPPLY (QUOTE CONTIG)
                         (APR PKT.OUT QUEUE))))


VC not proved, being written.

------------------

VC 'SENDER#11.5' is trivially true:

   (IMPLIES (CONTIG QUEUE)
            (CONTIG QUEUE))

------------------

Generating new VC 'SENDER#11.6':

   (IMPLIES (AND (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (FLATTEN PKT.OUT))
                 (PMAPP QUEUE)
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.OUT)))
                 (FOLLOWS QUEUE (FLATTEN PKT.OUT)))
            (CONSISTENT (QUOTE DOM)
                        (QUOTE EQUAL)
                        (FLATTEN (APR PKT.OUT QUEUE))))


VC not proved, being written.

------------------

Generating new VC 'SENDER#11.7':
   (FOLLOWS QUEUE (FLATTEN (APR PKT.OUT QUEUE)))


VC not proved, being written.

-------------------

VC 'SENDER#11.8' is trivially true:

   (IMPLIES (EQUAL (FLATTEN SOURCE)
                   (RANGE (REDUCE (QUOTE WITH)
                                  (NULL)
                                  (FLATTEN PKT.OUT))))
            (EQUAL (FLATTEN SOURCE)
                   (RANGE (REDUCE (QUOTE WITH)
                                  (NULL)
                                  (FLATTEN PKT.OUT)))))

-------------------

Generating new VC 'SENDER#11.9.1':

   (IMPLIES (AND (EQUAL NEXT 0)
                 (EQUAL (FLATTEN PKT.OUT)
                        (NULL))
                 (EQUAL QUEUE (NULL)))
            (EQUAL (HIGHEST (DOMAIN (FLATTEN (APR PKT.OUT QUEUE))))
                   (SUB1 NEXT)))


VC not proved, being written.

-------------------

Generating new VC 'SENDER#11.9.2':

   (IMPLIES (AND (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                        (SUB1 NEXT))
                 (PMAPP QUEUE)
                 (NUMBERP NEXT)
                 (NOT (EQUAL NEXT 0))
                 (IF (SEQP QUEUE)
                     (EQUAL (DOM (LST QUEUE))
                            (SUB1 NEXT))
```

```
                T))
         (EQUAL (HIGHEST (DOMAIN (FLATTEN (APR PKT.OUT QUEUE))))
                (SUB1 NEXT)))


VC not proved, being written.

--------------------

Generating new VC 'SENDER#11.10.1.1':

   (IMPLIES (AND (EQUAL (FLATTEN PKT.OUT)
                        (NULL))
                 (EQUAL QUEUE (NULL)))
            (EQUAL (FLATTEN (APR PKT.OUT QUEUE))
                   (NULL)))


VC not proved, being written.

--------------------

VC 'SENDER#11.10.1.2' is trivially true:

   (IMPLIES (EQUAL QUEUE (NULL))
            (EQUAL QUEUE (NULL)))

--------------------

VC 'SENDER#11.10.2' is trivially true:

   (IMPLIES (EQUAL NEXT 0)
            (EQUAL NEXT 0))

--------------------

VC 'SENDER#11.10.3' is trivially true:

   (IMPLIES (EQUAL NEXT 0)
            (EQUAL NEXT 0))

--------------------

Generating new VC 'SENDER#11.10.4.1':
   (IMPLIES (SEQP (FLATTEN PKT.OUT))
            (SEQP (FLATTEN (APR PKT.OUT QUEUE))))


VC not proved, being written.

------------------

VC 'SENDER#11.10.4.2' is trivially true:

   (IMPLIES (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                     (NULL)
                                     (FLATTEN PKT.OUT))))
                   (SUB1 NEXT))
            (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                     (NULL)
                                     (FLATTEN PKT.OUT))))
                   (SUB1 NEXT)))

------------------

VC 'SENDER#11.10.4.3.1' is trivially true:

   (IMPLIES (EQUAL (DOM (LST QUEUE))
                   (SUB1 NEXT))
            (EQUAL (DOM (LST QUEUE))
                   (SUB1 NEXT)))

------------------

VC 'SENDER#11.10.4.3.2' is trivially true:

   (IMPLIES (SEQP QUEUE)
            (SEQP QUEUE))

------------------

VC 'SENDER#11.10.4.3.3' is trivially true:

   (IMPLIES (SEQP QUEUE)
            (SEQP QUEUE))

------------------
```

```
VC 'SENDER#11.10.4.3.4' is trivially true:

   (IMPLIES T T)

----------------------------------------



The totals are:     64 trivially true VCs,
                    7 subsumed VCs,
                    5 proved VCs, and
                    37 generated VCs.

Compute time:   133.636 seconds,    GC time:     7.406 seconds.


VC reduction log for: 'RECEIVER'                  24-Jun-82 08:48:19


Assertion function:

   (RECEIVER.INT CREDITS PKT.IN SINK ACK.OUT NEXT EDGE QUEUE)
      =
   (AND
     (PMAPP QUEUE)
     (NUMBERP NEXT)
     (NUMBERP EDGE)
     (IMPLIES
       (AND (CONSISTENT (QUOTE DOM)
                        (QUOTE EQUAL)
                        (FLATTEN PKT.IN))
            (ALL (FAPPLY (QUOTE CONTIG)
                         PKT.IN)))
       (AND
         (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (FLATTEN PKT.IN))
                               (ADD1 NEXT)))
         (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                    (NULL)
                                    (FLATTEN PKT.IN)))
                    NEXT))
        (ALL (FAPPLY (QUOTE MPAIRP)
                     (FLATTEN PKT.IN)))
        (IF (IN 0 (DOMAIN (FLATTEN PKT.IN)))
            (AND (LESSP 0 NEXT)
                 (EQUAL (FLATTEN SINK)
                        (RANGE (LOWER (REDUCE (QUOTE WITH)
                                              (NULL)
                                              (FLATTEN PKT.IN))
                                      (SUB1 NEXT))))))
            (AND (EQUAL NEXT 0)
                 (EQUAL SINK (NULL)))))))
```

```
----------------------------------------
```

Path conditions for VC 'RECEIVER#3':

```
   (NUMBERP CRED)
```

Variable substitution:

```
   CREDITS       :=  (APR CREDITS CRED)
   PKT.IN        :=  PKT.IN
   SINK          :=  SINK
   ACK.OUT       :=  (APR ACK.OUT (ACKNOWL NEXT (PLUS EDGE CRED)))
   NEXT          :=  NEXT
   EDGE          :=  (PLUS EDGE CRED)
   QUEUE         :=  QUEUE
```

Instantiated assertion:

```
   (AND
     (PMAPP QUEUE)
     (NUMBERP NEXT)
     (NUMBERP (PLUS EDGE CRED))
     (IMPLIES
       (AND (CONSISTENT (QUOTE DOM)
                        (QUOTE EQUAL)
                        (FLATTEN PKT.IN))
            (ALL (FAPPLY (QUOTE CONTIG)
                         PKT.IN)))
       (AND
        (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (FLATTEN PKT.IN))
                              (ADD1 NEXT)))
        (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                   (NULL)
                                   (FLATTEN PKT.IN)))
                    NEXT))
        (ALL (FAPPLY (QUOTE MPAIRP)
                     (FLATTEN PKT.IN)))
        (IF (IN 0 (DOMAIN (FLATTEN PKT.IN)))
            (AND (LESSP 0 NEXT)
                 (EQUAL (FLATTEN SINK)
                        (RANGE (LOWER (REDUCE (QUOTE WITH)
                                              (NULL)
                                              (FLATTEN PKT.IN))
                                      (SUB1 NEXT)))))
            (AND (EQUAL NEXT 0)
                 (EQUAL SINK (NULL)))))))

-------------------

VC 'RECEIVER#3.1' is trivially true:

   (IMPLIES (PMAPP QUEUE)
            (PMAPP QUEUE))

-------------------

VC 'RECEIVER#3.2' is trivially true:

   (IMPLIES (NUMBERP NEXT)
            (NUMBERP NEXT))

-------------------

Generating new VC 'RECEIVER#3.3':

   (NUMBERP (PLUS EDGE CRED))

This conjecture simplifies, clearly, to:

      T.
```

```
Q.E.D.

VC has been proved.

Compute time:    2.684 sec,   GC time:    0.000 sec.

------------------

VC 'RECEIVER#3.4.1.1' is trivially true:

   (IMPLIES (CONSISTENT (QUOTE DOM)
                        (QUOTE EQUAL)
                        (FLATTEN PKT.IN))
            (CONSISTENT (QUOTE DOM)
                        (QUOTE EQUAL)
                        (FLATTEN PKT.IN)))

------------------

VC 'RECEIVER#3.4.1.2' is trivially true:

   (IMPLIES (ALL (FAPPLY (QUOTE CONTIG)
                         PKT.IN))
            (ALL (FAPPLY (QUOTE CONTIG)
                         PKT.IN)))

------------------

VC 'RECEIVER#3.4.2.1' is trivially true:

   (IMPLIES (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                          (NULL)
                                          (FLATTEN PKT.IN))
                                  (ADD1 NEXT)))
            (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                          (NULL)
                                          (FLATTEN PKT.IN))
                                  (ADD1 NEXT))))

------------------

VC 'RECEIVER#3.4.2.2' is trivially true:

   (IMPLIES (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                        (NULL)
                                        (FLATTEN PKT.IN)))
                          NEXT))
                (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                        (NULL)
                                        (FLATTEN PKT.IN)))
                          NEXT)))
```

VC 'RECEIVER#3.4.2.3' is trivially true:

```
   (IMPLIES (ALL (FAPPLY (QUOTE MPAIRP)
                         (FLATTEN PKT.IN)))
            (ALL (FAPPLY (QUOTE MPAIRP)
                         (FLATTEN PKT.IN))))
```

VC 'RECEIVER#3.4.2.4.1.1' is trivially true:

```
   (IMPLIES (LESSP 0 NEXT)
            (LESSP 0 NEXT))
```

VC 'RECEIVER#3.4.2.4.1.2' is trivially true:

```
   (IMPLIES (EQUAL (FLATTEN SINK)
                   (RANGE (LOWER (REDUCE (QUOTE WITH)
                                         (NULL)
                                         (FLATTEN PKT.IN))
                                 (SUB1 NEXT))))
            (EQUAL (FLATTEN SINK)
                   (RANGE (LOWER (REDUCE (QUOTE WITH)
                                         (NULL)
                                         (FLATTEN PKT.IN))
                                 (SUB1 NEXT)))))
```

VC 'RECEIVER#3.4.2.4.2' is trivially true:

```
   (IMPLIES (IN 0 (DOMAIN (FLATTEN PKT.IN)))
                                     (FLATTEN PKT.OUT))))
               (IN 0 (DOMAIN (FLATTEN PKT.IN)))
               (INITIAL (FLATTEN SINK)
                        (RANGE (CONSEC (REDUCE (QUOTE WITH)
                                               (QUOTE (1QUOTE NULL))
                                               (FLATTEN PKT.IN)))))
               (FOLLOWS PKT.IN PKT.OUT)
               (ALL (FAPPLY (QUOTE PMAPP) PKT.OUT)))
          (INITIAL (FLATTEN SINK)
                   (FLATTEN SOURCE))).
```

We use the above equality hypothesis by substituting:

```
      (RANGE (REDUCE (QUOTE WITH)
                     (QUOTE (1QUOTE NULL))
                     (FLATTEN PKT.OUT)))
```

for (FLATTEN SOURCE) and keeping the equality hypothesis. The result is:$$

```
      (IMPLIES
          (AND (CONSISTENT (QUOTE DOM)
                           (QUOTE EQUAL)
                           (FLATTEN PKT.OUT))
               (ALL (FAPPLY (QUOTE CONTIG) PKT.OUT))
               (EQUAL (FLATTEN SOURCE)
                      (RANGE (REDUCE (QUOTE WITH)
                                     (QUOTE (1QUOTE NULL))
                                     (FLATTEN PKT.OUT))))
               (IN 0 (DOMAIN (FLATTEN PKT.IN)))
               (INITIAL (FLATTEN SINK)
                        (RANGE (CONSEC (REDUCE (QUOTE WITH)
                                               (QUOTE (1QUOTE NULL))
                                               (FLATTEN PKT.IN)))))
               (FOLLOWS PKT.IN PKT.OUT)
               (ALL (FAPPLY (QUOTE PMAPP) PKT.OUT)))
          (INITIAL (FLATTEN SINK)
                   (RANGE (REDUCE (QUOTE WITH)
                                  (QUOTE (1QUOTE NULL))
                                  (FLATTEN PKT.OUT))))),
```

which further simplifies, applying the lemmas INITIAL.RANGE, PMAPP.REDUCE.WITH, IN.DOMAIN.REDUCE.WITH, ALL.FAPPLY.FOLLOWS, FOLLOWS.REDUCE.WITH, FOLLOWS.FLATTEN, ALL.FAPPLY.MPAIRP.FLATTEN, INITIAL.CONSEC.FOLLOWS, and INITIAL.TRANS, and expanding the function PMAPP, to:

```
         (IN 0 (DOMAIN (FLATTEN PKT.IN))))

--------------------

VC 'RECEIVER#3.4.2.4.3' is trivially true:

   (IMPLIES (IN 0 (DOMAIN (FLATTEN PKT.IN)))
            (IN 0 (DOMAIN (FLATTEN PKT.IN))))

--------------------

VC 'RECEIVER#3.4.2.4.4.1' is trivially true:

   (IMPLIES (EQUAL NEXT 0)
            (EQUAL NEXT 0))

--------------------

VC 'RECEIVER#3.4.2.4.4.2' is trivially true:

   (IMPLIES (EQUAL SINK (NULL))
            (EQUAL SINK (NULL)))


----------------------------------------


Path conditions for VC 'RECEIVER#4':

   (AND F (NUMBERP CRED))

Variable substitution:

   CREDITS        :=  (APR CREDITS CRED)
   PKT.IN         :=  PKT.IN
   SINK           :=  SINK
   ACK.OUT        :=  ACK.OUT
   NEXT           :=  NEXT
   EDGE           :=  EDGE
   QUEUE          :=  QUEUE

VC 'RECEIVER#4' is trivially true:

   (IMPLIES
(AND
  F
  (NUMBERP CRED)
  (PMAPP QUEUE)
  (NUMBERP NEXT)
  (NUMBERP EDGE)
  (IMPLIES
    (AND (CONSISTENT (QUOTE DOM)
                     (QUOTE EQUAL)
                     (FLATTEN PKT.IN))
         (ALL (FAPPLY (QUOTE CONTIG)
                      PKT.IN)))
    (AND
      (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                    (NULL)
                                    (FLATTEN PKT.IN))
                            (ADD1 NEXT)))
      (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                 (NULL)
                                 (FLATTEN PKT.IN)))
                  NEXT))
      (ALL (FAPPLY (QUOTE MPAIRP)
                   (FLATTEN PKT.IN)))
      (IF (IN 0 (DOMAIN (FLATTEN PKT.IN)))
          (AND (LESSP 0 NEXT)
               (EQUAL (FLATTEN SINK)
                      (RANGE (LOWER (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN PKT.IN))
                                    (SUB1 NEXT)))))
          (AND (EQUAL NEXT 0)
               (EQUAL SINK (NULL)))))))
(AND
  (PMAPP QUEUE)
  (NUMBERP NEXT)
  (NUMBERP EDGE)
  (IMPLIES
    (AND (CONSISTENT (QUOTE DOM)
                     (QUOTE EQUAL)
                     (FLATTEN PKT.IN))
         (ALL (FAPPLY (QUOTE CONTIG)
                      PKT.IN)))
    (AND
      (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                                            (NULL)
                                                            (FLATTEN PKT.IN))
                                                  (ADD1 NEXT)))
                      (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                                        (NULL)
                                                        (FLATTEN PKT.IN)))
                                     NEXT))
                      (ALL (FAPPLY (QUOTE MPAIRP)
                                       (FLATTEN PKT.IN)))
                      (IF (IN 0 (DOMAIN (FLATTEN PKT.IN)))
                           (AND (LESSP 0 NEXT)
                                 (EQUAL (FLATTEN SINK)
                                           (RANGE (LOWER (REDUCE (QUOTE WITH)
                                                                           (NULL)
                                                                           (FLATTEN PKT.IN))
                                                             (SUB1 NEXT)))))
                           (AND (EQUAL NEXT 0)
                                 (EQUAL SINK (NULL))))))))
```

```
----------------------------------------
```

Path conditions for VC 'RECEIVER#5':

```
   (AND (LESSP (DOM (LST PKT))
                NEXT)
        (SEQP PKT)
        (PMAPP PKT))
```

Variable substitution:

```
   CREDITS      :=  CREDITS
   PKT.IN       :=  (APR PKT.IN PKT)
   SINK         :=  SINK
   ACK.OUT      :=  (APR ACK.OUT (ACKNOWL NEXT EDGE))
   NEXT         :=  NEXT
   EDGE         :=  EDGE
   QUEUE        :=  QUEUE
```

Instantiated assertion:

```
   (AND
     (PMAPP QUEUE)
```

```
    (NUMBERP NEXT)
    (NUMBERP EDGE)
    (IMPLIES
      (AND (CONSISTENT (QUOTE DOM)
                       (QUOTE EQUAL)
                       (FLATTEN (APR PKT.IN PKT)))
           (ALL (FAPPLY (QUOTE CONTIG)
                        (APR PKT.IN PKT))))
      (AND
        (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (FLATTEN (APR PKT.IN PKT)))
                              (ADD1 NEXT)))
        (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                   (NULL)
                                   (FLATTEN (APR PKT.IN PKT))))
                    NEXT))
        (ALL (FAPPLY (QUOTE MPAIRP)
                     (FLATTEN (APR PKT.IN PKT))))
        (IF
          (IN 0 (DOMAIN (FLATTEN (APR PKT.IN PKT))))
          (AND (LESSP 0 NEXT)
               (EQUAL (FLATTEN SINK)
                      (RANGE (LOWER (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN (APR PKT.IN
                                                          PKT)))
                                    (SUB1 NEXT)))))
          (AND (EQUAL NEXT 0)
               (EQUAL SINK (NULL)))))))
```

Generating new VC 'RECEIVER#5.0.1':

```
   (EQUAL (FLATTEN (APR PKT.IN PKT))
          (JOIN (FLATTEN PKT.IN)
                PKT))
```

This conjecture simplifies, applying LST.APR and NLST.APR, and expanding the definition of FLATTEN, to:

```
      T.
```

```
Q.E.D.

VC has been proved.

Compute time:     5.477 sec,   GC time:     0.000 sec.

Equality substitution performed.

-------------------

Generating new VC 'RECEIVER#5.0.2.1':

   (IMPLIES (AND (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (NOT (CONSISTENT (QUOTE DOM)
                                  (QUOTE EQUAL)
                                  (FLATTEN PKT.IN))))
            F)


VC not proved, being written.

-------------------

Generating new VC 'RECEIVER#5.0.2.2':

   (IMPLIES (AND (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT)))
                 (NOT (ALL (FAPPLY (QUOTE CONTIG)
                                   PKT.IN))))
            F)


VC not proved, being written.

-------------------

Generating new VC 'RECEIVER#5.0.2.3':

   (IMPLIES (AND (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                    PKT))
                  (NUMBERP NEXT)
                  (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                             (NULL)
                                             (FLATTEN PKT.IN)))
                              NEXT))
                  (ALL (FAPPLY (QUOTE MPAIRP)
                               (FLATTEN PKT.IN)))
                  (IN 0 (DOMAIN (FLATTEN PKT.IN)))
                  (LESSP 0 NEXT)
                  (LESSP (DOM (LST PKT))
                         NEXT)
                  (PMAPP PKT))
             (EQUAL (REDUCE (QUOTE WITH)
                            (NULL)
                            (JOIN (FLATTEN PKT.IN)
                                  PKT))
                    (REDUCE (QUOTE WITH)
                            (NULL)
                            (FLATTEN PKT.IN))))
```

VC not proved, being written.

To be proved with lemmas (FOLLOWS.REDUCE.WITH.CONSISTENT)

Generating new VC 'RECEIVER#5.0.2.4':

```
   (IMPLIES (AND (EQUAL NEXT 0)
                 (LESSP (DOM (LST PKT))
                        NEXT))
            F)
```

This formula can be simplified, using the abbreviations NOT and OR, to the goal:

```
      (IMPLIES (EQUAL NEXT 0)
               (NOT (LESSP (DOM (LST PKT)) NEXT))).
```

This simplifies, using linear arithmetic, to:

```
        T.

Q.E.D.

VC has been proved.

Compute time:    8.980 sec,   GC time:    0.000 sec.

-------------------

VC 'RECEIVER#5.1' is trivially true:

   (IMPLIES (PMAPP QUEUE)
            (PMAPP QUEUE))

-------------------

VC 'RECEIVER#5.2' is trivially true:

   (IMPLIES (NUMBERP NEXT)
            (NUMBERP NEXT))

-------------------

VC 'RECEIVER#5.3' is trivially true:

   (IMPLIES (NUMBERP EDGE)
            (NUMBERP EDGE))

-------------------

VC 'RECEIVER#5.0.2.1' subsumes VC 'RECEIVER#5.4.1.1':

   (IMPLIES
     (AND
       (ALL (FAPPLY (QUOTE CONTIG)
                    (APR PKT.IN PKT)))
       (OR
         (NOT (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (JOIN (FLATTEN PKT.IN)
                                                  PKT))
                                    (ADD1 NEXT))))
         (LESSP (REACH (REDUCE (QUOTE WITH)
                                   (NULL)
                                   (JOIN (FLATTEN PKT.IN)
                                         PKT)))
                   NEXT)
          (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                               (JOIN (FLATTEN PKT.IN)
                                     PKT))))
          (IF
            (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                                   PKT)))
            (OR
              (NOT (LESSP 0 NEXT))
              (NOT
                (EQUAL (FLATTEN SINK)
                       (RANGE (LOWER (REDUCE (QUOTE WITH)
                                             (NULL)
                                             (JOIN (FLATTEN PKT.IN)
                                                   PKT))
                                     (SUB1 NEXT))))))
            (OR (NOT (EQUAL NEXT 0))
                (NOT (EQUAL SINK (NULL))))))
        (PMAPP QUEUE)
        (NUMBERP NEXT)
        (NUMBERP EDGE)
        (LESSP (DOM (LST PKT))
               NEXT)
        (SEQP PKT)
        (PMAPP PKT)
        (CONSISTENT (QUOTE DOM)
                    (QUOTE EQUAL)
                    (JOIN (FLATTEN PKT.IN)
                          PKT)))
      (CONSISTENT (QUOTE DOM)
                  (QUOTE EQUAL)
                  (FLATTEN PKT.IN)))
```

VC 'RECEIVER#5.0.2.2' subsumes VC 'RECEIVER#5.4.1.2':

```
   (IMPLIES
     (AND
       (CONSISTENT (QUOTE DOM)
                   (QUOTE EQUAL)
```

```
              (JOIN (FLATTEN PKT.IN)
                    PKT))
  (OR
    (NOT (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (JOIN (FLATTEN PKT.IN)
                                             PKT))
                               (ADD1 NEXT))))
    (LESSP (REACH (REDUCE (QUOTE WITH)
                          (NULL)
                          (JOIN (FLATTEN PKT.IN)
                                PKT)))
           NEXT)
    (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                      (JOIN (FLATTEN PKT.IN)
                            PKT))))
    (IF
      (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                          PKT)))
      (OR
        (NOT (LESSP 0 NEXT))
        (NOT
          (EQUAL (FLATTEN SINK)
                 (RANGE (LOWER (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (JOIN (FLATTEN PKT.IN)
                                             PKT))
                               (SUB1 NEXT))))))
      (OR (NOT (EQUAL NEXT 0))
          (NOT (EQUAL SINK (NULL))))))
  (PMAPP QUEUE)
  (NUMBERP NEXT)
  (NUMBERP EDGE)
  (LESSP (DOM (LST PKT))
         NEXT)
  (SEQP PKT)
  (PMAPP PKT)
  (ALL (FAPPLY (QUOTE CONTIG)
               (APR PKT.IN PKT))))
(ALL (FAPPLY (QUOTE CONTIG)
             PKT.IN)))
```

VC 'RECEIVER#5.4.2.1.1' is trivially true:

```
   (IMPLIES (AND (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                               (NULL)
                                               (FLATTEN PKT.IN))
                                       (ADD1 NEXT)))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT)))
                 (NOT (CONSISTENT (QUOTE DOM)
                                  (QUOTE EQUAL)
                                  (JOIN (FLATTEN PKT.IN)
                                        PKT))))
            (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                          (NULL)
                                          (JOIN (FLATTEN PKT.IN)
                                                PKT))
                                  (ADD1 NEXT))))
```

VC 'RECEIVER#5.4.2.1.2' is trivially true:

```
   (IMPLIES (AND (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                               (NULL)
                                               (FLATTEN PKT.IN))
                                       (ADD1 NEXT)))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT)))
                 (NOT (ALL (FAPPLY (QUOTE CONTIG)
                                   (APR PKT.IN PKT)))))
            (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                          (NULL)
                                          (JOIN (FLATTEN PKT.IN)
                                                PKT))
                                  (ADD1 NEXT))))
```

```
-------------------

Using equality hypothesis:
   (EQUAL (REDUCE (QUOTE WITH)
                  (NULL)
                  (JOIN (FLATTEN PKT.IN)
                        PKT))
          (REDUCE (QUOTE WITH)
                  (NULL)
                  (FLATTEN PKT.IN)))

VC 'RECEIVER#5.4.2.1.3' is trivially true:

   (IMPLIES (AND (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                               (NULL)
                                               (FLATTEN PKT.IN))
                                       (ADD1 NEXT)))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT))))
            (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                          (NULL)
                                          (FLATTEN PKT.IN))
                                  (ADD1 NEXT))))

-------------------

VC 'RECEIVER#5.4.2.2.1' is trivially true:

   (IMPLIES (AND (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN PKT.IN)))
                             NEXT))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT)))
                 (NOT (CONSISTENT (QUOTE DOM)
                                  (QUOTE EQUAL)
```

```
                                (JOIN (FLATTEN PKT.IN)
                                      PKT))))
           (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (JOIN (FLATTEN PKT.IN)
                                            PKT)))
                       NEXT)))
```

```
-------------------
```

VC 'RECEIVER#5.4.2.2.2' is trivially true:

```
   (IMPLIES (AND (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN PKT.IN)))
                             NEXT))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT)))
                 (NOT (ALL (FAPPLY (QUOTE CONTIG)
                                   (APR PKT.IN PKT)))))
            (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (JOIN (FLATTEN PKT.IN)
                                             PKT)))
                        NEXT)))
```

```
-------------------
```

Using equality hypothesis:

```
   (EQUAL (REDUCE (QUOTE WITH)
                  (NULL)
                  (JOIN (FLATTEN PKT.IN)
                        PKT))
          (REDUCE (QUOTE WITH)
                  (NULL)
                  (FLATTEN PKT.IN)))
```

VC 'RECEIVER#5.4.2.2.3' is trivially true:

```
   (IMPLIES (AND (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN PKT.IN)))
                             NEXT))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT)))))
            (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (FLATTEN PKT.IN)))
                        NEXT)))

--------------------

Generating new VC 'RECEIVER#5.4.2.3':

   (IMPLIES (AND (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.IN)))
                 (PMAPP PKT))
            (ALL (FAPPLY (QUOTE MPAIRP)
                         (JOIN (FLATTEN PKT.IN)
                               PKT))))


VC not proved, being written.

--------------------

VC 'RECEIVER#5.4.2.4.1.1' is trivially true:

   (IMPLIES (LESSP 0 NEXT)
            (LESSP 0 NEXT))

--------------------

VC 'RECEIVER#5.4.2.4.1.2.1' is trivially true:

   (IMPLIES (AND (EQUAL (FLATTEN SINK)
                        (RANGE (LOWER (REDUCE (QUOTE WITH)
                                              (NULL)
                                              (FLATTEN PKT.IN))
                                      (SUB1 NEXT))))
```

```
          (CONSISTENT (QUOTE DOM)
                      (QUOTE EQUAL)
                      (JOIN (FLATTEN PKT.IN)
                            PKT))
          (ALL (FAPPLY (QUOTE CONTIG)
                       (APR PKT.IN PKT)))
          (NOT (CONSISTENT (QUOTE DOM)
                           (QUOTE EQUAL)
                           (JOIN (FLATTEN PKT.IN)
                                 PKT))))
     (EQUAL (FLATTEN SINK)
            (RANGE (LOWER (REDUCE (QUOTE WITH)
                                  (NULL)
                                  (JOIN (FLATTEN PKT.IN)
                                        PKT))
                          (SUB1 NEXT)))))
```

VC 'RECEIVER#5.4.2.4.1.2.2' is trivially true:

```
   (IMPLIES (AND (EQUAL (FLATTEN SINK)
                        (RANGE (LOWER (REDUCE (QUOTE WITH)
                                              (NULL)
                                              (FLATTEN PKT.IN))
                                      (SUB1 NEXT))))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT)))
                 (NOT (ALL (FAPPLY (QUOTE CONTIG)
                                   (APR PKT.IN PKT)))))
            (EQUAL (FLATTEN SINK)
                   (RANGE (LOWER (REDUCE (QUOTE WITH)
                                         (NULL)
                                         (JOIN (FLATTEN PKT.IN)
                                               PKT))
                                 (SUB1 NEXT)))))
```

Using equality hypothesis:

```
   (EQUAL (REDUCE (QUOTE WITH)
                  (NULL)
                  (JOIN (FLATTEN PKT.IN)
                        PKT))
          (REDUCE (QUOTE WITH)
                  (NULL)
                  (FLATTEN PKT.IN)))
```

VC 'RECEIVER#5.4.2.4.1.2.3' is trivially true:

```
   (IMPLIES (AND (EQUAL (FLATTEN SINK)
                        (RANGE (LOWER (REDUCE (QUOTE WITH)
                                              (NULL)
                                              (FLATTEN PKT.IN))
                                      (SUB1 NEXT))))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT))))
            (EQUAL (FLATTEN SINK)
                   (RANGE (LOWER (REDUCE (QUOTE WITH)
                                         (NULL)
                                         (FLATTEN PKT.IN))
                                 (SUB1 NEXT)))))
```

VC 'RECEIVER#5.0.2.4' subsumes VC 'RECEIVER#5.4.2.4.2':

```
   (IMPLIES
     (AND
       (OR
         (NOT (LESSP 0 NEXT))
         (NOT (EQUAL (FLATTEN SINK)
                     (RANGE (LOWER (REDUCE (QUOTE WITH)
                                           (NULL)
                                           (JOIN (FLATTEN PKT.IN)
                                                 PKT))
                                   (SUB1 NEXT))))))
       (EQUAL NEXT 0)
       (EQUAL SINK (NULL))
       (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                                  (NULL)
                                                  (FLATTEN PKT.IN))
                                          (ADD1 NEXT)))
        (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                   (NULL)
                                   (FLATTEN PKT.IN)))
                    NEXT))
        (ALL (FAPPLY (QUOTE MPAIRP)
                     (FLATTEN PKT.IN)))
        (CONSISTENT (QUOTE DOM)
                    (QUOTE EQUAL)
                    (JOIN (FLATTEN PKT.IN)
                          PKT))
        (ALL (FAPPLY (QUOTE CONTIG)
                     (APR PKT.IN PKT)))
        (PMAPP QUEUE)
        (NUMBERP NEXT)
        (NUMBERP EDGE)
        (LESSP (DOM (LST PKT))
               NEXT)
        (SEQP PKT)
        (PMAPP PKT)
        (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                            PKT))))
      (IN 0 (DOMAIN (FLATTEN PKT.IN))))

-------------------

Generating new VC 'RECEIVER#5.4.2.4.3':

   (IMPLIES (IN 0 (DOMAIN (FLATTEN PKT.IN)))
            (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                                PKT))))


VC not proved, being written.

-------------------

VC 'RECEIVER#5.4.2.4.4.1' is trivially true:

   (IMPLIES (EQUAL NEXT 0)
            (EQUAL NEXT 0))
```

```
-------------------

VC 'RECEIVER#5.4.2.4.4.2' is trivially true:

   (IMPLIES (EQUAL SINK (NULL))
            (EQUAL SINK (NULL)))


----------------------------------------


Path conditions for VC 'RECEIVER#6':

   (AND (LESSP NEXT (DOM (FIRST PKT)))
        (SEQP PKT)
        (PMAPP PKT))

Variable substitution:

   CREDITS      :=  CREDITS
   PKT.IN       :=  (APR PKT.IN PKT)
   SINK         :=  SINK
   ACK.OUT      :=  ACK.OUT
   NEXT         :=  NEXT
   EDGE         :=  EDGE
   QUEUE        :=  (REDUCE (QUOTE WITH)
                            QUEUE
                            (LOWER PKT EDGE))

Instantiated assertion:

   (AND
     (PMAPP (REDUCE (QUOTE WITH)
                    QUEUE
                    (LOWER PKT EDGE)))
     (NUMBERP NEXT)
     (NUMBERP EDGE)
     (IMPLIES
       (AND (CONSISTENT (QUOTE DOM)
                        (QUOTE EQUAL)
                        (FLATTEN (APR PKT.IN PKT)))
            (ALL (FAPPLY (QUOTE CONTIG)
                         (APR PKT.IN PKT))))
       (AND
         (FOLLOWS (REDUCE (QUOTE WITH)
                          QUEUE
                          (LOWER PKT EDGE))
                  (UPPER (REDUCE (QUOTE WITH)
                                 (NULL)
                                 (FLATTEN (APR PKT.IN PKT)))
                         (ADD1 NEXT)))
         (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                    (NULL)
                                    (FLATTEN (APR PKT.IN PKT))))
                     NEXT))
         (ALL (FAPPLY (QUOTE MPAIRP)
                      (FLATTEN (APR PKT.IN PKT))))
         (IF
           (IN 0 (DOMAIN (FLATTEN (APR PKT.IN PKT))))
           (AND (LESSP 0 NEXT)
                (EQUAL (FLATTEN SINK)
                       (RANGE (LOWER (REDUCE (QUOTE WITH)
                                             (NULL)
                                             (FLATTEN (APR PKT.IN
                                                           PKT)))
                                     (SUB1 NEXT)))))
           (AND (EQUAL NEXT 0)
                (EQUAL SINK (NULL)))))))
```

VC 'RECEIVER#5.0.1' subsumes VC 'RECEIVER#6.0.1':

```
   (EQUAL (FLATTEN (APR PKT.IN PKT))
          (JOIN (FLATTEN PKT.IN)
                PKT))
```

Equality substitution performed.

Generating new VC 'RECEIVER#6.1':

```
   (IMPLIES (PMAPP QUEUE)
            (PMAPP (REDUCE (QUOTE WITH)
                           QUEUE
                           (LOWER PKT EDGE))))
```

```
VC not proved, being written.

-------------------

VC 'RECEIVER#6.2' is trivially true:

   (IMPLIES (NUMBERP NEXT)
            (NUMBERP NEXT))

-------------------

VC 'RECEIVER#6.3' is trivially true:

   (IMPLIES (NUMBERP EDGE)
            (NUMBERP EDGE))

-------------------

VC 'RECEIVER#5.0.2.1' subsumes VC 'RECEIVER#6.4.1.1':

   (IMPLIES
     (AND
       (ALL (FAPPLY (QUOTE CONTIG)
                    (APR PKT.IN PKT)))
       (OR
         (NOT (FOLLOWS (REDUCE (QUOTE WITH)
                               QUEUE
                               (LOWER PKT EDGE))
                       (UPPER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (JOIN (FLATTEN PKT.IN)
                                            PKT))
                              (ADD1 NEXT)))))
         (LESSP (REACH (REDUCE (QUOTE WITH)
                               (NULL)
                               (JOIN (FLATTEN PKT.IN)
                                     PKT)))
                NEXT)
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (JOIN (FLATTEN PKT.IN)
                                 PKT))))
         (IF
           (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                                   PKT)))
          (OR
            (NOT (LESSP 0 NEXT))
            (NOT
              (EQUAL (FLATTEN SINK)
                     (RANGE (LOWER (REDUCE (QUOTE WITH)
                                           (NULL)
                                           (JOIN (FLATTEN PKT.IN)
                                                 PKT))
                                   (SUB1 NEXT))))))
          (OR (NOT (EQUAL NEXT 0))
              (NOT (EQUAL SINK (NULL))))))
      (PMAPP QUEUE)
      (NUMBERP NEXT)
      (NUMBERP EDGE)
      (LESSP NEXT (DOM (FIRST PKT)))
      (SEQP PKT)
      (PMAPP PKT)
      (CONSISTENT (QUOTE DOM)
                  (QUOTE EQUAL)
                  (JOIN (FLATTEN PKT.IN)
                        PKT)))
    (CONSISTENT (QUOTE DOM)
                (QUOTE EQUAL)
                (FLATTEN PKT.IN)))
```

VC 'RECEIVER#5.0.2.2' subsumes VC 'RECEIVER#6.4.1.2':

```
  (IMPLIES
    (AND
      (CONSISTENT (QUOTE DOM)
                  (QUOTE EQUAL)
                  (JOIN (FLATTEN PKT.IN)
                        PKT))
      (OR
        (NOT (FOLLOWS (REDUCE (QUOTE WITH)
                              QUEUE
                              (LOWER PKT EDGE))
                      (UPPER (REDUCE (QUOTE WITH)
                                     (NULL)
                                     (JOIN (FLATTEN PKT.IN)
                                           PKT))
```

```
                               (ADD1 NEXT))))
        (LESSP (REACH (REDUCE (QUOTE WITH)
                               (NULL)
                               (JOIN (FLATTEN PKT.IN)
                                     PKT)))
               NEXT)
        (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                          (JOIN (FLATTEN PKT.IN)
                                PKT))))
        (IF
          (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                               PKT)))
          (OR
            (NOT (LESSP 0 NEXT))
            (NOT
              (EQUAL (FLATTEN SINK)
                     (RANGE (LOWER (REDUCE (QUOTE WITH)
                                           (NULL)
                                           (JOIN (FLATTEN PKT.IN)
                                                 PKT))
                                   (SUB1 NEXT))))))
          (OR (NOT (EQUAL NEXT 0))
              (NOT (EQUAL SINK (NULL))))))
       (PMAPP QUEUE)
       (NUMBERP NEXT)
       (NUMBERP EDGE)
       (LESSP NEXT (DOM (FIRST PKT)))
       (SEQP PKT)
       (PMAPP PKT)
       (ALL (FAPPLY (QUOTE CONTIG)
                    (APR PKT.IN PKT))))
     (ALL (FAPPLY (QUOTE CONTIG)
                  PKT.IN)))

--------------------

Generating new VC 'RECEIVER#6.4.2.1':

   (IMPLIES (AND (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                               (NULL)
                                               (FLATTEN PKT.IN))
                                       (ADD1 NEXT)))
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.IN)))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (PMAPP QUEUE)
                 (NUMBERP NEXT)
                 (LESSP NEXT (DOM (FIRST PKT)))
                 (PMAPP PKT))
            (FOLLOWS (REDUCE (QUOTE WITH)
                             QUEUE
                             (LOWER PKT EDGE))
                     (UPPER (REDUCE (QUOTE WITH)
                                    (NULL)
                                    (JOIN (FLATTEN PKT.IN)
                                          PKT))
                            (ADD1 NEXT))))
```

VC not proved, being written.

To be proved with lemmas (FOLLOWS.REDUCE.WITH.UPPER)

```
-------------------
```

Generating new VC 'RECEIVER#6.4.2.2':

```
   (IMPLIES (AND (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN PKT.IN)))
                             NEXT))
                 (NUMBERP NEXT)
                 (LESSP NEXT (DOM (FIRST PKT)))
                 (PMAPP PKT))
            (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (JOIN (FLATTEN PKT.IN)
                                             PKT)))
                        NEXT)))
```

VC not proved, being written.

```
-------------------
```

```
VC 'RECEIVER#5.4.2.3' subsumes VC 'RECEIVER#6.4.2.3':

   (IMPLIES
     (AND
       (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                     (NULL)
                                     (FLATTEN PKT.IN))
                             (ADD1 NEXT)))
       (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                  (NULL)
                                  (FLATTEN PKT.IN)))
                   NEXT))
       (IF (IN 0 (DOMAIN (FLATTEN PKT.IN)))
           (AND (LESSP 0 NEXT)
                (EQUAL (FLATTEN SINK)
                       (RANGE (LOWER (REDUCE (QUOTE WITH)
                                             (NULL)
                                             (FLATTEN PKT.IN))
                                     (SUB1 NEXT)))))
           (AND (EQUAL NEXT 0)
                (EQUAL SINK (NULL))))
       (CONSISTENT (QUOTE DOM)
                   (QUOTE EQUAL)
                   (JOIN (FLATTEN PKT.IN)
                         PKT))
       (ALL (FAPPLY (QUOTE CONTIG)
                    (APR PKT.IN PKT)))
       (PMAPP QUEUE)
       (NUMBERP NEXT)
       (NUMBERP EDGE)
       (LESSP NEXT (DOM (FIRST PKT)))
       (SEQP PKT)
       (PMAPP PKT)
       (ALL (FAPPLY (QUOTE MPAIRP)
                    (FLATTEN PKT.IN))))
     (ALL (FAPPLY (QUOTE MPAIRP)
                  (JOIN (FLATTEN PKT.IN)
                        PKT))))

--------------------

VC 'RECEIVER#6.4.2.4.1.1' is trivially true:
   (IMPLIES (LESSP 0 NEXT)
            (LESSP 0 NEXT))

-------------------

Using equality hypothesis:
   (EQUAL (FLATTEN SINK)
          (RANGE (LOWER (REDUCE (QUOTE WITH)
                                (NULL)
                                (FLATTEN PKT.IN))
                        (SUB1 NEXT))))

Generating new VC 'RECEIVER#6.4.2.4.1.2':

   (IMPLIES (AND (LESSP 0 NEXT)
                 (NUMBERP NEXT)
                 (LESSP NEXT (DOM (FIRST PKT)))
                 (PMAPP PKT))
            (EQUAL (RANGE (LOWER (REDUCE (QUOTE WITH)
                                         (NULL)
                                         (FLATTEN PKT.IN))
                                 (SUB1 NEXT)))
                   (RANGE (LOWER (REDUCE (QUOTE WITH)
                                         (NULL)
                                         (JOIN (FLATTEN PKT.IN)
                                               PKT))
                                 (SUB1 NEXT)))))


VC not proved, being written.

-------------------

Generating new VC 'RECEIVER#6.4.2.4.2':

   (IMPLIES (AND (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                                     PKT)))
                 (EQUAL NEXT 0)
                 (LESSP NEXT (DOM (FIRST PKT)))
                 (PMAPP PKT))
            (IN 0 (DOMAIN (FLATTEN PKT.IN))))


VC not proved, being written.
```

VC 'RECEIVER#5.4.2.4.3' subsumes VC 'RECEIVER#6.4.2.4.3':

```
   (IMPLIES (AND (OR (NOT (EQUAL NEXT 0))
                     (NOT (EQUAL SINK (NULL))))
                 (LESSP 0 NEXT)
                 (EQUAL (FLATTEN SINK)
                        (RANGE (LOWER (REDUCE (QUOTE WITH)
                                              (NULL)
                                              (FLATTEN PKT.IN))
                                      (SUB1 NEXT))))
                 (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                               (NULL)
                                               (FLATTEN PKT.IN))
                                       (ADD1 NEXT)))
                 (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN PKT.IN)))
                             NEXT))
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.IN)))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT)))
                 (PMAPP QUEUE)
                 (NUMBERP NEXT)
                 (NUMBERP EDGE)
                 (LESSP NEXT (DOM (FIRST PKT)))
                 (SEQP PKT)
                 (PMAPP PKT)
                 (IN 0 (DOMAIN (FLATTEN PKT.IN))))
            (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                                PKT))))
```

VC 'RECEIVER#6.4.2.4.4.1' is trivially true:

```
   (IMPLIES (EQUAL NEXT 0)
          (EQUAL NEXT 0))
-------------------

VC 'RECEIVER#6.4.2.4.4.2' is trivially true:

   (IMPLIES (EQUAL SINK (NULL))
            (EQUAL SINK (NULL)))


----------------------------------------


Path conditions for VC 'RECEIVER#7':

   (AND (EQUAL M (REDUCE (QUOTE WITH)
                         QUEUE
                         (UPPER (LOWER PKT EDGE)
                                NEXT)))
        (NOT (LESSP (DOM (LST PKT))
                    NEXT))
        (NOT (LESSP NEXT (DOM (FIRST PKT))))
        (SEQP PKT)
        (NOT (LESSP EDGE NEXT))
        (PMAPP PKT))

Variable substitution:

   CREDITS      :=  CREDITS
   PKT.IN       :=  (APR PKT.IN PKT)
   SINK         :=  (APR SINK (RANGE (CONSEC M)))
   ACK.OUT      :=  (APR ACK.OUT (ACKNOWL (REACH M)
                                          EDGE))
   NEXT         :=  (REACH M)
   EDGE         :=  EDGE
   QUEUE        :=  (UPPER QUEUE (ADD1 (REACH M)))

Instantiated assertion:

   (AND
     (PMAPP (UPPER QUEUE (ADD1 (REACH M))))
     (NUMBERP (REACH M))
     (NUMBERP EDGE)
     (IMPLIES
      (AND (CONSISTENT (QUOTE DOM)
                       (QUOTE EQUAL)
                       (FLATTEN (APR PKT.IN PKT)))
           (ALL (FAPPLY (QUOTE CONTIG)
                        (APR PKT.IN PKT))))
      (AND
        (FOLLOWS (UPPER QUEUE (ADD1 (REACH M)))
                 (UPPER (REDUCE (QUOTE WITH)
                                (NULL)
                                (FLATTEN (APR PKT.IN PKT)))
                        (ADD1 (REACH M))))
        (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                   (NULL)
                                   (FLATTEN (APR PKT.IN PKT))))
                    (REACH M)))
        (ALL (FAPPLY (QUOTE MPAIRP)
                     (FLATTEN (APR PKT.IN PKT))))
        (IF
          (IN 0 (DOMAIN (FLATTEN (APR PKT.IN PKT))))
          (AND (LESSP 0 (REACH M))
               (EQUAL (FLATTEN (APR SINK (RANGE (CONSEC M))))
                      (RANGE (LOWER (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN (APR PKT.IN
                                                          PKT)))
                                    (SUB1 (REACH M))))))
          (AND (EQUAL (REACH M)
                      0)
               (EQUAL (APR SINK (RANGE (CONSEC M)))
                      (NULL)))))))

-------------------

VC 'RECEIVER#5.0.1' subsumes VC 'RECEIVER#7.0.1':

   (EQUAL (FLATTEN (APR PKT.IN PKT))
          (JOIN (FLATTEN PKT.IN)
                PKT))

Equality substitution performed.

-------------------

Generating new VC 'RECEIVER#7.1':
   (IMPLIES (PMAPP QUEUE)
            (PMAPP (UPPER QUEUE (ADD1 (REACH M)))))


VC not proved, being written.

-------------------

Generating new VC 'RECEIVER#7.2':

   (NUMBERP (REACH M))

This conjecture simplifies, clearly, to:

      T.

Q.E.D.

VC has been proved.

Compute time:    2.775 sec,   GC time:    0.000 sec.

-------------------

VC 'RECEIVER#7.3' is trivially true:

   (IMPLIES (NUMBERP EDGE)
            (NUMBERP EDGE))

-------------------

VC 'RECEIVER#5.0.2.1' subsumes VC 'RECEIVER#7.4.1.1':

   (IMPLIES
     (AND
       (ALL (FAPPLY (QUOTE CONTIG)
                    (APR PKT.IN PKT)))
       (OR
         (NOT (FOLLOWS (UPPER QUEUE (ADD1 (REACH M)))
                       (UPPER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (JOIN (FLATTEN PKT.IN)
                                            PKT))
```

```
                              (ADD1 (REACH M)))))
    (LESSP (REACH (REDUCE (QUOTE WITH)
                          (NULL)
                          (JOIN (FLATTEN PKT.IN)
                                PKT)))
           (REACH M))
    (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                      (JOIN (FLATTEN PKT.IN)
                            PKT))))
    (IF
      (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                          PKT)))
      (OR
        (NOT (LESSP 0 (REACH M)))
        (NOT
          (EQUAL (FLATTEN (APR SINK (RANGE (CONSEC M))))
                 (RANGE (LOWER (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (JOIN (FLATTEN PKT.IN)
                                             PKT))
                               (SUB1 (REACH M)))))))
      (OR (NOT (EQUAL (REACH M)
                      0))
          (NOT (EQUAL (APR SINK (RANGE (CONSEC M)))
                      (NULL))))))
  (PMAPP QUEUE)
  (NUMBERP NEXT)
  (NUMBERP EDGE)
  (EQUAL M (REDUCE (QUOTE WITH)
                   QUEUE
                   (UPPER (LOWER PKT EDGE)
                          NEXT)))
  (NOT (LESSP (DOM (LST PKT))
              NEXT))
  (NOT (LESSP NEXT (DOM (FIRST PKT))))
  (SEQP PKT)
  (NOT (LESSP EDGE NEXT))
  (PMAPP PKT)
  (CONSISTENT (QUOTE DOM)
              (QUOTE EQUAL)
              (JOIN (FLATTEN PKT.IN)
                    PKT)))
(CONSISTENT (QUOTE DOM)
            (QUOTE EQUAL)
```

```
                 (FLATTEN PKT.IN)))

--------------------

VC 'RECEIVER#5.0.2.2' subsumes VC 'RECEIVER#7.4.1.2':

   (IMPLIES
     (AND
       (CONSISTENT (QUOTE DOM)
                   (QUOTE EQUAL)
                   (JOIN (FLATTEN PKT.IN)
                         PKT))
       (OR
         (NOT (FOLLOWS (UPPER QUEUE (ADD1 (REACH M)))
                       (UPPER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (JOIN (FLATTEN PKT.IN)
                                            PKT))
                              (ADD1 (REACH M))))))
         (LESSP (REACH (REDUCE (QUOTE WITH)
                               (NULL)
                               (JOIN (FLATTEN PKT.IN)
                                     PKT)))
                (REACH M))
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (JOIN (FLATTEN PKT.IN)
                                 PKT))))
         (IF
           (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                               PKT)))
           (OR
             (NOT (LESSP 0 (REACH M)))
             (NOT
               (EQUAL (FLATTEN (APR SINK (RANGE (CONSEC M))))
                      (RANGE (LOWER (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (JOIN (FLATTEN PKT.IN)
                                                  PKT))
                                    (SUB1 (REACH M))))))))
           (OR (NOT (EQUAL (REACH M)
                           0))
               (NOT (EQUAL (APR SINK (RANGE (CONSEC M)))
                           (NULL))))))
       (PMAPP QUEUE)
```

```
        (NUMBERP NEXT)
        (NUMBERP EDGE)
        (EQUAL M (REDUCE (QUOTE WITH)
                         QUEUE
                         (UPPER (LOWER PKT EDGE)
                                NEXT)))
        (NOT (LESSP (DOM (LST PKT))
                    NEXT))
        (NOT (LESSP NEXT (DOM (FIRST PKT))))
        (SEQP PKT)
        (NOT (LESSP EDGE NEXT))
        (PMAPP PKT)
        (ALL (FAPPLY (QUOTE CONTIG)
                     (APR PKT.IN PKT))))
      (ALL (FAPPLY (QUOTE CONTIG)
                   PKT.IN)))
```

Generating new VC 'RECEIVER#7.4.2.1':

```
   (IMPLIES (AND (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                               (NULL)
                                               (FLATTEN PKT.IN))
                                       (ADD1 NEXT)))
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.IN)))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (PMAPP QUEUE)
                 (PMAPP PKT))
            (FOLLOWS (UPPER QUEUE (ADD1 (REACH M)))
                     (UPPER (REDUCE (QUOTE WITH)
                                    (NULL)
                                    (JOIN (FLATTEN PKT.IN)
                                          PKT))
                            (ADD1 (REACH M)))))
```

VC not proved, being written.

To be proved with lemmas (FOLLOWS.REDUCE.WITH.3)

```
-------------------

Generating new VC 'RECEIVER#7.4.2.2.1':

   (IMPLIES (AND (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN PKT.IN)))
                             NEXT))
                 (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                               (NULL)
                                               (FLATTEN PKT.IN))
                                       (ADD1 NEXT)))
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.IN)))
                 (IN 0 (DOMAIN (FLATTEN PKT.IN)))
                 (LESSP 0 NEXT)
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT)))
                 (PMAPP QUEUE)
                 (NUMBERP NEXT)
                 (NUMBERP EDGE)
                 (EQUAL M (REDUCE (QUOTE WITH)
                                  QUEUE
                                  (UPPER (LOWER PKT EDGE)
                                         NEXT)))
                 (NOT (LESSP (DOM (LST PKT))
                             NEXT))
                 (NOT (LESSP NEXT (DOM (FIRST PKT))))
                 (SEQP PKT)
                 (NOT (LESSP EDGE NEXT))
                 (PMAPP PKT))
            (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (JOIN (FLATTEN PKT.IN)
                                             PKT)))
                        (REACH M))))
```

```
VC not proved, being written.

To be proved with lemmas (DOM.FIRST.REDUCE.UPPER.LOWER
                          FOLLOWS.REDUCE.CONSISTENT.JOIN)


-------------------

Generating new VC 'RECEIVER#7.4.2.2.2':

   (IMPLIES (AND (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                               (NULL)
                                               (FLATTEN PKT.IN))
                                       (ADD1 NEXT)))
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.IN)))
                 (NOT (IN 0 (DOMAIN (FLATTEN PKT.IN))))
                 (EQUAL NEXT 0)
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT)))
                 (PMAPP QUEUE)
                 (NUMBERP EDGE)
                 (EQUAL M (REDUCE (QUOTE WITH)
                                  QUEUE
                                  (UPPER (LOWER PKT EDGE)
                                         NEXT)))
                 (NOT (LESSP NEXT (DOM (FIRST PKT))))
                 (SEQP PKT)
                 (PMAPP PKT))
            (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (JOIN (FLATTEN PKT.IN)
                                             PKT)))
                        (REACH M))))


VC not proved, being written.

To be proved with lemmas (FOLLOWS.REDUCE.WITH.LOWER.REDUCE
                          DOM.FIRST.REDUCE.WITH.ZERO)
```

VC 'RECEIVER#5.4.2.3' subsumes VC 'RECEIVER#7.4.2.3':

```
(IMPLIES
  (AND
    (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                  (NULL)
                                  (FLATTEN PKT.IN))
                          (ADD1 NEXT)))
    (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                               (NULL)
                               (FLATTEN PKT.IN)))
                NEXT))
    (IF (IN 0 (DOMAIN (FLATTEN PKT.IN)))
        (AND (LESSP 0 NEXT)
             (EQUAL (FLATTEN SINK)
                    (RANGE (LOWER (REDUCE (QUOTE WITH)
                                          (NULL)
                                          (FLATTEN PKT.IN))
                                  (SUB1 NEXT))))))
        (AND (EQUAL NEXT 0)
             (EQUAL SINK (NULL))))
    (CONSISTENT (QUOTE DOM)
                (QUOTE EQUAL)
                (JOIN (FLATTEN PKT.IN)
                      PKT))
    (ALL (FAPPLY (QUOTE CONTIG)
                 (APR PKT.IN PKT)))
    (PMAPP QUEUE)
    (NUMBERP NEXT)
    (NUMBERP EDGE)
    (EQUAL M (REDUCE (QUOTE WITH)
                     QUEUE
                     (UPPER (LOWER PKT EDGE)
                            NEXT)))
    (NOT (LESSP (DOM (LST PKT))
                NEXT))
    (NOT (LESSP NEXT (DOM (FIRST PKT))))
    (SEQP PKT)
    (NOT (LESSP EDGE NEXT))
    (PMAPP PKT)
```

```
        (ALL (FAPPLY (QUOTE MPAIRP)
                     (FLATTEN PKT.IN))))
      (ALL (FAPPLY (QUOTE MPAIRP)
                   (JOIN (FLATTEN PKT.IN)
                         PKT))))
```

```
********************
```

Generating new VC 'RECEIVER#7.4.2.4.1.1':

```
   (IMPLIES (AND (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT)))
                 (PMAPP QUEUE)
                 (NUMBERP NEXT)
                 (NUMBERP EDGE)
                 (EQUAL M (REDUCE (QUOTE WITH)
                                  QUEUE
                                  (UPPER (LOWER PKT EDGE)
                                         NEXT)))
                 (NOT (LESSP (DOM (LST PKT))
                             NEXT))
                 (NOT (LESSP NEXT (DOM (FIRST PKT))))
                 (SEQP PKT)
                 (NOT (LESSP EDGE NEXT))
                 (PMAPP PKT))
            (LESSP 0 (REACH M)))
```

VC not proved, being written.

To be proved with lemmas (SEQP.UPPER.LOWER)

```
********************
```

Generating new VC 'RECEIVER#7.4.2.4.1.2':

```
   (IMPLIES (AND (EQUAL (FLATTEN SINK)
                        (RANGE (LOWER (REDUCE (QUOTE WITH)
                                              (NULL)
                                              (FLATTEN PKT.IN))
                                      (SUB1 NEXT))))
                 (LESSP 0 NEXT)
                 (IN 0 (DOMAIN (FLATTEN PKT.IN)))
```

```
        (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (FLATTEN PKT.IN))
                              (ADD1 NEXT)))
        (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                   (NULL)
                                   (FLATTEN PKT.IN)))
                    NEXT))
        (ALL (FAPPLY (QUOTE MPAIRP)
                     (FLATTEN PKT.IN)))
        (CONSISTENT (QUOTE DOM)
                    (QUOTE EQUAL)
                    (JOIN (FLATTEN PKT.IN)
                          PKT))
        (ALL (FAPPLY (QUOTE CONTIG)
                     (APR PKT.IN PKT)))
        (PMAPP QUEUE)
        (NUMBERP NEXT)
        (NUMBERP EDGE)
        (EQUAL M (REDUCE (QUOTE WITH)
                         QUEUE
                         (UPPER (LOWER PKT EDGE)
                                NEXT)))
        (NOT (LESSP (DOM (LST PKT))
                    NEXT))
        (NOT (LESSP NEXT (DOM (FIRST PKT))))
        (SEQP PKT)
        (NOT (LESSP EDGE NEXT))
        (PMAPP PKT))
   (EQUAL (FLATTEN (APR SINK (RANGE (CONSEC M))))
          (RANGE (LOWER (REDUCE (QUOTE WITH)
                                (NULL)
                                (JOIN (FLATTEN PKT.IN)
                                      PKT))
                        (SUB1 (REACH M))))))
```

VC not proved, being written.

```
To be proved with lemmas (FOLLOWS.REDUCE.WITH.UPPER.REDUCE
                          FOLLOWS.REDUCE.CONSISTENT.JOIN
                            DOM.FIRST.REDUCE.UPPER.LOWER
                                 LOWER.SUB1.REACH.4
                            LOWER.REDUCE.WITH.CONSISTENT)
```

```
-------------------

VC 'RECEIVER#7.4.2.4.1.1' subsumes VC 'RECEIVER#7.4.2.4.2.1':

   (IMPLIES (AND (NOT (LESSP 0 (REACH M)))
                 (EQUAL NEXT 0)
                 (EQUAL SINK (NULL))
                 (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                               (NULL)
                                               (FLATTEN PKT.IN))
                                       (ADD1 NEXT)))
                 (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN PKT.IN)))
                             NEXT))
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.IN)))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT)))
                 (PMAPP QUEUE)
                 (NUMBERP NEXT)
                 (NUMBERP EDGE)
                 (EQUAL M (REDUCE (QUOTE WITH)
                                  QUEUE
                                  (UPPER (LOWER PKT EDGE)
                                         NEXT)))
                 (NOT (LESSP (DOM (LST PKT))
                             NEXT))
                 (NOT (LESSP NEXT (DOM (FIRST PKT))))
                 (SEQP PKT)
                 (NOT (LESSP EDGE NEXT))
                 (PMAPP PKT))
            (IN 0 (DOMAIN (FLATTEN PKT.IN))))

-------------------

Generating new VC 'RECEIVER#7.4.2.4.2.2':
```

```
(IMPLIES
  (AND (NOT (EQUAL (FLATTEN (APR SINK (RANGE (CONSEC M))))
                   (RANGE (LOWER (REDUCE (QUOTE WITH)
                                         (NULL)
                                         (JOIN (FLATTEN PKT.IN)
                                               PKT))
                                 (SUB1 (REACH M))))))
       (EQUAL NEXT 0)
       (EQUAL SINK (NULL))
       (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                     (NULL)
                                     (FLATTEN PKT.IN))
                             (ADD1 NEXT)))
       (ALL (FAPPLY (QUOTE MPAIRP)
                    (FLATTEN PKT.IN)))
       (CONSISTENT (QUOTE DOM)
                   (QUOTE EQUAL)
                   (JOIN (FLATTEN PKT.IN)
                         PKT))
       (ALL (FAPPLY (QUOTE CONTIG)
                    (APR PKT.IN PKT)))
       (PMAPP QUEUE)
       (NUMBERP EDGE)
       (EQUAL M (REDUCE (QUOTE WITH)
                        QUEUE
                        (UPPER (LOWER PKT EDGE)
                               NEXT)))
       (NOT (LESSP NEXT (DOM (FIRST PKT))))
       (SEQP PKT)
       (PMAPP PKT))
  F)
```

VC not proved, being written.

```
To be proved with lemmas (LOWER.SUB1.REACH.ZERO
                                     DOM.FIRST.REDUCE.WITH.ZERO
                                FOLLOWS.REDUCE.WITH.LOWER.REDUCE)
```

VC 'RECEIVER#5.4.2.4.3' subsumes VC 'RECEIVER#7.4.2.4.3':

```
(IMPLIES (AND (OR (NOT (EQUAL (REACH M)
                              0))
                  (NOT (EQUAL (APR SINK (RANGE (CONSEC M)))
                              (NULL))))
              (LESSP 0 NEXT)
              (EQUAL (FLATTEN SINK)
                     (RANGE (LOWER (REDUCE (QUOTE WITH)
                                           (NULL)
                                           (FLATTEN PKT.IN))
                                   (SUB1 NEXT))))
              (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN PKT.IN))
                                    (ADD1 NEXT)))
              (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                         (NULL)
                                         (FLATTEN PKT.IN)))
                          NEXT))
              (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.IN)))
              (CONSISTENT (QUOTE DOM)
                          (QUOTE EQUAL)
                          (JOIN (FLATTEN PKT.IN)
                                PKT))
              (ALL (FAPPLY (QUOTE CONTIG)
                           (APR PKT.IN PKT)))
              (PMAPP QUEUE)
              (NUMBERP NEXT)
              (NUMBERP EDGE)
              (EQUAL M (REDUCE (QUOTE WITH)
                               QUEUE
                               (UPPER (LOWER PKT EDGE)
                                      NEXT)))
              (NOT (LESSP (DOM (LST PKT))
                          NEXT))
              (NOT (LESSP NEXT (DOM (FIRST PKT))))
              (SEQP PKT)
              (NOT (LESSP EDGE NEXT))
              (PMAPP PKT)
              (IN 0 (DOMAIN (FLATTEN PKT.IN))))
         (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                             PKT))))
```

```
Generating new VC 'RECEIVER#7.4.2.4.4.1':

   (IMPLIES (AND (EQUAL NEXT 0)
                 (NOT (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                                          PKT))))
                 (NOT (LESSP NEXT (DOM (FIRST PKT))))
                 (SEQP PKT)
                 (PMAPP PKT))
            F)

VC not proved, being written.

To be proved with lemmas (IN.ZERO.DOMAIN)


--------------------

VC 'RECEIVER#7.4.2.4.4.1' subsumes VC 'RECEIVER#7.4.2.4.4.2':

   (IMPLIES (AND (EQUAL NEXT 0)
                 (NOT (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                                          PKT))))
                 (NOT (IN 0 (DOMAIN (FLATTEN PKT.IN))))
                 (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                               (NULL)
                                               (FLATTEN PKT.IN))
                                       (ADD1 NEXT)))
                 (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN PKT.IN)))
                             NEXT))
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.IN)))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT)))
                 (PMAPP QUEUE)
                 (NUMBERP NEXT)
                 (NUMBERP EDGE)
```

```
                (EQUAL M (REDUCE (QUOTE WITH)
                                 QUEUE
                                 (UPPER (LOWER PKT EDGE)
                                        NEXT)))
                (NOT (LESSP (DOM (LST PKT))
                            NEXT))
                (NOT (LESSP NEXT (DOM (FIRST PKT))))
                (SEQP PKT)
                (NOT (LESSP EDGE NEXT))
                (PMAPP PKT)
                (EQUAL SINK (NULL)))
           (EQUAL (APR SINK (RANGE (CONSEC M)))
                  (NULL)))
```

Path conditions for VC 'RECEIVER#8':

```
   (AND (OR (EQUAL PKT (NULL))
            (AND (NOT (LESSP (DOM (LST PKT))
                             NEXT))
                 (NOT (LESSP NEXT (DOM (FIRST PKT))))
                 (SEQP PKT)
                 (LESSP EDGE NEXT)))
        (PMAPP PKT))
```

Variable substitution:

```
   CREDITS      :=  CREDITS
   PKT.IN       :=  (APR PKT.IN PKT)
   SINK         :=  SINK
   ACK.OUT      :=  ACK.OUT
   NEXT         :=  NEXT
   EDGE         :=  EDGE
   QUEUE        :=  QUEUE
```

Instantiated assertion:

```
   (AND
     (PMAPP QUEUE)
     (NUMBERP NEXT)
     (NUMBERP EDGE)
```

```
    (IMPLIES
      (AND (CONSISTENT (QUOTE DOM)
                       (QUOTE EQUAL)
                       (FLATTEN (APR PKT.IN PKT)))
           (ALL (FAPPLY (QUOTE CONTIG)
                        (APR PKT.IN PKT))))
      (AND
        (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (FLATTEN (APR PKT.IN PKT)))
                              (ADD1 NEXT)))
        (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                   (NULL)
                                   (FLATTEN (APR PKT.IN PKT))))
                    NEXT))
        (ALL (FAPPLY (QUOTE MPAIRP)
                     (FLATTEN (APR PKT.IN PKT))))
        (IF
          (IN 0 (DOMAIN (FLATTEN (APR PKT.IN PKT))))
          (AND (LESSP 0 NEXT)
               (EQUAL (FLATTEN SINK)
                      (RANGE (LOWER (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN (APR PKT.IN
                                                          PKT)))
                                    (SUB1 NEXT)))))
          (AND (EQUAL NEXT 0)
               (EQUAL SINK (NULL)))))))
```

VC 'RECEIVER#5.0.1' subsumes VC 'RECEIVER#8.0.1':

```
   (EQUAL (FLATTEN (APR PKT.IN PKT))
          (JOIN (FLATTEN PKT.IN)
                PKT))
```

Equality substitution performed.

VC 'RECEIVER#8.1' is trivially true:

```
   (IMPLIES (PMAPP QUEUE)
          (PMAPP QUEUE))

--------------------

VC 'RECEIVER#8.2' is trivially true:

   (IMPLIES (NUMBERP NEXT)
            (NUMBERP NEXT))

--------------------

VC 'RECEIVER#8.3' is trivially true:

   (IMPLIES (NUMBERP EDGE)
            (NUMBERP EDGE))

--------------------

VC 'RECEIVER#5.0.2.1' subsumes VC 'RECEIVER#8.4.1.1':

   (IMPLIES
     (AND
       (ALL (FAPPLY (QUOTE CONTIG)
                    (APR PKT.IN PKT)))
       (OR
         (NOT (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (JOIN (FLATTEN PKT.IN)
                                                  PKT))
                                    (ADD1 NEXT)))))
         (LESSP (REACH (REDUCE (QUOTE WITH)
                               (NULL)
                               (JOIN (FLATTEN PKT.IN)
                                     PKT)))
                NEXT)
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (JOIN (FLATTEN PKT.IN)
                                 PKT)))))
         (IF
           (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                               PKT)))
           (OR
             (NOT (LESSP 0 NEXT))
             (NOT
                (EQUAL (FLATTEN SINK)
                       (RANGE (LOWER (REDUCE (QUOTE WITH)
                                             (NULL)
                                             (JOIN (FLATTEN PKT.IN)
                                                   PKT))
                                     (SUB1 NEXT))))))
            (OR (NOT (EQUAL NEXT 0))
                (NOT (EQUAL SINK (NULL))))))
        (PMAPP QUEUE)
        (NUMBERP NEXT)
        (NUMBERP EDGE)
        (OR (EQUAL PKT (NULL))
            (AND (NOT (LESSP (DOM (LST PKT))
                             NEXT))
                 (NOT (LESSP NEXT (DOM (FIRST PKT))))
                 (SEQP PKT)
                 (LESSP EDGE NEXT)))
        (PMAPP PKT)
        (CONSISTENT (QUOTE DOM)
                    (QUOTE EQUAL)
                    (JOIN (FLATTEN PKT.IN)
                          PKT)))
      (CONSISTENT (QUOTE DOM)
                  (QUOTE EQUAL)
                  (FLATTEN PKT.IN)))
```

VC 'RECEIVER#5.0.2.2' subsumes VC 'RECEIVER#8.4.1.2':

```
   (IMPLIES
     (AND
       (CONSISTENT (QUOTE DOM)
                   (QUOTE EQUAL)
                   (JOIN (FLATTEN PKT.IN)
                         PKT))
       (OR
         (NOT (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (JOIN (FLATTEN PKT.IN)
                                                  PKT))
                                    (ADD1 NEXT)))))
         (LESSP (REACH (REDUCE (QUOTE WITH)
                               (NULL)
```

```
                              (JOIN (FLATTEN PKT.IN)
                                    PKT)))
                NEXT)
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                          (JOIN (FLATTEN PKT.IN)
                                PKT))))
         (IF
           (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                              PKT)))
           (OR
             (NOT (LESSP 0 NEXT))
             (NOT
               (EQUAL (FLATTEN SINK)
                      (RANGE (LOWER (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (JOIN (FLATTEN PKT.IN)
                                                  PKT))
                                    (SUB1 NEXT))))))
           (OR (NOT (EQUAL NEXT 0))
               (NOT (EQUAL SINK (NULL))))))
       (PMAPP QUEUE)
       (NUMBERP NEXT)
       (NUMBERP EDGE)
       (OR (EQUAL PKT (NULL))
           (AND (NOT (LESSP (DOM (LST PKT))
                            NEXT))
                (NOT (LESSP NEXT (DOM (FIRST PKT))))
                (SEQP PKT)
                (LESSP EDGE NEXT)))
       (PMAPP PKT)
       (ALL (FAPPLY (QUOTE CONTIG)
                    (APR PKT.IN PKT))))
     (ALL (FAPPLY (QUOTE CONTIG)
                  PKT.IN)))

--------------------

Generating new VC 'RECEIVER#8.4.2.1':

   (IMPLIES (AND (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                              (NULL)
                                              (FLATTEN PKT.IN))
                                      (ADD1 NEXT)))
                 (ALL (FAPPLY (QUOTE MPAIRP)
                                   (FLATTEN PKT.IN)))
                       (CONSISTENT (QUOTE DOM)
                                   (QUOTE EQUAL)
                                   (JOIN (FLATTEN PKT.IN)
                                         PKT))
                       (PMAPP QUEUE)
                       (PMAPP PKT))
                  (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                                (NULL)
                                                (JOIN (FLATTEN PKT.IN)
                                                      PKT))
                                        (ADD1 NEXT))))


VC not proved, being written.

To be proved with lemmas (FOLLOWS.REDUCE.WITH.3)


-------------------

Generating new VC 'RECEIVER#8.4.2.2.1':

   (IMPLIES (AND (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN PKT.IN)))
                                 NEXT))
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.IN)))
                 (IN 0 (DOMAIN (FLATTEN PKT.IN)))
                 (LESSP 0 NEXT)
                 (NUMBERP NEXT)
                 (PMAPP PKT))
            (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (JOIN (FLATTEN PKT.IN)
                                             PKT)))
                        NEXT)))


VC not proved, being written.

-------------------
```

```
Generating new VC 'RECEIVER#8.4.2.2.2':

   (IMPLIES (AND (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.IN)))
                 (NOT (IN 0 (DOMAIN (FLATTEN PKT.IN))))
                 (EQUAL NEXT 0)
                 (PMAPP PKT))
            (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                       (NULL)
                                       (JOIN (FLATTEN PKT.IN)
                                             PKT)))
                        NEXT)))


VC not proved, being written.

-------------------

VC 'RECEIVER#5.4.2.3' subsumes VC 'RECEIVER#8.4.2.3':

   (IMPLIES
     (AND
       (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                     (NULL)
                                     (FLATTEN PKT.IN))
                             (ADD1 NEXT)))
       (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                  (NULL)
                                  (FLATTEN PKT.IN)))
                   NEXT))
       (IF (IN 0 (DOMAIN (FLATTEN PKT.IN)))
           (AND (LESSP 0 NEXT)
                (EQUAL (FLATTEN SINK)
                       (RANGE (LOWER (REDUCE (QUOTE WITH)
                                             (NULL)
                                             (FLATTEN PKT.IN))
                                     (SUB1 NEXT)))))
           (AND (EQUAL NEXT 0)
                (EQUAL SINK (NULL))))
       (CONSISTENT (QUOTE DOM)
                   (QUOTE EQUAL)
                   (JOIN (FLATTEN PKT.IN)
                         PKT))
       (ALL (FAPPLY (QUOTE CONTIG)
                        (APR PKT.IN PKT)))
        (PMAPP QUEUE)
        (NUMBERP NEXT)
        (NUMBERP EDGE)
        (OR (EQUAL PKT (NULL))
            (AND (NOT (LESSP (DOM (LST PKT))
                             NEXT))
                 (NOT (LESSP NEXT (DOM (FIRST PKT))))
                 (SEQP PKT)
                 (LESSP EDGE NEXT)))
        (PMAPP PKT)
        (ALL (FAPPLY (QUOTE MPAIRP)
                     (FLATTEN PKT.IN))))
      (ALL (FAPPLY (QUOTE MPAIRP)
                   (JOIN (FLATTEN PKT.IN)
                         PKT))))
```

VC 'RECEIVER#8.4.2.4.1.1' is trivially true:

```
   (IMPLIES (LESSP 0 NEXT)
            (LESSP 0 NEXT))
```

Generating new VC 'RECEIVER#8.4.2.4.1.2':

```
   (IMPLIES (AND (EQUAL (FLATTEN SINK)
                        (RANGE (LOWER (REDUCE (QUOTE WITH)
                                              (NULL)
                                              (FLATTEN PKT.IN))
                                      (SUB1 NEXT))))
                 (LESSP 0 NEXT)
                 (IN 0 (DOMAIN (FLATTEN PKT.IN)))
                 (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN PKT.IN)))
                             NEXT))
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.IN)))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                  PKT))
                   (NUMBERP NEXT)
                   (PMAPP PKT))
               (EQUAL (FLATTEN SINK)
                      (RANGE (LOWER (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (JOIN (FLATTEN PKT.IN)
                                                  PKT))
                                    (SUB1 NEXT)))))
```

VC not proved, being written.

To be proved with lemmas (LOWER.REDUCE.WITH.CONSISTENT)

Generating new VC 'RECEIVER#8.4.2.4.2.1':

```
   (IMPLIES (AND (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                                     PKT)))
                 (EQUAL PKT (NULL)))
            (IN 0 (DOMAIN (FLATTEN PKT.IN))))
```

VC not proved, being written.

Generating new VC 'RECEIVER#8.4.2.4.2.2':

```
   (IMPLIES (AND (EQUAL NEXT 0)
                 (LESSP EDGE NEXT))
            F)
```

This formula can be simplified, using the abbreviations NOT and OR, to:

```
      (IMPLIES (EQUAL NEXT 0)
               (NOT (LESSP EDGE NEXT))),
```

which simplifies, using linear arithmetic, to:

```
    T.

Q.E.D.

VC has been proved.

Compute time:    7.111 sec,   GC time:    3.787 sec.

--------------------

VC 'RECEIVER#5.4.2.4.3' subsumes VC 'RECEIVER#8.4.2.4.3':

   (IMPLIES (AND (OR (NOT (EQUAL NEXT 0))
                     (NOT (EQUAL SINK (NULL))))
                 (LESSP 0 NEXT)
                 (EQUAL (FLATTEN SINK)
                        (RANGE (LOWER (REDUCE (QUOTE WITH)
                                              (NULL)
                                              (FLATTEN PKT.IN))
                                      (SUB1 NEXT))))
                 (FOLLOWS QUEUE (UPPER (REDUCE (QUOTE WITH)
                                               (NULL)
                                               (FLATTEN PKT.IN))
                                       (ADD1 NEXT)))
                 (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                            (NULL)
                                            (FLATTEN PKT.IN)))
                             NEXT))
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.IN)))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (JOIN (FLATTEN PKT.IN)
                                   PKT))
                 (ALL (FAPPLY (QUOTE CONTIG)
                              (APR PKT.IN PKT)))
                 (PMAPP QUEUE)
                 (NUMBERP NEXT)
                 (NUMBERP EDGE)
                 (OR (EQUAL PKT (NULL))
                     (AND (NOT (LESSP (DOM (LST PKT))
                                      NEXT))
                          (NOT (LESSP NEXT (DOM (FIRST PKT))))
                          (SEQP PKT)
```

```
                         (LESSP EDGE NEXT)))
                (PMAPP PKT)
                (IN 0 (DOMAIN (FLATTEN PKT.IN))))
           (IN 0 (DOMAIN (JOIN (FLATTEN PKT.IN)
                                    PKT))))

------------------

VC 'RECEIVER#8.4.2.4.4.1' is trivially true:

   (IMPLIES (EQUAL NEXT 0)
            (EQUAL NEXT 0))

------------------

VC 'RECEIVER#8.4.2.4.4.2' is trivially true:

   (IMPLIES (EQUAL SINK (NULL))
            (EQUAL SINK (NULL)))


------------------------------------


The totals are:      41 trivially true VCs,
                     20 subsumed VCs,
                     5 proved VCs, and
                     23 generated VCs.

Compute time:   140.753 seconds,    GC time:     3.787 seconds.
```

APPENDIX B
VC PROOFS

VC Proof Log   24-Jun-82 06:43:07

++++++++++++++++++++++++++++++++++++

Proof of VC 'NANOTCP#1'

```
     (IMPLIES (AND (SENDER.EXT SOURCE ACK.IN RECEIPTS PKT.OUT)
                   (RECEIVER.EXT CREDITS PKT.IN SINK ACK.OUT)
                   (FOLLOWS PKT.IN PKT.OUT)
                   (ALL (FAPPLY (QUOTE PMAPP) PKT.OUT)))
              (INITIAL (FLATTEN SINK)
                       (FLATTEN SOURCE)))
```

This formula can be simplified, using the abbreviations SENDER.EXT, AND, and IMPLIES, to:$$

```
      (IMPLIES (AND (CONSISTENT (QUOTE DOM)
                                (QUOTE EQUAL)
                                (FLATTEN PKT.OUT))
                    (ALL (FAPPLY (QUOTE CONTIG) PKT.OUT))
                    (EQUAL (FLATTEN SOURCE)
                           (RANGE (REDUCE (QUOTE WITH)
                                          (QUOTE (1QUOTE NULL))
                                          (FLATTEN PKT.OUT))))
                    (RECEIVER.EXT CREDITS PKT.IN SINK ACK.OUT)
                    (FOLLOWS PKT.IN PKT.OUT)
                    (ALL (FAPPLY (QUOTE PMAPP) PKT.OUT)))
               (INITIAL (FLATTEN SINK)
                        (FLATTEN SOURCE))),
```

which we simplify, applying IN.DOMAIN.REDUCE.WITH, ALL.FAPPLY.MPAIRP.FLATTEN, ALL.FAPPLY.FOLLOWS, and CONSISTENT.FLATTEN.FOLLOWS, and opening up IN, DOMAIN, PMAPP, RECEIVER.EXT, FLATTEN, SEQP, and INITIAL, to:$$

```
      (IMPLIES
          (AND (CONSISTENT (QUOTE DOM)
                           (QUOTE EQUAL)
                           (FLATTEN PKT.OUT))
               (ALL (FAPPLY (QUOTE CONTIG) PKT.OUT))
               (EQUAL (FLATTEN SOURCE)
                      (RANGE (REDUCE (QUOTE WITH)
                                     (QUOTE (1QUOTE NULL))
                                     (FLATTEN PKT.OUT))))
                (IN 0 (DOMAIN (FLATTEN PKT.IN)))
                (INITIAL (FLATTEN SINK)
                         (RANGE (CONSEC (REDUCE (QUOTE WITH)
                                                (QUOTE (1QUOTE NULL))
                                                (FLATTEN PKT.IN)))))
                (FOLLOWS PKT.IN PKT.OUT)
                (ALL (FAPPLY (QUOTE PMAPP) PKT.OUT)))
            (INITIAL (FLATTEN SINK)
                     (FLATTEN SOURCE))).
```

We use the above equality hypothesis by substituting:

```
     (RANGE (REDUCE (QUOTE WITH)
                    (QUOTE (1QUOTE NULL))
                    (FLATTEN PKT.OUT)))
```

for (FLATTEN SOURCE) and keeping the equality hypothesis. The result is:$$

```
     (IMPLIES
         (AND (CONSISTENT (QUOTE DOM)
                          (QUOTE EQUAL)
                          (FLATTEN PKT.OUT))
              (ALL (FAPPLY (QUOTE CONTIG) PKT.OUT))
              (EQUAL (FLATTEN SOURCE)
                     (RANGE (REDUCE (QUOTE WITH)
                                    (QUOTE (1QUOTE NULL))
                                    (FLATTEN PKT.OUT))))
              (IN 0 (DOMAIN (FLATTEN PKT.IN)))
              (INITIAL (FLATTEN SINK)
                       (RANGE (CONSEC (REDUCE (QUOTE WITH)
                                              (QUOTE (1QUOTE NULL))
                                              (FLATTEN PKT.IN)))))
              (FOLLOWS PKT.IN PKT.OUT)
              (ALL (FAPPLY (QUOTE PMAPP) PKT.OUT)))
         (INITIAL (FLATTEN SINK)
                  (RANGE (REDUCE (QUOTE WITH)
                                 (QUOTE (1QUOTE NULL))
                                 (FLATTEN PKT.OUT))))),
```

which further simplifies, applying the lemmas INITIAL.RANGE, PMAPP.REDUCE.WITH, IN.DOMAIN.REDUCE.WITH, ALL.FAPPLY.FOLLOWS, FOLLOWS.REDUCE.WITH, FOLLOWS.FLATTEN, ALL.FAPPLY.MPAIRP.FLATTEN, INITIAL.CONSEC.FOLLOWS, and INITIAL.TRANS, and expanding the function PMAPP, to:

```
      T.

Q.E.D.

73672 conses
81.063 seconds
20.092 seconds, garbage collection time

------------------------------------------

VC Proof Log                                    24-Jun-82 06:45:59

++++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#1'

     (SENDER.INT (NULL)
                 (NULL)
                 (NULL)
                 (NULL)
                 0 0 0
                 (NULL)
                 F 0)

This conjecture can be simplified, using the abbreviations CONTIG
and SENDER.INT, to ten new goals:

Case 10.(PMAPP (QUOTE (1QUOTE NULL))).

  This simplifies, unfolding the definition of PMAPP, to:

        T.

Case 9. (NUMBERP 0),

  which we simplify, clearly, to:

        T.

Case 8. (ALL (FAPPLY (QUOTE MPAIRP)
                    (FLATTEN (QUOTE (1QUOTE NULL))))),

     which simplifies, opening up the definitions of FLATTEN, SEQP,
     FAPPLY, and ALL, to:

           T.

   Case 7. (ALL (FAPPLY (QUOTE CONTIG)
                        (QUOTE (1QUOTE NULL)))),

     which we simplify, opening up SEQP, FAPPLY, and ALL, to:

           T.

   Case 6. (CONSECP (DOMAIN (QUOTE (1QUOTE NULL)))).

     This simplifies, opening up the definitions of DOMAIN and CONSECP,
     to:

           T.

   Case 5. (CONSISTENT (QUOTE DOM)
                       (QUOTE EQUAL)
                       (FLATTEN (QUOTE (1QUOTE NULL)))).

     This simplifies, appealing to the lemma
     CONSISTENT.DOM.EQUAL.PMAPP, and unfolding the definitions of
     FLATTEN and PMAPP, to:

           T.

   Case 4. (FOLLOWS (QUOTE (1QUOTE NULL))
                    (FLATTEN (QUOTE (1QUOTE NULL)))).

     This simplifies, opening up FLATTEN and FOLLOWS, to:

           T.

   Case 3. (EQUAL (FLATTEN (QUOTE (1QUOTE NULL)))
                  (RANGE (REDUCE (QUOTE WITH)
                                 (QUOTE (1QUOTE NULL))
                                 (FLATTEN (QUOTE (1QUOTE NULL)))))),

     which simplifies, rewriting with REDUCE.WITH.NULL, and expanding
```

```
  FLATTEN, PMAPP, RANGE, and EQUAL, to:

        T.

Case 2. (EQUAL (HIGHEST (DOMAIN (FLATTEN (QUOTE (1QUOTE NULL)))))
               (SUB1 0)),

  which we simplify, opening up the definitions of FLATTEN, DOMAIN,
  HIGHEST, SUB1, and EQUAL, to:

        T.

Case 1. (IF
         (EQUAL 0 0)
         (IF (EQUAL (FLATTEN (QUOTE (1QUOTE NULL)))
                    (QUOTE (1QUOTE NULL)))
             (EQUAL (QUOTE (1QUOTE NULL))
                    (QUOTE (1QUOTE NULL)))
             F)
         (IF
          (SEQP (FLATTEN (QUOTE (1QUOTE NULL))))
          (IF
           (EQUAL
                (DOM (LST (REDUCE (QUOTE WITH)
                                  (QUOTE (1QUOTE NULL))
                                  (FLATTEN (QUOTE (1QUOTE NULL))))))
                (SUB1 0))
           (IF (SEQP (QUOTE (1QUOTE NULL)))
               (EQUAL (DOM (LST (QUOTE (1QUOTE NULL))))
                      (SUB1 0))
               T)
           F)
          F)).

  This simplifies, unfolding EQUAL and FLATTEN, to:

        T.

Q.E.D.

1575 conses
4.267 seconds
0.0 seconds, garbage collection time
```

++

Proof of VC 'SENDER#2'

```
     (IMPLIES (SENDER.INT SOURCE ACK.IN RECEIPTS PKT.OUT UNACK NEXT
                          EDGE QUEUE TIMING TO.TIME)
              (SENDER.EXT SOURCE ACK.IN RECEIPTS PKT.OUT))
```

This conjecture can be simplified, using the abbreviations CONTIG, SENDER.INT, and IMPLIES, to the conjecture:$$

```
      (IMPLIES
       (AND
          (PMAPP QUEUE)
          (NUMBERP NEXT)
          (ALL (FAPPLY (QUOTE MPAIRP)
                       (FLATTEN PKT.OUT)))
          (ALL (FAPPLY (QUOTE CONTIG) PKT.OUT))
          (CONSECP (DOMAIN QUEUE))
          (CONSISTENT (QUOTE DOM)
                      (QUOTE EQUAL)
                      (FLATTEN PKT.OUT))
          (FOLLOWS QUEUE (FLATTEN PKT.OUT))
          (EQUAL (FLATTEN SOURCE)
                 (RANGE (REDUCE (QUOTE WITH)
                                (QUOTE (1QUOTE NULL))
                                (FLATTEN PKT.OUT))))
          (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                 (SUB1 NEXT))
          (IF (EQUAL NEXT 0)
              (IF (EQUAL (FLATTEN PKT.OUT)
                         (QUOTE (1QUOTE NULL)))
                  (EQUAL QUEUE (QUOTE (1QUOTE NULL)))
                  F)
              (IF (SEQP (FLATTEN PKT.OUT))
                  (IF (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                               (QUOTE (1QUOTE NULL))
```

```
                                                        (FLATTEN PKT.OUT))))
                                   (SUB1 NEXT))
                            (IF (SEQP QUEUE)
                                (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT))
                                T)
                            F)
                        F)))
      (SENDER.EXT SOURCE ACK.IN RECEIPTS PKT.OUT)),
```

which we simplify, applying REDUCE.WITH.NULL and CONSISTENT.DOM.EQUAL.PMAPP, and expanding the definitions of SENDER.EXT, RANGE, and PMAPP, to the conjecture:

```
      (IMPLIES (AND (PMAPP QUEUE)
                    (NUMBERP NEXT)
                    (ALL (FAPPLY (QUOTE MPAIRP)
                                 (FLATTEN PKT.OUT)))
                    (ALL (FAPPLY (QUOTE CONTIG) PKT.OUT))
                    (CONSECP (DOMAIN QUEUE))
                    (CONSISTENT (QUOTE DOM)
                                (QUOTE EQUAL)
                                (FLATTEN PKT.OUT))
                    (FOLLOWS QUEUE (FLATTEN PKT.OUT))
                    (EQUAL (FLATTEN SOURCE)
                           (RANGE (REDUCE (QUOTE WITH)
                                          (QUOTE (1QUOTE NULL))
                                          (FLATTEN PKT.OUT))))
                    (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                           (SUB1 NEXT))
                    (EQUAL NEXT 0)
                    (EQUAL (FLATTEN PKT.OUT)
                           (QUOTE (1QUOTE NULL)))
                    (EQUAL QUEUE (QUOTE (1QUOTE NULL))))
               (EQUAL (FLATTEN SOURCE)
                      (QUOTE (1QUOTE NULL)))),
```

which we again simplify, using linear arithmetic, to:

```
      (IMPLIES (AND (EQUAL 0 0)
                    (PMAPP (QUOTE (1QUOTE NULL)))
                    (NUMBERP 0)
                    (ALL (FAPPLY (QUOTE MPAIRP)
                                 (QUOTE (1QUOTE NULL))))
                    (ALL (FAPPLY (QUOTE CONTIG) PKT.OUT))
```

```
                        (CONSECP (DOMAIN (QUOTE (1QUOTE NULL))))
                        (CONSISTENT (QUOTE DOM)
                                    (QUOTE EQUAL)
                                    (QUOTE (1QUOTE NULL)))
                        (FOLLOWS (QUOTE (1QUOTE NULL))
                                 (QUOTE (1QUOTE NULL)))
                        (EQUAL (FLATTEN SOURCE)
                               (RANGE (REDUCE (QUOTE WITH)
                                              (QUOTE (1QUOTE NULL))
                                              (QUOTE (1QUOTE NULL)))))
                        (EQUAL (HIGHEST (DOMAIN (QUOTE (1QUOTE NULL))))
                               (SUB1 0))
                        (EQUAL (FLATTEN PKT.OUT)
                               (QUOTE (1QUOTE NULL))))
                    (EQUAL (FLATTEN SOURCE)
                           (QUOTE (1QUOTE NULL))))).
```

However this simplifies again, applying CONSISTENT.DOM.EQUAL.PMAPP and REDUCE.WITH.NULL, and unfolding EQUAL, PMAPP, NUMBERP, SEQP, FAPPLY, ALL, DOMAIN, CONSECP, FOLLOWS, and RANGE, to:

```
      T.
```

Q.E.D.

27601 conses
31.152 seconds
6.655 seconds, garbage collection time

VC Proof Log   24-Jun-82 06:48:50

++

Proof of VC 'SENDER#3.0.1.1'

```
     (OR (EQUAL (UNION QUEUE (ENMAP MESS NEXT))
                (JOIN QUEUE (ENMAP MESS NEXT)))
         (NOT (EQUAL QUEUE (NULL))))
```

```
This conjecture can be simplified, using the abbreviations NOT, OR,
and UNION, to:

      (IMPLIES (NOT (EQUAL (REDUCE (QUOTE WITH)
                                   QUEUE
                                   (ENMAP MESS NEXT))
                           (JOIN QUEUE (ENMAP MESS NEXT))))
               (NOT (EQUAL QUEUE (QUOTE (1QUOTE NULL))))).

This simplifies, applying the lemmas PMAPP.ENMAP, REDUCE.WITH.NULL,
and JOIN.NULL.PMAPP, to:

      T.

Q.E.D.

1655 conses
4.463 seconds
0.0 seconds, garbage collection time


----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#3.0.1.2'

     (OR (EQUAL (UNION QUEUE (ENMAP MESS NEXT))
                (JOIN QUEUE (ENMAP MESS NEXT)))
         (NOT (PMAPP QUEUE))
         (NOT (NUMBERP NEXT))
         (EQUAL NEXT 0)
         (IF (SEQP QUEUE)
             (NOT (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT)))
             F)
         (NOT (SEQP MESS)))

This conjecture can be simplified, using the abbreviations NOT, OR,
and UNION, to:$
```

```
      (IMPLIES
           (AND (NOT (EQUAL (REDUCE (QUOTE WITH)
                                    QUEUE
                                    (ENMAP MESS NEXT))
                            (JOIN QUEUE (ENMAP MESS NEXT))))
                (PMAPP QUEUE)
                (NUMBERP NEXT)
                (NOT (EQUAL NEXT 0))
                (NOT (IF (SEQP QUEUE)
                         (NOT (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT)))
                         F)))
           (NOT (SEQP MESS))).
```

This simplifies, opening up the definition of NOT, to two new formulas:

```
Case 2. (IMPLIES (AND (NOT (EQUAL (REDUCE (QUOTE WITH)
                                          QUEUE
                                          (ENMAP MESS NEXT))
                                  (JOIN QUEUE (ENMAP MESS NEXT))))
                      (PMAPP QUEUE)
                      (NUMBERP NEXT)
                      (NOT (EQUAL NEXT 0))
                      (NOT (SEQP QUEUE)))
                 (NOT (SEQP MESS))).
```

But this simplifies again, using linear arithmetic, applying PMAPP.ENMAP, DOM.MPAIR, FIRST.ENMAP, LST.NSEQP, and REDUCE.WITH.LESSP, and expanding the definition of DOM, to:

```
      T.
```

```
Case 1. (IMPLIES (AND (NOT (EQUAL (REDUCE (QUOTE WITH)
                                          QUEUE
                                          (ENMAP MESS NEXT))
                                  (JOIN QUEUE (ENMAP MESS NEXT))))
                      (PMAPP QUEUE)
                      (NUMBERP NEXT)
                      (NOT (EQUAL NEXT 0))
                      (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT)))
                 (NOT (SEQP MESS))),
```

which again simplifies, using linear arithmetic and applying

```
  PMAPP.ENMAP, DOM.MPAIR, FIRST.ENMAP, and REDUCE.WITH.LESSP, to:

        T.

Q.E.D.

14118 conses
15.705 seconds
3.335 seconds, garbage collection time

----------------------------------------





++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#3.0.2.1'

     (OR (EQUAL (REDUCE (QUOTE WITH)
                        (NULL)
                        (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
                (JOIN (REDUCE (QUOTE WITH)
                              (NULL)
                              (FLATTEN PKT.OUT))
                      (ENMAP MESS NEXT)))
         (NOT (EQUAL (FLATTEN PKT.OUT) (NULL))))

This conjecture can be simplified, using the abbreviations NOT and
OR, to:$

      (IMPLIES
       (NOT
           (EQUAL (REDUCE (QUOTE WITH)
                          (QUOTE (1QUOTE NULL))
                          (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
                  (JOIN (REDUCE (QUOTE WITH)
                                (QUOTE (1QUOTE NULL))
                                (FLATTEN PKT.OUT))
                        (ENMAP MESS NEXT))))
       (NOT (EQUAL (FLATTEN PKT.OUT)
                   (QUOTE (1QUOTE NULL))))).
```

This simplifies, applying the lemmas LST.APR, NLST.APR, REDUCE.JOIN, REDUCE.WITH.NULL, PMAPP.ENMAP, and JOIN.NULL.PMAPP, and opening up FLATTEN and PMAPP, to:

```
      (IMPLIES (NOT (EQUAL (REDUCE (QUOTE WITH)
                                   (FLATTEN PKT.OUT)
                                   (ENMAP MESS NEXT))
                           (ENMAP MESS NEXT)))
               (NOT (EQUAL (FLATTEN PKT.OUT)
                           (QUOTE (1QUOTE NULL))))),
```

which we again simplify, applying PMAPP.ENMAP and REDUCE.WITH.NULL, to:

```
      T.
```

Q.E.D.

```
7939 conses
8.053 seconds
3.28 seconds, garbage collection time
```

++

Proof of VC 'SENDER#3.0.2.2'

```
     (OR (EQUAL (REDUCE (QUOTE WITH)
                        (NULL)
                        (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
                (JOIN (REDUCE (QUOTE WITH)
                              (NULL)
                              (FLATTEN PKT.OUT))
                      (ENMAP MESS NEXT)))
         (NOT (NUMBERP NEXT))
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.OUT))))
```

```
          (EQUAL NEXT 0)
          (NOT (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                        (NULL)
                                        (FLATTEN PKT.OUT))))
                      (SUB1 NEXT)))
          (NOT (SEQP MESS)))
```

This formula can be simplified, using the abbreviations NOT and OR, to:

```
      (IMPLIES
       (AND
        (NOT
           (EQUAL (REDUCE (QUOTE WITH)
                          (QUOTE (1QUOTE NULL))
                          (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
                  (JOIN (REDUCE (QUOTE WITH)
                                (QUOTE (1QUOTE NULL))
                                (FLATTEN PKT.OUT))
                        (ENMAP MESS NEXT))))
        (NUMBERP NEXT)
        (ALL (FAPPLY (QUOTE MPAIRP)
                     (FLATTEN PKT.OUT)))
        (NOT (EQUAL NEXT 0))
        (EQUAL (DOM (LST (REDUCE (QUOTE WITH)
                                 (QUOTE (1QUOTE NULL))
                                 (FLATTEN PKT.OUT))))
               (SUB1 NEXT)))
       (NOT (SEQP MESS))).
```

This simplifies, using linear arithmetic, applying LST.APR, NLST.APR, REDUCE.JOIN, PMAPP.ENMAP, PMAPP.REDUCE.WITH, DOM.MPAIR, FIRST.ENMAP, and REDUCE.WITH.LESSP, and expanding the functions FLATTEN and PMAPP, to:

      T.

Q.E.D.

10022 conses
10.151 seconds
0.0 seconds, garbage collection time

```
----------------------------------------



++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#3.1.1'

     (OR (PMAPP (JOIN QUEUE (ENMAP MESS NEXT)))
         (NOT (EQUAL QUEUE (NULL))))

This formula can be simplified, using the abbreviations NOT and OR,
to the goal:

      (IMPLIES (NOT (PMAPP (JOIN QUEUE (ENMAP MESS NEXT))))
               (NOT (EQUAL QUEUE (QUOTE (1QUOTE NULL))))).

This simplifies, appealing to the lemmas PMAPP.ENMAP and
JOIN.NULL.PMAPP, to:

      T.

Q.E.D.

777 conses
1.035 seconds
0.0 seconds, garbage collection time


----------------------------------------



++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#3.1.2'

     (OR (PMAPP (JOIN QUEUE (ENMAP MESS NEXT)))
         (NOT (PMAPP QUEUE))
         (NOT (NUMBERP NEXT))
```

```
          (EQUAL NEXT 0)
          (IF (SEQP QUEUE)
              (NOT (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT)))
              F)
          (NOT (SEQP MESS)))
```

$ This formula can be simplified, using the abbreviations NOT and OR, to the goal:

```
      (IMPLIES
           (AND (NOT (PMAPP (JOIN QUEUE (ENMAP MESS NEXT))))
                (PMAPP QUEUE)
                (NUMBERP NEXT)
                (NOT (EQUAL NEXT 0))
                (NOT (IF (SEQP QUEUE)
                         (NOT (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT)))
                         F)))
           (NOT (SEQP MESS))).
```

This simplifies, appealing to the lemmas PMAPP.ENMAP, PMAPP.JOIN.2, SEQP.ENMAP, FIRST.ENMAP, and DOM.MPAIR, and expanding the definition of NOT, to:

```
      (IMPLIES (AND (SEQP QUEUE)
                    (NOT (LESSP (DOM (LST QUEUE)) NEXT))
                    (PMAPP QUEUE)
                    (NUMBERP NEXT)
                    (NOT (EQUAL NEXT 0))
                    (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT)))
               (NOT (SEQP MESS))),
```

which again simplifies, using linear arithmetic, to:

```
      T.
```

Q.E.D.

```
2679 conses
3.689 seconds
3.277 seconds, garbage collection time
[259 cns / .6 s + 0.0 gc + 0.0 io (= 1  0)]
```

```
++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#3.3'

     (OR (ALL (FAPPLY (QUOTE MPAIRP)
                      (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.OUT)))))

This conjecture can be simplified, using the abbreviations NOT and
OR, to the conjecture:

      (IMPLIES
       (NOT
           (ALL (FAPPLY (QUOTE MPAIRP)
                        (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))))
       (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                         (FLATTEN PKT.OUT))))),

which we simplify, applying LST.APR, NLST.APR, FAPPLY.JOIN,
ALL.JOIN, PMAPP.ENMAP, and ALL.FAPPLY.MPAIRP, and expanding the
definition of FLATTEN, to:

      T.

Q.E.D.

1945 conses
2.668 seconds
0.0 seconds, garbage collection time


----------------------------------------




++++++++++++++++++++++++++++++++++++++++
```

Proof of VC 'SENDER#3.4'

```
     (OR (ALL (FAPPLY (QUOTE CONTIG)
                      (APR PKT.OUT (ENMAP MESS NEXT))))
         (NOT (ALL (FAPPLY (QUOTE CONTIG) PKT.OUT))))
```

This formula can be simplified, using the abbreviations NOT and OR, to:

```
      (IMPLIES (NOT (ALL (FAPPLY (QUOTE CONTIG)
                                 (APR PKT.OUT (ENMAP MESS NEXT)))))
               (NOT (ALL (FAPPLY (QUOTE CONTIG) PKT.OUT)))),
```

which we simplify, applying CONSECP.DOMAIN.ENMAP, APPLY1.CONTIG, LST.APR, and NLST.APR, and opening up CONTIG, FAPPLY, and ALL, to:

```
      T.
```

Q.E.D.

```
919 conses
1.456 seconds
0.0 seconds, garbage collection time
```

++

Proof of VC 'SENDER#3.5.1'

```
     (OR (CONTIG (JOIN QUEUE (ENMAP MESS NEXT)))
         (NOT (EQUAL QUEUE (NULL))))
```

This formula can be simplified, using the abbreviations NOT, OR, DOMAIN.JOIN, and CONTIG, to the goal:

```
      (IMPLIES (NOT (CONSECP (JOIN (DOMAIN QUEUE)
                                   (DOMAIN (ENMAP MESS NEXT)))))
```

```
                    (NOT (EQUAL QUEUE (QUOTE (1QUOTE NULL))))).

This simplifies, appealing to the lemmas PISEQP.DOMAIN.2,
JOIN.NULL.PISEQP, and CONSECP.DOMAIN.ENMAP, and expanding the
definition of DOMAIN, to:

      T.

Q.E.D.

857 conses
1.345 seconds
0.0 seconds, garbage collection time


----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#3.5.2'

     (OR (CONTIG (JOIN QUEUE (ENMAP MESS NEXT)))
         (NOT (CONTIG QUEUE))
         (NOT (PMAPP QUEUE))
         (NOT (NUMBERP NEXT))
         (EQUAL NEXT 0)
         (IF (SEQP QUEUE)
             (NOT (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT)))
             F)
         (NOT (SEQP MESS)))

This formula can be simplified, using the abbreviations NOT, OR,
DOMAIN.JOIN, and CONTIG, to:

      (IMPLIES
           (AND (NOT (CONSECP (JOIN (DOMAIN QUEUE)
                                    (DOMAIN (ENMAP MESS NEXT)))))
                (CONSECP (DOMAIN QUEUE))
                (PMAPP QUEUE)
                (NUMBERP NEXT)
                (NOT (EQUAL NEXT 0))
                (NOT (IF (SEQP QUEUE)
                         (NOT (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT)))
                         F)))
           (NOT (SEQP MESS))).
```

This simplifies, expanding the definition of NOT, to two new conjectures:

```
Case 2. (IMPLIES
              (AND (NOT (CONSECP (JOIN (DOMAIN QUEUE)
                                       (DOMAIN (ENMAP MESS NEXT)))))
                   (CONSECP (DOMAIN QUEUE))
                   (PMAPP QUEUE)
                   (NUMBERP NEXT)
                   (NOT (EQUAL NEXT 0))
                   (NOT (SEQP QUEUE)))
              (NOT (SEQP MESS))),
```

which again simplifies, applying PISEQP.DOMAIN.2, JOIN.NULL.PISEQP, and CONSECP.DOMAIN.ENMAP, and opening up the function DOMAIN, to:

T.

```
Case 1. (IMPLIES
              (AND (NOT (CONSECP (JOIN (DOMAIN QUEUE)
                                       (DOMAIN (ENMAP MESS NEXT)))))
                   (CONSECP (DOMAIN QUEUE))
                   (PMAPP QUEUE)
                   (NUMBERP NEXT)
                   (NOT (EQUAL NEXT 0))
                   (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT)))
              (NOT (SEQP MESS))),
```

which we again simplify, using linear arithmetic and applying CONSECP.DOMAIN.ENMAP, DOM.MPAIR, FIRST.ENMAP, and CONSECP.JOIN.DOMAIN, to:

T.

Q.E.D.

5857 conses

```
6.779 seconds
0.0 seconds, garbage collection time

----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#3.6.1'

     (OR (CONSISTENT (QUOTE DOM)
                     (QUOTE EQUAL)
                     (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
         (NOT (EQUAL (FLATTEN PKT.OUT) (NULL))))

This formula can be simplified, using the abbreviations NOT and OR,
to the goal:$

      (IMPLIES
        (NOT (CONSISTENT (QUOTE DOM)
                         (QUOTE EQUAL)
                         (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))
        (NOT (EQUAL (FLATTEN PKT.OUT)
                    (QUOTE (1QUOTE NULL))))).

This simplifies, appealing to the lemmas LST.APR and NLST.APR, and
expanding the definition of FLATTEN, to:

      (IMPLIES (NOT (CONSISTENT (QUOTE DOM)
                                (QUOTE EQUAL)
                                (JOIN (FLATTEN PKT.OUT)
                                      (ENMAP MESS NEXT))))
               (NOT (EQUAL (FLATTEN PKT.OUT)
                           (QUOTE (1QUOTE NULL))))),

which again simplifies, applying PMAPP.ENMAP, JOIN.NULL.PMAPP, and
CONSISTENT.DOM.EQUAL.PMAPP, to:

      T.
```

```
Q.E.D.

6850 conses
7.304 seconds
3.314 seconds, garbage collection time

----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#3.6.2'

     (OR (CONSISTENT (QUOTE DOM)
                     (QUOTE EQUAL)
                     (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
         (NOT (CONSISTENT (QUOTE DOM)
                          (QUOTE EQUAL)
                          (FLATTEN PKT.OUT)))
         (NOT (NUMBERP NEXT))
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.OUT))))
         (NOT (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                     (SUB1 NEXT)))
         (EQUAL NEXT 0)
         (NOT (SEQP MESS)))

This formula can be simplified, using the abbreviations NOT and OR,
to:

      (IMPLIES
       (AND
        (NOT (CONSISTENT (QUOTE DOM)
                         (QUOTE EQUAL)
                         (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))
        (CONSISTENT (QUOTE DOM)
                    (QUOTE EQUAL)
                    (FLATTEN PKT.OUT))
        (NUMBERP NEXT)
        (ALL (FAPPLY (QUOTE MPAIRP)
                    (FLATTEN PKT.OUT)))
          (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                 (SUB1 NEXT))
          (NOT (EQUAL NEXT 0)))
         (NOT (SEQP MESS))),
```

which simplifies, using linear arithmetic, applying LST.APR, NLST.APR, DISJOINT.LESSP, SEQP.ENMAP, LOWEST.DOMAIN, FIRST.ENMAP, DOM.MPAIR, FAPPLY.DOM.DOMAIN, CONSISTENT.DOM.EQUAL.PMAPP, PMAPP.ENMAP, and CONSISTENT.JOIN.DISJOINT, and expanding the definition of FLATTEN, to:

```
      T.
```

Q.E.D.

```
4379 conses
5.642 seconds
0.0 seconds, garbage collection time
```

--

++

Proof of VC 'SENDER#3.7'

```
     (OR (FOLLOWS (JOIN QUEUE (ENMAP MESS NEXT))
                  (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
         (NOT (FOLLOWS QUEUE (FLATTEN PKT.OUT))))
```

This formula can be simplified, using the abbreviations NOT and OR, to the goal:

```
      (IMPLIES
           (NOT (FOLLOWS (JOIN QUEUE (ENMAP MESS NEXT))
                         (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))
           (NOT (FOLLOWS QUEUE (FLATTEN PKT.OUT)))).
```

This simplifies, appealing to the lemmas LST.APR, NLST.APR, and

FOLLOWS.JOIN.JOIN, and expanding the definition of FLATTEN, to:

```
      T.
```

Q.E.D.

```
1218 conses
1.631 seconds
0.0 seconds, garbage collection time
```

+++++++++++++++++++++++++++++++++++++++

Proof of VC 'SENDER#3.8'

```
     (OR (EQUAL (FLATTEN (APR SOURCE MESS))
                (RANGE (JOIN (REDUCE (QUOTE WITH)
                                     (NULL)
                                     (FLATTEN PKT.OUT))
                             (ENMAP MESS NEXT))))
         (NOT (EQUAL (FLATTEN SOURCE)
                     (RANGE (REDUCE (QUOTE WITH)
                                    (NULL)
                                    (FLATTEN PKT.OUT)))))
         (NOT (PSEQP MESS)))
```

This formula can be simplified, using the abbreviations NOT, OR, and RANGE.JOIN, to:$

```
      (IMPLIES
         (AND (NOT (EQUAL (FLATTEN (APR SOURCE MESS))
                          (JOIN (RANGE (REDUCE (QUOTE WITH)
                                               (QUOTE (1QUOTE NULL))
                                               (FLATTEN PKT.OUT)))
                                (RANGE (ENMAP MESS NEXT)))))
              (EQUAL (FLATTEN SOURCE)
                     (RANGE (REDUCE (QUOTE WITH)
                                    (QUOTE (1QUOTE NULL))
```

```
                                   (FLATTEN PKT.OUT)))))
        (NOT (PSEQP MESS))),
```

which simplifies, applying LST.APR, NLST.APR, RANGE.ENMAP, and EQUAL.JOIN.JOIN, and expanding the definition of FLATTEN, to:

```
      T.
```

Q.E.D.

```
4442 conses
5.23 seconds
3.3 seconds, garbage collection time
```

```
++++++++++++++++++++++++++++++++++++++++
```

Proof of VC 'SENDER#3.9.1'

```
     (OR
      (EQUAL
        (HIGHEST (DOMAIN (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))
        (SUB1 (PLUS NEXT (SIZE MESS))))
      (NOT (EQUAL NEXT 0))
      (NOT (EQUAL (FLATTEN PKT.OUT) (NULL)))
      (NOT (SEQP MESS)))
```

This conjecture can be simplified, using the abbreviations NOT and OR, to:

```
      (IMPLIES
       (AND
        (NOT
         (EQUAL
          (HIGHEST
                 (DOMAIN (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))
          (SUB1 (PLUS NEXT (SIZE MESS)))))
        (EQUAL NEXT 0)
```

```
   (EQUAL (FLATTEN PKT.OUT)
          (QUOTE (1QUOTE NULL))))
  (NOT (SEQP MESS))).

This simplifies, applying the lemmas LST.APR, NLST.APR, DOMAIN.JOIN,
PISEQP.DOMAIN.2, JOIN.NULL.PISEQP, PMAPP.ENMAP, HIGHEST.DOMAIN,
LST.ENMAP, and DOM.MPAIR, and opening up FLATTEN, DOMAIN, EQUAL,
and PLUS, to:

      T.

Q.E.D.

1874 conses
2.826 seconds
0.0 seconds, garbage collection time


----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#3.9.2'

     (OR
      (EQUAL
        (HIGHEST (DOMAIN (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))
        (SUB1 (PLUS NEXT (SIZE MESS))))
      (NOT (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                  (SUB1 NEXT)))
      (NOT (NUMBERP NEXT))
      (EQUAL NEXT 0)
      (NOT (SEQP MESS)))

This formula can be simplified, using the abbreviations NOT and OR,
to:

      (IMPLIES
       (AND
        (NOT
         (EQUAL
          (HIGHEST
                 (DOMAIN (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))
          (SUB1 (PLUS NEXT (SIZE MESS)))))
        (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
               (SUB1 NEXT))
        (NUMBERP NEXT)
        (NOT (EQUAL NEXT 0)))
       (NOT (SEQP MESS))).
```

This simplifies, applying LST.APR, NLST.APR, DOMAIN.JOIN, PISEQP.DOMAIN.2, HIGHEST.JOIN, PMAPP.ENMAP, HIGHEST.DOMAIN, LST.ENMAP, DOM.MPAIR, and SUB1.ADD1, and expanding the functions FLATTEN, MAX, and PLUS, to the following two new goals:

```
Case 2. (IMPLIES
               (AND (NOT (LESSP (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                                (PLUS NEXT (SUB1 (SIZE MESS))))))
                    (NOT (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                                (PLUS (SUB1 NEXT) (SIZE MESS))))
                    (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                           (SUB1 NEXT))
                    (NUMBERP NEXT)
                    (NOT (EQUAL NEXT 0)))
               (NOT (SEQP MESS))).
```

  This simplifies again, using linear arithmetic and applying the
  lemma LESSP.SIZE.SEQP, to:

```
        T.
```

```
Case 1. (IMPLIES (AND (LESSP (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                             (PLUS NEXT (SUB1 (SIZE MESS))))
                      (NOT (EQUAL (PLUS NEXT (SUB1 (SIZE MESS)))
                                  (PLUS (SUB1 NEXT) (SIZE MESS))))
                      (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                             (SUB1 NEXT))
                      (NUMBERP NEXT)
                      (NOT (EQUAL NEXT 0)))
                 (NOT (SEQP MESS))),
```

  which again simplifies, using linear arithmetic, to the
  conjecture:

```
      (IMPLIES (AND (EQUAL (SIZE MESS) 0)
                    (LESSP (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                           (PLUS NEXT (SUB1 (SIZE MESS))))
                    (NOT (EQUAL (PLUS NEXT (SUB1 (SIZE MESS)))
                                (PLUS (SUB1 NEXT) (SIZE MESS))))
                    (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                           (SUB1 NEXT))
                    (NUMBERP NEXT)
                    (NOT (EQUAL NEXT 0)))
               (NOT (SEQP MESS))),

  which we again simplify, using linear arithmetic, to:$

      (IMPLIES (AND (EQUAL 0 0)
                    (EQUAL (SIZE MESS) 0)
                    (LESSP (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                           (PLUS NEXT (SUB1 0)))
                    (NOT (EQUAL (PLUS NEXT (SUB1 0))
                                (PLUS (SUB1 NEXT) 0)))
                    (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                           (SUB1 NEXT))
                    (NUMBERP NEXT)
                    (NOT (EQUAL NEXT 0)))
               (NOT (SEQP MESS))).

  However this simplifies again, using linear arithmetic and
  applying LESSP.SIZE.SEQP, to:

      T.

Q.E.D.

8574 conses
9.581 seconds
3.474 seconds, garbage collection time


----------------------------------------



++++++++++++++++++++++++++++++++++++++++
Proof of VC  'SENDER#3.10.1.1'

     (OR (NOT (EQUAL (PLUS NEXT (SIZE MESS)) 0))
         (NOT (EQUAL NEXT 0))
         (NOT (SEQP MESS)))

This formula can be simplified, using the abbreviations NOT and OR,
to:

      (IMPLIES (AND (EQUAL (PLUS NEXT (SIZE MESS)) 0)
                    (EQUAL NEXT 0))
               (NOT (SEQP MESS))),

which simplifies, using linear arithmetic and applying
LESSP.SIZE.SEQP, to:

      T.

Q.E.D.

736 conses
1.104 seconds
0.0 seconds, garbage collection time

----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#3.10.2'

     (OR (NOT (EQUAL (PLUS NEXT (SIZE MESS)) 0))
         (NOT (NUMBERP NEXT))
         (EQUAL NEXT 0))

This formula can be simplified, using the abbreviations NOT and OR,
to:

      (IMPLIES (AND (EQUAL (PLUS NEXT (SIZE MESS)) 0)
                        (NUMBERP NEXT))
                   (EQUAL NEXT 0)),

which simplifies, using linear arithmetic, to:

      T.

Q.E.D.

628 conses
.859 seconds
0.0 seconds, garbage collection time

----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#3.10.3.1'

     (OR (SEQP (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT))))
         (NOT (SEQP MESS)))

This formula can be simplified, using the abbreviations NOT and OR,
to the goal:

      (IMPLIES
              (NOT (SEQP (FLATTEN (APR PKT.OUT (ENMAP MESS NEXT)))))
              (NOT (SEQP MESS))).

This simplifies, appealing to the lemmas LST.APR, NLST.APR,
SEQP.JOIN, and SEQP.ENMAP, and expanding the definition of FLATTEN,
to:

      T.

Q.E.D.

803 conses
1.216 seconds
```

```
0.0 seconds, garbage collection time

----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#3.10.3.2'

     (OR (EQUAL (DOM (LST (JOIN (REDUCE (QUOTE WITH)
                                        (NULL)
                                        (FLATTEN PKT.OUT))
                                (ENMAP MESS NEXT))))
                (SUB1 (PLUS NEXT (SIZE MESS))))
         (NOT (SEQP MESS)))

This conjecture can be simplified, using the abbreviations NOT and
OR, to:

      (IMPLIES
           (NOT (EQUAL (DOM (LST (JOIN (REDUCE (QUOTE WITH)
                                               (QUOTE (1QUOTE NULL))
                                               (FLATTEN PKT.OUT))
                                       (ENMAP MESS NEXT))))
                       (SUB1 (PLUS NEXT (SIZE MESS)))))
           (NOT (SEQP MESS))),

which simplifies, rewriting with the lemmas LST.JOIN.2, SEQP.ENMAP,
LST.ENMAP, and DOM.MPAIR, to:

      (IMPLIES (NOT (EQUAL (PLUS NEXT (SUB1 (SIZE MESS)))
                           (SUB1 (PLUS NEXT (SIZE MESS)))))
               (NOT (SEQP MESS))),

which we again simplify, using linear arithmetic, to two new
conjectures:

Case 2. (IMPLIES (AND (EQUAL (SIZE MESS) 0)
                      (NOT (EQUAL (PLUS NEXT (SUB1 (SIZE MESS)))
                                  (SUB1 (PLUS NEXT (SIZE MESS))))))
```

```
                 (NOT (SEQP MESS))).

  But this again simplifies, using linear arithmetic, to two new
  goals:

  Case 2.2.
          (IMPLIES (AND (EQUAL 0 0)
                        (EQUAL (SIZE MESS) 0)
                        (NOT (EQUAL (PLUS NEXT (SUB1 0))
                                    (SUB1 (PLUS NEXT 0)))))
                   (NOT (SEQP MESS))),

    which again simplifies, using linear arithmetic and applying
    LESSP.SIZE.SEQP, to:

          T.

  Case 2.1.
          (IMPLIES (AND (EQUAL (PLUS NEXT 0) 0)
                        (EQUAL (SIZE MESS) 0)
                        (NOT (EQUAL (PLUS NEXT (SUB1 0))
                                    (SUB1 (PLUS NEXT 0)))))
                   (NOT (SEQP MESS))).

    But this simplifies again, using linear arithmetic, to:

          (IMPLIES (AND (EQUAL 0 0)
                        (EQUAL (PLUS NEXT 0) 0)
                        (EQUAL (SIZE MESS) 0)
                        (NOT (EQUAL (PLUS NEXT (SUB1 0))
                                    (SUB1 0))))
                   (NOT (SEQP MESS))).

    But this simplifies again, using linear arithmetic and
    rewriting with LESSP.SIZE.SEQP, to:

          T.

Case 1. (IMPLIES (AND (EQUAL (PLUS NEXT (SIZE MESS)) 0)
                      (NOT (EQUAL (PLUS NEXT (SUB1 (SIZE MESS)))
                                  (SUB1 (PLUS NEXT (SIZE MESS))))))
                 (NOT (SEQP MESS))).

  This again simplifies, using linear arithmetic, to two new
  formulas:

  Case 1.2.
          (IMPLIES (AND (EQUAL (SIZE MESS) 0)
                        (EQUAL (PLUS NEXT (SIZE MESS)) 0)
                        (NOT (EQUAL (PLUS NEXT (SUB1 (SIZE MESS)))
                                    (SUB1 0)))))
                   (NOT (SEQP MESS))),

    which again simplifies, using linear arithmetic, to:

          (IMPLIES (AND (EQUAL 0 0)
                        (EQUAL (SIZE MESS) 0)
                        (EQUAL (PLUS NEXT 0) 0)
                        (NOT (EQUAL (PLUS NEXT (SUB1 0))
                                    (SUB1 0))))
                   (NOT (SEQP MESS))),

    which again simplifies, using linear arithmetic and applying
    the lemma LESSP.SIZE.SEQP, to:

          T.

  Case 1.1.
          (IMPLIES (AND (EQUAL 0 0)
                        (EQUAL (PLUS NEXT (SIZE MESS)) 0)
                        (NOT (EQUAL (PLUS NEXT (SUB1 (SIZE MESS)))
                                    (SUB1 0)))))
                   (NOT (SEQP MESS))),

    which again simplifies, using linear arithmetic and applying
    LESSP.SIZE.SEQP, to:

          T.

Q.E.D.

6313 conses
7.798 seconds
0.0 seconds, garbage collection time

----------------------------------------
```

```
+++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#3.10.3.3'

     (OR (NOT (SEQP (JOIN QUEUE (ENMAP MESS NEXT))))
         (EQUAL (DOM (LST (JOIN QUEUE (ENMAP MESS NEXT))))
                (SUB1 (PLUS NEXT (SIZE MESS))))
         (NOT (SEQP MESS)))

This formula can be simplified, using the abbreviations NOT and OR,
to:

      (IMPLIES
         (AND (SEQP (JOIN QUEUE (ENMAP MESS NEXT)))
              (NOT (EQUAL (DOM (LST (JOIN QUEUE (ENMAP MESS NEXT))))
                          (SUB1 (PLUS NEXT (SIZE MESS))))))
         (NOT (SEQP MESS))),

which simplifies, applying SEQP.JOIN, SEQP.ENMAP, LST.JOIN.2,
LST.ENMAP, and DOM.MPAIR, to the new conjecture:

      (IMPLIES (NOT (EQUAL (PLUS NEXT (SUB1 (SIZE MESS)))
                           (SUB1 (PLUS NEXT (SIZE MESS)))))
               (NOT (SEQP MESS))),

which again simplifies, using linear arithmetic, to two new
conjectures:

Case 2. (IMPLIES (AND (EQUAL (SIZE MESS) 0)
                      (NOT (EQUAL (PLUS NEXT (SUB1 (SIZE MESS)))
                                  (SUB1 (PLUS NEXT (SIZE MESS))))))
                 (NOT (SEQP MESS))),

  which we again simplify, using linear arithmetic, to two new
  formulas:

  Case 2.2.
        (IMPLIES (AND (EQUAL 0 0)
                      (EQUAL (SIZE MESS) 0)
                      (NOT (EQUAL (PLUS NEXT (SUB1 0))
                                   (SUB1 (PLUS NEXT 0)))))
                    (NOT (SEQP MESS))).
```

However this simplifies again, using linear arithmetic and applying LESSP.SIZE.SEQP, to:

```
      T.
```

Case 2.1.

```
      (IMPLIES (AND (EQUAL (PLUS NEXT 0) 0)
                    (EQUAL (SIZE MESS) 0)
                    (NOT (EQUAL (PLUS NEXT (SUB1 0))
                                (SUB1 (PLUS NEXT 0)))))
               (NOT (SEQP MESS))).
```

However this simplifies again, using linear arithmetic, to the conjecture:

```
      (IMPLIES (AND (EQUAL 0 0)
                    (EQUAL (PLUS NEXT 0) 0)
                    (EQUAL (SIZE MESS) 0)
                    (NOT (EQUAL (PLUS NEXT (SUB1 0))
                                (SUB1 0))))
               (NOT (SEQP MESS))).
```

This simplifies again, using linear arithmetic and rewriting with LESSP.SIZE.SEQP, to:

```
      T.
```

```
Case 1. (IMPLIES (AND (EQUAL (PLUS NEXT (SIZE MESS)) 0)
                      (NOT (EQUAL (PLUS NEXT (SUB1 (SIZE MESS)))
                                  (SUB1 (PLUS NEXT (SIZE MESS))))))
                 (NOT (SEQP MESS))),
```

which again simplifies, using linear arithmetic, to the following two new conjectures:

Case 1.2.

```
      (IMPLIES (AND (EQUAL (SIZE MESS) 0)
                    (EQUAL (PLUS NEXT (SIZE MESS)) 0)
                    (NOT (EQUAL (PLUS NEXT (SUB1 (SIZE MESS)))
                                (SUB1 0))))
               (NOT (SEQP MESS))).
```

```
    But this simplifies again, using linear arithmetic, to the
    conjecture:$

          (IMPLIES (AND (EQUAL 0 0)
                        (EQUAL (SIZE MESS) 0)
                        (EQUAL (PLUS NEXT 0) 0)
                        (NOT (EQUAL (PLUS NEXT (SUB1 0))
                                    (SUB1 0)))))
                   (NOT (SEQP MESS))),

    which again simplifies, using linear arithmetic and rewriting
    with LESSP.SIZE.SEQP, to:

          T.

   Case 1.1.
          (IMPLIES (AND (EQUAL 0 0)
                        (EQUAL (PLUS NEXT (SIZE MESS)) 0)
                        (NOT (EQUAL (PLUS NEXT (SUB1 (SIZE MESS)))
                                    (SUB1 0)))))
                   (NOT (SEQP MESS))),

    which again simplifies, using linear arithmetic and applying
    LESSP.SIZE.SEQP, to:

          T.

Q.E.D.

3905 conses
5.658 seconds
3.353 seconds, garbage collection time


------------------------------------



++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#5.0.1'
```

```
      (OR (EQUAL (FLATTEN (APR SOURCE MESS))
                 (FLATTEN SOURCE))
          (NOT (EQUAL MESS (NULL))))
```

This conjecture can be simplified, using the abbreviations NOT and OR, to the conjecture:

```
      (IMPLIES (NOT (EQUAL (FLATTEN (APR SOURCE MESS))
                           (FLATTEN SOURCE)))
               (NOT (EQUAL MESS (QUOTE (1QUOTE NULL))))),
```

which we simplify, applying LST.APR and NLST.APR, and expanding the definitions of JOIN, SEQP, and FLATTEN, to:

```
      T.
```

Q.E.D.

```
618 conses
1.062 seconds
0.0 seconds, garbage collection time
```

++

Proof of VC 'SENDER#8.1'

```
      (OR (PMAPP (UPPER QUEUE (ACKNO ACK)))
          (NOT (PMAPP QUEUE)))
```

This conjecture can be simplified, using the abbreviations NOT and OR, to:

```
      (IMPLIES (NOT (PMAPP (UPPER QUEUE (ACKNO ACK))))
               (NOT (PMAPP QUEUE))).
```

This simplifies, applying the lemma PMAPP.UPPER, to:

```
      T.

Q.E.D.

1498 conses
2.756 seconds
0.0 seconds, garbage collection time

----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#8.5'

     (OR (CONTIG (UPPER QUEUE (ACKNO ACK)))
         (NOT (CONTIG QUEUE))
         (NOT (PMAPP QUEUE)))

This conjecture can be simplified, using the abbreviations NOT, OR,
and CONTIG, to the conjecture:

      (IMPLIES
             (AND (NOT (CONSECP (DOMAIN (UPPER QUEUE (ACKNO ACK)))))
                  (CONSECP (DOMAIN QUEUE)))
             (NOT (PMAPP QUEUE))),

which we simplify, applying CONSECP.DOMAIN.UPPER, to:

      T.

Q.E.D.

2109 conses
2.614 seconds
0.0 seconds, garbage collection time

----------------------------------------
```

```
+++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#8.7'

     (OR (FOLLOWS (UPPER QUEUE (ACKNO ACK))
                  (FLATTEN PKT.OUT))
         (NOT (FOLLOWS QUEUE (FLATTEN PKT.OUT))))

This conjecture can be simplified, using the abbreviations NOT and
OR, to the new conjecture:

      (IMPLIES (NOT (FOLLOWS (UPPER QUEUE (ACKNO ACK))
                             (FLATTEN PKT.OUT)))
               (NOT (FOLLOWS QUEUE (FLATTEN PKT.OUT)))).

This simplifies, applying FOLLOWS.UPPER, to:

      T.

Q.E.D.

1275 conses
1.85 seconds
0.0 seconds, garbage collection time


----------------------------------------




+++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#8.10.1.2'

     (OR (EQUAL (UPPER QUEUE (ACKNO ACK))
                (NULL))
         (NOT (EQUAL QUEUE (NULL))))
```

```
This formula can be simplified, using the abbreviations NOT and OR,
to:

      (IMPLIES (NOT (EQUAL (UPPER QUEUE (ACKNO ACK))
                           (QUOTE (1QUOTE NULL))))
               (NOT (EQUAL QUEUE (QUOTE (1QUOTE NULL))))),

which simplifies, expanding the functions SEQP, UPPER, and EQUAL,
to:

      T.

Q.E.D.

625 conses
.922 seconds
0.0 seconds, garbage collection time

----------------------------------------



++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#8.10.4.3.1'

     (OR (EQUAL (DOM (LST (UPPER QUEUE (ACKNO ACK))))
                (SUB1 NEXT))
         (NOT (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT)))
         (NOT (SEQP (UPPER QUEUE (ACKNO ACK))))
         (NOT (PMAPP QUEUE)))

This formula can be simplified, using the abbreviations NOT and OR,
to:

      (IMPLIES
               (AND (NOT (EQUAL (DOM (LST (UPPER QUEUE (ACKNO ACK))))
                                (SUB1 NEXT)))
                    (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT))
                    (SEQP (UPPER QUEUE (ACKNO ACK))))
                (NOT (PMAPP QUEUE))),

which simplifies, applying LST.UPPER, to:

      T.

Q.E.D.

2784 conses
3.168 seconds
0.0 seconds, garbage collection time

----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#8.10.4.3.2'

     (OR (NOT (SEQP (UPPER QUEUE (ACKNO ACK))))
         (SEQP QUEUE))

This conjecture can be simplified, using the abbreviations NOT and
OR, to:

      (IMPLIES (SEQP (UPPER QUEUE (ACKNO ACK)))
               (SEQP QUEUE)).

This simplifies, opening up the definitions of UPPER and SEQP, to:

      T.

Q.E.D.

524 conses
.832 seconds
0.0 seconds, garbage collection time

----------------------------------------
```

++

Proof of VC 'SENDER#11.0.1'

```
     (OR (EQUAL (REDUCE (QUOTE WITH)
                        (NULL)
                        (FLATTEN (APR PKT.OUT QUEUE)))
                (REDUCE (QUOTE WITH)
                        (NULL)
                        (FLATTEN PKT.OUT)))
         (NOT (PMAPP QUEUE))
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.OUT))))
         (NOT (CONSISTENT (QUOTE DOM)
                          (QUOTE EQUAL)
                          (FLATTEN PKT.OUT)))
         (NOT (FOLLOWS QUEUE (FLATTEN PKT.OUT))))
```

This conjecture can be simplified, using the abbreviations NOT and OR, to:$

```
      (IMPLIES
            (AND (NOT (EQUAL (REDUCE (QUOTE WITH)
                                     (QUOTE (1QUOTE NULL))
                                     (FLATTEN (APR PKT.OUT QUEUE)))
                             (REDUCE (QUOTE WITH)
                                     (QUOTE (1QUOTE NULL))
                                     (FLATTEN PKT.OUT))))
                 (PMAPP QUEUE)
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.OUT)))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (FLATTEN PKT.OUT)))
            (NOT (FOLLOWS QUEUE (FLATTEN PKT.OUT)))).
```

This simplifies, applying the lemmas LST.APR, NLST.APR, REDUCE.JOIN, PMAPP.REDUCE.WITH, FOLLOWS.REDUCE.WITH.2, and REDUCE.WITH.FOLLOWS, and opening up FLATTEN and PMAPP, to:

```
      T.

Q.E.D.

8105 conses
8.809 seconds
3.387 seconds, garbage collection time


----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#11.3'

     (OR (ALL (FAPPLY (QUOTE MPAIRP)
                      (FLATTEN (APR PKT.OUT QUEUE))))
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.OUT))))
         (NOT (PMAPP QUEUE)))

This formula can be simplified, using the abbreviations NOT and OR,
to:

      (IMPLIES
             (AND (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                                    (FLATTEN (APR PKT.OUT QUEUE)))))
                  (ALL (FAPPLY (QUOTE MPAIRP)
                               (FLATTEN PKT.OUT))))
             (NOT (PMAPP QUEUE))),

which simplifies, applying LST.APR, NLST.APR, FAPPLY.JOIN, ALL.JOIN,
and ALL.FAPPLY.MPAIRP, and expanding the definition of FLATTEN, to:

      T.

Q.E.D.

1522 conses
```

```
2.134 seconds
0.0 seconds, garbage collection time

----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#11.4'

     (OR (ALL (FAPPLY (QUOTE CONTIG)
                      (APR PKT.OUT QUEUE)))
         (NOT (ALL (FAPPLY (QUOTE CONTIG) PKT.OUT)))
         (NOT (CONTIG QUEUE)))

This formula can be simplified, using the abbreviations NOT, OR,
and CONTIG, to the goal:

      (IMPLIES (AND (NOT (ALL (FAPPLY (QUOTE CONTIG)
                                      (APR PKT.OUT QUEUE))))
                    (ALL (FAPPLY (QUOTE CONTIG) PKT.OUT)))
               (NOT (CONSECP (DOMAIN QUEUE)))).

This simplifies, appealing to the lemmas APPLY1.CONTIG, LST.APR,
and NLST.APR, and expanding the definitions of CONTIG, FAPPLY, and
ALL, to:

      T.

Q.E.D.

998 conses
1.449 seconds
0.0 seconds, garbage collection time

----------------------------------------
```

```
++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#11.6'

     (OR (CONSISTENT (QUOTE DOM)
                     (QUOTE EQUAL)
                     (FLATTEN (APR PKT.OUT QUEUE)))
         (NOT (CONSISTENT (QUOTE DOM)
                          (QUOTE EQUAL)
                          (FLATTEN PKT.OUT)))
         (NOT (PMAPP QUEUE))
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.OUT))))
         (NOT (FOLLOWS QUEUE (FLATTEN PKT.OUT))))

This formula can be simplified, using the abbreviations NOT and OR,
to:$

      (IMPLIES (AND (NOT (CONSISTENT (QUOTE DOM)
                                     (QUOTE EQUAL)
                                     (FLATTEN (APR PKT.OUT QUEUE))))
                    (CONSISTENT (QUOTE DOM)
                                (QUOTE EQUAL)
                                (FLATTEN PKT.OUT))
                    (PMAPP QUEUE)
                    (ALL (FAPPLY (QUOTE MPAIRP)
                                 (FLATTEN PKT.OUT))))
               (NOT (FOLLOWS QUEUE (FLATTEN PKT.OUT)))),

which simplifies, applying LST.APR, NLST.APR, and
CONSISTENT.JOIN.FOLLOWS, and opening up the definition of FLATTEN,
to:

      T.

Q.E.D.

13672 conses
15.486 seconds
3.396 seconds, garbage collection time
```

```
----------------------------------------



++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#11.7'

     (FOLLOWS QUEUE
              (FLATTEN (APR PKT.OUT QUEUE)))

This formula simplifies, applying LST.APR, NLST.APR, and
FOLLOWS.JOIN.2, and opening up FLATTEN, to:

      T.

Q.E.D.

336 conses
.661 seconds
0.0 seconds, garbage collection time

----------------------------------------



++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#11.9.1'

     (OR (EQUAL (HIGHEST (DOMAIN (FLATTEN (APR PKT.OUT QUEUE))))
                (SUB1 NEXT))
         (NOT (EQUAL NEXT 0))
         (NOT (EQUAL (FLATTEN PKT.OUT) (NULL)))
         (NOT (EQUAL QUEUE (NULL))))

This formula can be simplified, using the abbreviations NOT and OR,
to:
```

```
(IMPLIES
 (AND
  (NOT (EQUAL (HIGHEST (DOMAIN (FLATTEN (APR PKT.OUT QUEUE))))
              (SUB1 NEXT)))
  (EQUAL NEXT 0)
  (EQUAL (FLATTEN PKT.OUT)
         (QUOTE (1QUOTE NULL))))
 (NOT (EQUAL QUEUE (QUOTE (1QUOTE NULL))))),
```

which simplifies, applying LST.APR and NLST.APR, and opening up the functions JOIN, SEQP, FLATTEN, DOMAIN, HIGHEST, SUB1, and EQUAL, to:

```
      T.
```

Q.E.D.

1154 conses
1.587 seconds
0.0 seconds, garbage collection time

++

Proof of VC 'SENDER#11.9.2'

```
(OR (EQUAL (HIGHEST (DOMAIN (FLATTEN (APR PKT.OUT QUEUE))))
           (SUB1 NEXT))
    (NOT (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                (SUB1 NEXT)))
    (NOT (PMAPP QUEUE))
    (NOT (NUMBERP NEXT))
    (EQUAL NEXT 0)
    (IF (SEQP QUEUE)
        (NOT (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT)))
        F))
```

$ This formula can be simplified, using the abbreviations NOT and OR, to:

```
      (IMPLIES
       (AND
        (NOT (EQUAL (HIGHEST (DOMAIN (FLATTEN (APR PKT.OUT QUEUE))))
                    (SUB1 NEXT)))
        (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
               (SUB1 NEXT))
        (PMAPP QUEUE)
        (NUMBERP NEXT)
        (NOT (EQUAL NEXT 0)))
       (IF (SEQP QUEUE)
           (NOT (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT)))
           F)).
```

This simplifies, applying LST.APR, NLST.APR, DOMAIN.JOIN, PISEQP.DOMAIN.2, HIGHEST.JOIN, and HIGHEST.DOMAIN, and expanding the functions FLATTEN, MAX, and NOT, to:

```
      (IMPLIES (AND (LESSP (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                           (DOM (LST QUEUE)))
                    (NOT (EQUAL (DOM (LST QUEUE)) (SUB1 NEXT)))
                    (EQUAL (HIGHEST (DOMAIN (FLATTEN PKT.OUT)))
                           (SUB1 NEXT))
                    (PMAPP QUEUE)
                    (NUMBERP NEXT)
                    (NOT (EQUAL NEXT 0)))
               (SEQP QUEUE)),
```

which again simplifies, applying LST.NSEQP, and opening up the functions DOM, EQUAL, and LESSP, to:

```
      T.
```

Q.E.D.

```
7238 conses
8.073 seconds
3.328 seconds, garbage collection time
```

```
++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#11.10.1.1'

     (OR (EQUAL (FLATTEN (APR PKT.OUT QUEUE))
                (NULL))
         (NOT (EQUAL (FLATTEN PKT.OUT) (NULL)))
         (NOT (EQUAL QUEUE (NULL))))

This formula can be simplified, using the abbreviations NOT and OR,
to the goal:

      (IMPLIES (AND (NOT (EQUAL (FLATTEN (APR PKT.OUT QUEUE))
                                (QUOTE (1QUOTE NULL))))
                    (EQUAL (FLATTEN PKT.OUT)
                           (QUOTE (1QUOTE NULL))))
               (NOT (EQUAL QUEUE (QUOTE (1QUOTE NULL))))).

This simplifies, appealing to the lemmas LST.APR and NLST.APR, and
expanding the definitions of JOIN, SEQP, and FLATTEN, to:

      T.

Q.E.D.

779 conses
1.231 seconds
0.0 seconds, garbage collection time

----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'SENDER#11.10.4.1'

     (OR (SEQP (FLATTEN (APR PKT.OUT QUEUE)))
         (NOT (SEQP (FLATTEN PKT.OUT))))
```

```
This conjecture can be simplified, using the abbreviations NOT and
OR, to:

      (IMPLIES (NOT (SEQP (FLATTEN (APR PKT.OUT QUEUE))))
               (NOT (SEQP (FLATTEN PKT.OUT)))),

which simplifies, rewriting with the lemmas LST.APR and NLST.APR,
and expanding the function FLATTEN, to:

      T.

Q.E.D.

455 conses
.868 seconds
0.0 seconds, garbage collection time

----------------------------------------


                                             27-Jun-82 08:50:51
VC Proof Log

++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#1'

     (RECEIVER.INT (NULL)
                   (NULL)
                   (NULL)
                   (NULL)
                   0 0
                   (NULL))

This formula can be simplified, using the abbreviation RECEIVER.INT,
to the following three new conjectures:

Case 3. (PMAPP (QUOTE (1QUOTE NULL))).

  This simplifies, opening up the definition of PMAPP, to:
        T.

Case 2. (NUMBERP 0),

  which simplifies, clearly, to:

        T.

Case 1. (IMPLIES
         (AND (CONSISTENT (QUOTE DOM)
                          (QUOTE EQUAL)
                          (FLATTEN (QUOTE (1QUOTE NULL))))
              (ALL (FAPPLY (QUOTE CONTIG)
                           (QUOTE (1QUOTE NULL)))))
         (IF
          (FOLLOWS (QUOTE (1QUOTE NULL))
                   (UPPER (REDUCE (QUOTE WITH)
                                  (QUOTE (1QUOTE NULL))
                                  (FLATTEN (QUOTE (1QUOTE NULL))))
                          1))
          (IF
           (LESSP (REACH (REDUCE (QUOTE WITH)
                                 (QUOTE (1QUOTE NULL))
                                 (FLATTEN (QUOTE (1QUOTE NULL)))))
                  0)
           F
           (IF
            (ALL (FAPPLY (QUOTE MPAIRP)
                         (FLATTEN (QUOTE (1QUOTE NULL)))))
            (IF
             (IN 0
                 (DOMAIN (FLATTEN (QUOTE (1QUOTE NULL)))))
             (IF
              (LESSP 0 0)
              (EQUAL
               (FLATTEN (QUOTE (1QUOTE NULL)))
               (RANGE
                     (LOWER (REDUCE (QUOTE WITH)
                                    (QUOTE (1QUOTE NULL))
                                    (FLATTEN (QUOTE (1QUOTE NULL))))
                            (SUB1 0))))
              F)
             (IF (EQUAL 0 0)
                 (EQUAL (QUOTE (1QUOTE NULL))
                        (QUOTE (1QUOTE NULL)))
                    F))
                F))
              F)),
```

which we simplify, applying CONSISTENT.DOM.EQUAL.PMAPP and REDUCE.WITH.NULL, and expanding the definitions of FLATTEN, PMAPP, SEQP, FAPPLY, ALL, UPPER, FOLLOWS, REACH, LESSP, DOMAIN, IN, and EQUAL, to:

```
      T.
```

Q.E.D.

3081 conses
11.023 seconds
0.0 seconds, garbage collection time

++

Proof of VC 'RECEIVER#2'

```
    (IMPLIES (RECEIVER.INT CREDITS PKT.IN SINK ACK.OUT NEXT EDGE
                           QUEUE)
             (RECEIVER.EXT CREDITS PKT.IN SINK ACK.OUT))
```

This conjecture can be simplified, using the abbreviations RECEIVER.EXT, RECEIVER.INT, and IMPLIES, to:$$

```
      (IMPLIES
       (AND
        (PMAPP QUEUE)
        (NUMBERP NEXT)
        (NUMBERP EDGE)
        (IF
         (CONSISTENT (QUOTE DOM)
                     (QUOTE EQUAL)
```

```
              (FLATTEN PKT.IN))
  (IF
   (ALL (FAPPLY (QUOTE CONTIG) PKT.IN))
   (IF
    (FOLLOWS QUEUE
             (UPPER (REDUCE (QUOTE WITH)
                            (QUOTE (1QUOTE NULL))
                            (FLATTEN PKT.IN))
                    (ADD1 NEXT)))
    (IF
     (LESSP (REACH (REDUCE (QUOTE WITH)
                           (QUOTE (1QUOTE NULL))
                           (FLATTEN PKT.IN)))
            NEXT)
     F
     (IF
      (ALL (FAPPLY (QUOTE MPAIRP)
                   (FLATTEN PKT.IN)))
      (IF
       (IN 0 (DOMAIN (FLATTEN PKT.IN)))
       (IF (LESSP 0 NEXT)
           (EQUAL (FLATTEN SINK)
                  (RANGE (LOWER (REDUCE (QUOTE WITH)
                                        (QUOTE (1QUOTE NULL))
                                        (FLATTEN PKT.IN))
                                (SUB1 NEXT))))
           F)
       (IF (EQUAL NEXT 0)
           (EQUAL SINK (QUOTE (1QUOTE NULL)))
           F))
      F))
    F)
   T)
  T)
 (CONSISTENT (QUOTE DOM)
             (QUOTE EQUAL)
             (FLATTEN PKT.IN))
 (ALL (FAPPLY (QUOTE CONTIG) PKT.IN)))
(IF (IN 0
        (DOMAIN (REDUCE (QUOTE WITH)
                        (QUOTE (1QUOTE NULL))
                        (FLATTEN PKT.IN))))
    (INITIAL (FLATTEN SINK)
             (RANGE (CONSEC (REDUCE (QUOTE WITH)
                                          (QUOTE (1QUOTE NULL))
                                          (FLATTEN PKT.IN)))))
          (EQUAL SINK (QUOTE (1QUOTE NULL))))),
```

which simplifies, rewriting with the lemma IN.DOMAIN.REDUCE.WITH, and expanding the functions EQUAL, LESSP, PMAPP, DOMAIN, and IN, to the new conjecture:$

```
     (IMPLIES
           (AND (PMAPP QUEUE)
                (NUMBERP NEXT)
                (NUMBERP EDGE)
                (FOLLOWS QUEUE
                         (UPPER (REDUCE (QUOTE WITH)
                                        (QUOTE (1QUOTE NULL))
                                        (FLATTEN PKT.IN))
                                (ADD1 NEXT)))
                (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                           (QUOTE (1QUOTE NULL))
                                           (FLATTEN PKT.IN)))
                            NEXT))
                (ALL (FAPPLY (QUOTE MPAIRP)
                             (FLATTEN PKT.IN)))
                (IN 0 (DOMAIN (FLATTEN PKT.IN)))
                (NOT (EQUAL NEXT 0))
                (EQUAL (FLATTEN SINK)
                       (RANGE (LOWER (REDUCE (QUOTE WITH)
                                             (QUOTE (1QUOTE NULL))
                                             (FLATTEN PKT.IN))
                                     (SUB1 NEXT))))
                (CONSISTENT (QUOTE DOM)
                            (QUOTE EQUAL)
                            (FLATTEN PKT.IN))
                (ALL (FAPPLY (QUOTE CONTIG) PKT.IN)))
           (INITIAL (FLATTEN SINK)
                    (RANGE (CONSEC (REDUCE (QUOTE WITH)
                                           (QUOTE (1QUOTE NULL))
                                           (FLATTEN PKT.IN)))))).
```

Appealing to the lemma SUB1.ELIM, we now replace NEXT by (ADD1 X) to eliminate (SUB1 NEXT). We rely upon the type restriction lemma noted when SUB1 was introduced to constrain the new variable. The result is the conjecture:$

```
     (IMPLIES
            (AND (NUMBERP X)
                 (PMAPP QUEUE)
                 (NUMBERP EDGE)
                 (FOLLOWS QUEUE
                          (UPPER (REDUCE (QUOTE WITH)
                                         (QUOTE (1QUOTE NULL))
                                         (FLATTEN PKT.IN))
                                 (ADD1 (ADD1 X))))
                 (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                            (QUOTE (1QUOTE NULL))
                                            (FLATTEN PKT.IN)))
                             (ADD1 X)))
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.IN)))
                 (IN 0 (DOMAIN (FLATTEN PKT.IN)))
                 (NOT (EQUAL (ADD1 X) 0))
                 (EQUAL (FLATTEN SINK)
                        (RANGE (LOWER (REDUCE (QUOTE WITH)
                                              (QUOTE (1QUOTE NULL))
                                              (FLATTEN PKT.IN))
                                      X)))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (FLATTEN PKT.IN))
                 (ALL (FAPPLY (QUOTE CONTIG) PKT.IN)))
            (INITIAL (FLATTEN SINK)
                     (RANGE (CONSEC (REDUCE (QUOTE WITH)
                                            (QUOTE (1QUOTE NULL))
                                            (FLATTEN PKT.IN)))))).
```

But this simplifies further, rewriting with SUB1.ADD1, SEQP.REDUCE.WITH, and EQUAL.REACH.ZERO, and unfolding the functions SEQP and LESSP, to the conjecture:$$

```
     (IMPLIES
        (AND (NUMBERP X)
             (PMAPP QUEUE)
             (NUMBERP EDGE)
             (FOLLOWS QUEUE
                      (UPPER (REDUCE (QUOTE WITH)
                                     (QUOTE (1QUOTE NULL))
                                     (FLATTEN PKT.IN))
                             (ADD1 (ADD1 X))))
```

```
            (SEQP (FLATTEN PKT.IN))
            (NOT (LESSP (SUB1 (REACH (REDUCE (QUOTE WITH)
                                             (QUOTE (1QUOTE NULL))
                                             (FLATTEN PKT.IN))))
                        X))
            (ALL (FAPPLY (QUOTE MPAIRP)
                         (FLATTEN PKT.IN)))
            (IN 0 (DOMAIN (FLATTEN PKT.IN)))
            (EQUAL (FLATTEN SINK)
                   (RANGE (LOWER (REDUCE (QUOTE WITH)
                                         (QUOTE (1QUOTE NULL))
                                         (FLATTEN PKT.IN))
                                 X)))
            (CONSISTENT (QUOTE DOM)
                        (QUOTE EQUAL)
                        (FLATTEN PKT.IN))
            (ALL (FAPPLY (QUOTE CONTIG) PKT.IN)))
       (INITIAL (FLATTEN SINK)
                (RANGE (CONSEC (REDUCE (QUOTE WITH)
                                       (QUOTE (1QUOTE NULL))
                                       (FLATTEN PKT.IN)))))).
```

We use the above equality hypothesis by substituting:

```
      (RANGE (LOWER (REDUCE (QUOTE WITH)
                            (QUOTE (1QUOTE NULL))
                            (FLATTEN PKT.IN))
                    X))
```

for (FLATTEN SINK) and keeping the equality hypothesis. We would thus like to prove the goal:$

```
      (IMPLIES
         (AND (NUMBERP X)
              (PMAPP QUEUE)
              (NUMBERP EDGE)
              (FOLLOWS QUEUE
                       (UPPER (REDUCE (QUOTE WITH)
                                      (QUOTE (1QUOTE NULL))
                                      (FLATTEN PKT.IN))
                              (ADD1 (ADD1 X))))
              (SEQP (FLATTEN PKT.IN))
              (NOT (LESSP (SUB1 (REACH (REDUCE (QUOTE WITH)
                                               (QUOTE (1QUOTE NULL))
                                               (FLATTEN PKT.IN))))
                          X))
```

```
          (ALL (FAPPLY (QUOTE MPAIRP)
                       (FLATTEN PKT.IN)))
          (IN 0 (DOMAIN (FLATTEN PKT.IN)))
          (EQUAL (FLATTEN SINK)
                 (RANGE (LOWER (REDUCE (QUOTE WITH)
                                       (QUOTE (1QUOTE NULL))
                                       (FLATTEN PKT.IN))
                               X)))
          (CONSISTENT (QUOTE DOM)
                      (QUOTE EQUAL)
                      (FLATTEN PKT.IN))
          (ALL (FAPPLY (QUOTE CONTIG) PKT.IN)))
     (INITIAL (RANGE (LOWER (REDUCE (QUOTE WITH)
                                    (QUOTE (1QUOTE NULL))
                                    (FLATTEN PKT.IN))
                            X))
              (RANGE (CONSEC (REDUCE (QUOTE WITH)
                                     (QUOTE (1QUOTE NULL))
                                     (FLATTEN PKT.IN)))))).
```

However this simplifies further, using linear arithmetic, rewriting with INITIAL.LOWER.CONSEC, PMAPP.REDUCE.WITH, and INITIAL.RANGE, and opening up the definition of PMAPP, to the conjecture:

```
      (IMPLIES
         (AND (NUMBERP X)
              (PMAPP QUEUE)
              (NUMBERP EDGE)
              (FOLLOWS QUEUE
                       (UPPER (REDUCE (QUOTE WITH)
                                      (QUOTE (1QUOTE NULL))
                                      (FLATTEN PKT.IN))
                              (ADD1 (ADD1 X))))
              (SEQP (FLATTEN PKT.IN))
              (NOT (LESSP (SUB1 (REACH (REDUCE (QUOTE WITH)
                                               (QUOTE (1QUOTE NULL))
                                               (FLATTEN PKT.IN))))
                          X))
              (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.IN)))
              (IN 0 (DOMAIN (FLATTEN PKT.IN)))
              (EQUAL (FLATTEN SINK)
                     (RANGE (LOWER (REDUCE (QUOTE WITH)
                                           (QUOTE (1QUOTE NULL))
```

```
                                        (FLATTEN PKT.IN))
                                X)))
                (CONSISTENT (QUOTE DOM)
                            (QUOTE EQUAL)
                            (FLATTEN PKT.IN))
                (ALL (FAPPLY (QUOTE CONTIG) PKT.IN))
                (EQUAL (REACH (REDUCE (QUOTE WITH)
                                      (QUOTE (1QUOTE NULL))
                                      (FLATTEN PKT.IN)))
                       0))
           (INITIAL (RANGE (LOWER (REDUCE (QUOTE WITH)
                                          (QUOTE (1QUOTE NULL))
                                          (FLATTEN PKT.IN))
                                  X))
                    (RANGE (CONSEC (REDUCE (QUOTE WITH)
                                           (QUOTE (1QUOTE NULL))
                                           (FLATTEN PKT.IN)))))),
```

which again simplifies, using linear arithmetic, to:$

```
      (IMPLIES
            (AND (EQUAL 0 0)
                 (NUMBERP X)
                 (PMAPP QUEUE)
                 (NUMBERP EDGE)
                 (FOLLOWS QUEUE
                          (UPPER (REDUCE (QUOTE WITH)
                                         (QUOTE (1QUOTE NULL))
                                         (FLATTEN PKT.IN))
                                 (ADD1 (ADD1 X))))
                 (SEQP (FLATTEN PKT.IN))
                 (NOT (LESSP (SUB1 0) X))
                 (ALL (FAPPLY (QUOTE MPAIRP)
                              (FLATTEN PKT.IN)))
                 (IN 0 (DOMAIN (FLATTEN PKT.IN)))
                 (EQUAL (FLATTEN SINK)
                        (RANGE (LOWER (REDUCE (QUOTE WITH)
                                              (QUOTE (1QUOTE NULL))
                                              (FLATTEN PKT.IN))
                                      X)))
                 (CONSISTENT (QUOTE DOM)
                             (QUOTE EQUAL)
                             (FLATTEN PKT.IN))
                 (ALL (FAPPLY (QUOTE CONTIG) PKT.IN))
```

```
                    (EQUAL (REACH (REDUCE (QUOTE WITH)
                                          (QUOTE (1QUOTE NULL))
                                          (FLATTEN PKT.IN)))
                           0))
               (INITIAL (RANGE (LOWER (REDUCE (QUOTE WITH)
                                              (QUOTE (1QUOTE NULL))
                                              (FLATTEN PKT.IN))
                                      X))
                        (RANGE (CONSEC (REDUCE (QUOTE WITH)
                                               (QUOTE (1QUOTE NULL))
                                               (FLATTEN PKT.IN)))))),

which finally simplifies, applying EQUAL.REACH.ZERO and
SEQP.REDUCE.WITH, and unfolding EQUAL, SUB1, LESSP, and SEQP, to:

      T.

Q.E.D.

237092 conses
285.842 seconds
27.909 seconds, garbage collection time

----------------------------------------


VC Proof Log                                          27-Jun-82 08:59:15

++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#5.0.2.1'

     (OR (NOT (CONSISTENT (QUOTE DOM)
                          (QUOTE EQUAL)
                          (JOIN (FLATTEN PKT.IN) PKT)))
         (CONSISTENT (QUOTE DOM)
                     (QUOTE EQUAL)
                     (FLATTEN PKT.IN)))

$  This formula can be simplified, using the abbreviations NOT and
OR, to:
      (IMPLIES (CONSISTENT (QUOTE DOM)
                           (QUOTE EQUAL)
                           (JOIN (FLATTEN PKT.IN) PKT))
               (CONSISTENT (QUOTE DOM)
                           (QUOTE EQUAL)
                           (FLATTEN PKT.IN))),

which simplifies, applying CONSISTENT.JOIN.NOT, to:

      T.

Q.E.D.

570 conses
1.417 seconds
3.344 seconds, garbage collection time

----------------------------------------



++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#5.0.2.2'

     (OR (NOT (ALL (FAPPLY (QUOTE CONTIG)
                           (APR PKT.IN PKT))))
         (ALL (FAPPLY (QUOTE CONTIG) PKT.IN)))

This formula can be simplified, using the abbreviations NOT and OR,
to:

      (IMPLIES (ALL (FAPPLY (QUOTE CONTIG)
                            (APR PKT.IN PKT)))
               (ALL (FAPPLY (QUOTE CONTIG) PKT.IN))).

This simplifies, applying APPLY1.CONTIG, LST.APR, and NLST.APR, and
expanding the functions CONTIG, FAPPLY, and ALL, to:

      T.
```

```
Q.E.D.

907 conses
1.49 seconds
0.0 seconds, garbage collection time

----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#5.0.2.3'

     (OR (EQUAL (REDUCE (QUOTE WITH)
                        (NULL)
                        (JOIN (FLATTEN PKT.IN) PKT))
                (REDUCE (QUOTE WITH)
                        (NULL)
                        (FLATTEN PKT.IN)))
         (NOT (CONSISTENT (QUOTE DOM)
                          (QUOTE EQUAL)
                          (JOIN (FLATTEN PKT.IN) PKT)))
         (NOT (NUMBERP NEXT))
         (LESSP (REACH (REDUCE (QUOTE WITH)
                               (NULL)
                               (FLATTEN PKT.IN)))
                NEXT)
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.IN))))
         (NOT (IN 0 (DOMAIN (FLATTEN PKT.IN))))
         (NOT (LESSP 0 NEXT))
         (NOT (LESSP (DOM (LST PKT)) NEXT))
         (NOT (PMAPP PKT)))

This formula can be simplified, using the abbreviations NOT and OR,
to:

      (IMPLIES (AND (NOT (EQUAL (REDUCE (QUOTE WITH)
                                        (QUOTE (1QUOTE NULL))
                                        (JOIN (FLATTEN PKT.IN) PKT))
                              (REDUCE (QUOTE WITH)
                                      (QUOTE (1QUOTE NULL))
                                      (FLATTEN PKT.IN))))
                  (CONSISTENT (QUOTE DOM)
                              (QUOTE EQUAL)
                              (JOIN (FLATTEN PKT.IN) PKT))
                  (NUMBERP NEXT)
                  (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                             (QUOTE (1QUOTE NULL))
                                             (FLATTEN PKT.IN)))
                              NEXT))
                  (ALL (FAPPLY (QUOTE MPAIRP)
                               (FLATTEN PKT.IN)))
                  (IN 0 (DOMAIN (FLATTEN PKT.IN)))
                  (LESSP 0 NEXT)
                  (LESSP (DOM (LST PKT)) NEXT))
             (NOT (PMAPP PKT))),

which we simplify, using linear arithmetic, applying REDUCE.JOIN,
PMAPP.REDUCE.WITH, FOLLOWS.REDUCE.WITH.CONSISTENT,
IN.DOMAIN.REDUCE.WITH, ALL.FAPPLY.MPAIRP, HIGHEST.DOMAIN, and
REDUCE.WITH.FOLLOWS, and opening up PMAPP, to:

      T.

Q.E.D.

22805 conses
26.299 seconds
0.0 seconds, garbage collection time
[561 cns / 1.4 s + 0.0 gc + 0.0 io (= 2  1)]

----------------------------------------



++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#5.4.2.3'
```

```
      (OR (ALL (FAPPLY (QUOTE MPAIRP)
                       (JOIN (FLATTEN PKT.IN) PKT)))
          (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                            (FLATTEN PKT.IN))))
          (NOT (PMAPP PKT)))
```

This formula can be simplified, using the abbreviations NOT and OR, to:$

```
       (IMPLIES (AND (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                                       (JOIN (FLATTEN PKT.IN) PKT))))
                     (ALL (FAPPLY (QUOTE MPAIRP)
                                  (FLATTEN PKT.IN))))
                (NOT (PMAPP PKT))),
```

which simplifies, applying FAPPLY.JOIN, ALL.JOIN, and ALL.FAPPLY.MPAIRP, to:

```
       T.
```

Q.E.D.

1560 conses
2.353 seconds
3.394 seconds, garbage collection time

++

Proof of VC 'RECEIVER#5.4.2.4.3'

```
      (OR (IN 0
              (DOMAIN (JOIN (FLATTEN PKT.IN) PKT)))
          (NOT (IN 0 (DOMAIN (FLATTEN PKT.IN)))))
```

This formula can be simplified, using the abbreviations NOT, IN.JOIN, OR, and DOMAIN.JOIN, to:

```
      T.

This simplifies, clearly, to:

      T.

Q.E.D.

405 conses
.902 seconds
0.0 seconds, garbage collection time

----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#6.1'

     (OR (PMAPP (REDUCE (QUOTE WITH)
                        QUEUE
                        (LOWER PKT EDGE)))
         (NOT (PMAPP QUEUE)))

This formula can be simplified, using the abbreviations NOT and OR,
to:

      (IMPLIES (NOT (PMAPP (REDUCE (QUOTE WITH)
                                   QUEUE
                                   (LOWER PKT EDGE))))
               (NOT (PMAPP QUEUE))),

which simplifies, applying PMAPP.REDUCE.WITH, to:

      T.

Q.E.D.

5204 conses
5.092 seconds
```

```
0.0 seconds, garbage collection time

----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#6.4.2.1'

     (OR (FOLLOWS (REDUCE (QUOTE WITH)
                          QUEUE
                          (LOWER PKT EDGE))
                  (UPPER (REDUCE (QUOTE WITH)
                                 (NULL)
                                 (JOIN (FLATTEN PKT.IN) PKT))
                         (ADD1 NEXT)))
         (NOT (FOLLOWS QUEUE
                       (UPPER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (FLATTEN PKT.IN))
                              (ADD1 NEXT))))
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.IN))))
         (NOT (CONSISTENT (QUOTE DOM)
                          (QUOTE EQUAL)
                          (JOIN (FLATTEN PKT.IN) PKT)))
         (NOT (PMAPP QUEUE))
         (NOT (NUMBERP NEXT))
         (NOT (LESSP NEXT (DOM (FIRST PKT))))
         (NOT (PMAPP PKT)))

This conjecture can be simplified, using the abbreviations NOT and
OR, to:$

      (IMPLIES
       (AND
           (NOT (FOLLOWS (REDUCE (QUOTE WITH)
                                 QUEUE
                                 (LOWER PKT EDGE))
                         (UPPER (REDUCE (QUOTE WITH)
                                                    (QUOTE (1QUOTE NULL))
                                                    (JOIN (FLATTEN PKT.IN) PKT))
                                          (ADD1 NEXT))))
                    (FOLLOWS QUEUE
                             (UPPER (REDUCE (QUOTE WITH)
                                            (QUOTE (1QUOTE NULL))
                                            (FLATTEN PKT.IN))
                                    (ADD1 NEXT)))
                    (ALL (FAPPLY (QUOTE MPAIRP)
                                 (FLATTEN PKT.IN)))
                    (CONSISTENT (QUOTE DOM)
                                (QUOTE EQUAL)
                                (JOIN (FLATTEN PKT.IN) PKT))
                    (PMAPP QUEUE)
                    (NUMBERP NEXT)
                    (LESSP NEXT (DOM (FIRST PKT))))
                (NOT (PMAPP PKT))).

This simplifies, using linear arithmetic, applying the lemmas
REDUCE.JOIN, PMAPP.REDUCE.WITH, FOLLOWS.LOWER, FOLLOWS.SAME,
CONSISTENT.DOM.EQUAL.JOIN.REDUCE, FOLLOWS.REDUCE.WITH.SAME, and
FOLLOWS.REDUCE.WITH.UPPER, and opening up PMAPP, to:

      T.

Q.E.D.

21467 conses
23.242 seconds
3.444 seconds, garbage collection time
[465 cns / 1.0 s + 0.0 gc + 0.0 io (= 2  1)]


----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#6.4.2.2'

     (OR (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                           (NULL)
                                           (JOIN (FLATTEN PKT.IN) PKT)))
                            NEXT))
                (LESSP (REACH (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (FLATTEN PKT.IN)))
                       NEXT)
                (NOT (NUMBERP NEXT))
                (NOT (LESSP NEXT (DOM (FIRST PKT))))
                (NOT (PMAPP PKT)))
```

This conjecture can be simplified, using the abbreviations NOT and OR, to:

```
      (IMPLIES
            (AND (LESSP (REACH (REDUCE (QUOTE WITH)
                                       (QUOTE (1QUOTE NULL))
                                       (JOIN (FLATTEN PKT.IN) PKT)))
                            NEXT)
                 (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                            (QUOTE (1QUOTE NULL))
                                            (FLATTEN PKT.IN)))
                             NEXT))
                 (NUMBERP NEXT)
                 (LESSP NEXT (DOM (FIRST PKT))))
            (NOT (PMAPP PKT))),
```

which simplifies, rewriting with the lemmas REDUCE.JOIN, PMAPP.REDUCE.WITH, and LESSP.REACH.REDUCE.WITH, and expanding the function PMAPP, to:

```
      T.
```

Q.E.D.

```
11351 conses
11.246 seconds
0.0 seconds, garbage collection time
```

```
++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#6.4.2.4.1.2'

     (OR (EQUAL (RANGE (LOWER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (FLATTEN PKT.IN))
                              (SUB1 NEXT)))
                (RANGE (LOWER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (JOIN (FLATTEN PKT.IN) PKT))
                              (SUB1 NEXT))))
         (NOT (LESSP 0 NEXT))
         (NOT (NUMBERP NEXT))
         (NOT (LESSP NEXT (DOM (FIRST PKT))))
         (NOT (PMAPP PKT)))

This conjecture can be simplified, using the abbreviations NOT and
OR, to:$

      (IMPLIES
       (AND
        (NOT
           (EQUAL (RANGE (LOWER (REDUCE (QUOTE WITH)
                                        (QUOTE (1QUOTE NULL))
                                        (FLATTEN PKT.IN))
                                (SUB1 NEXT)))
                  (RANGE (LOWER (REDUCE (QUOTE WITH)
                                        (QUOTE (1QUOTE NULL))
                                        (JOIN (FLATTEN PKT.IN) PKT))
                                (SUB1 NEXT)))))
        (LESSP 0 NEXT)
        (NUMBERP NEXT)
        (LESSP NEXT (DOM (FIRST PKT))))
       (NOT (PMAPP PKT))),

which simplifies, using linear arithmetic, rewriting with the
lemmas REDUCE.JOIN, PMAPP.REDUCE.WITH, and LOWER.REDUCE.WITH.LESSP,
and expanding the functions PMAPP and LESSP, to:

      T.
```

```
Q.E.D.

17377 conses
17.309 seconds
3.527 seconds, garbage collection time


-----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#6.4.2.4.2'

     (OR (NOT (IN 0
                  (DOMAIN (JOIN (FLATTEN PKT.IN) PKT))))
         (NOT (EQUAL NEXT 0))
         (NOT (LESSP NEXT (DOM (FIRST PKT))))
         (NOT (PMAPP PKT))
         (IN 0 (DOMAIN (FLATTEN PKT.IN))))

This formula can be simplified, using the abbreviations NOT, OR,
and DOMAIN.JOIN, to:

      (IMPLIES (AND (IN 0
                        (JOIN (DOMAIN (FLATTEN PKT.IN))
                              (DOMAIN PKT)))
                    (EQUAL NEXT 0)
                    (LESSP NEXT (DOM (FIRST PKT)))
                    (PMAPP PKT))
               (IN 0 (DOMAIN (FLATTEN PKT.IN)))),

which simplifies, applying IN.JOIN and IN.DOMAIN.LESSP.FIRST.2, to:

      T.

Q.E.D.

6875 conses
8.182 seconds
0.0 seconds, garbage collection time
```

```
----------------------------------------



++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#7.1'

     (OR (PMAPP (UPPER QUEUE (ADD1 (REACH M))))
         (NOT (PMAPP QUEUE)))

This conjecture can be simplified, using the abbreviations NOT and
OR, to:

      (IMPLIES (NOT (PMAPP (UPPER QUEUE (ADD1 (REACH M)))))
               (NOT (PMAPP QUEUE))),

which simplifies, rewriting with the lemma PMAPP.UPPER, to:

      T.

Q.E.D.

1551 conses
1.821 seconds
0.0 seconds, garbage collection time

----------------------------------------



++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#7.4.2.1'

     (OR (FOLLOWS (UPPER QUEUE (ADD1 (REACH M)))
                  (UPPER (REDUCE (QUOTE WITH)
                                        (NULL)
                                        (JOIN (FLATTEN PKT.IN) PKT))
                                (ADD1 (REACH M))))
        (NOT (FOLLOWS QUEUE
                              (UPPER (REDUCE (QUOTE WITH)
                                             (NULL)
                                             (FLATTEN PKT.IN))
                                     (ADD1 NEXT))))
        (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                          (FLATTEN PKT.IN))))
        (NOT (CONSISTENT (QUOTE DOM)
                         (QUOTE EQUAL)
                         (JOIN (FLATTEN PKT.IN) PKT)))
        (NOT (PMAPP QUEUE))
        (NOT (PMAPP PKT)))
```

This conjecture can be simplified, using the abbreviations NOT and OR, to the conjecture:$

```
      (IMPLIES
       (AND
          (NOT (FOLLOWS (UPPER QUEUE (ADD1 (REACH M)))
                        (UPPER (REDUCE (QUOTE WITH)
                                       (QUOTE (1QUOTE NULL))
                                       (JOIN (FLATTEN PKT.IN) PKT))
                               (ADD1 (REACH M)))))
          (FOLLOWS QUEUE
                   (UPPER (REDUCE (QUOTE WITH)
                                  (QUOTE (1QUOTE NULL))
                                  (FLATTEN PKT.IN))
                          (ADD1 NEXT)))
          (ALL (FAPPLY (QUOTE MPAIRP)
                       (FLATTEN PKT.IN)))
          (CONSISTENT (QUOTE DOM)
                      (QUOTE EQUAL)
                      (JOIN (FLATTEN PKT.IN) PKT))
          (PMAPP QUEUE))
       (NOT (PMAPP PKT))),
```

which we simplify, applying REDUCE.JOIN, FOLLOWS.REDUCE.WITH.3, FOLLOWS.UPPER, FOLLOWS.REDUCE.REDUCE.WITH, FOLLOWS.TRANS, FOLLOWS.REDUCE.WITH.SAME, CONSISTENT.DOM.EQUAL.JOIN.REDUCE, PMAPP.REDUCE.WITH, ALL.FAPPLY.MPAIRP, and FOLLOWS.UPPER.UPPER, and expanding the definitions of PMAPP and FOLLOWS, to:

```
      T.

Q.E.D.

28484 conses
31.805 seconds
3.459 seconds, garbage collection time
[330 cns / .7 s + 0.0 gc + 0.0 io (= 1  1)]


----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#7.4.2.2.1'

     (OR (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                    (NULL)
                                    (JOIN (FLATTEN PKT.IN) PKT)))
                     (REACH M)))
         (LESSP (REACH (REDUCE (QUOTE WITH)
                               (NULL)
                               (FLATTEN PKT.IN)))
                NEXT)
         (NOT (FOLLOWS QUEUE
                       (UPPER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (FLATTEN PKT.IN))
                              (ADD1 NEXT))))
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.IN))))
         (NOT (IN 0 (DOMAIN (FLATTEN PKT.IN))))
         (NOT (LESSP 0 NEXT))
         (NOT (CONSISTENT (QUOTE DOM)
                          (QUOTE EQUAL)
                          (JOIN (FLATTEN PKT.IN) PKT)))
         (NOT (ALL (FAPPLY (QUOTE CONTIG)
                           (APR PKT.IN PKT))))
         (NOT (PMAPP QUEUE))
```

```
        (NOT (NUMBERP NEXT))
        (NOT (NUMBERP EDGE))
        (NOT (EQUAL M
                    (REDUCE (QUOTE WITH)
                            QUEUE
                            (UPPER (LOWER PKT EDGE) NEXT))))
        (LESSP (DOM (LST PKT)) NEXT)
        (LESSP NEXT (DOM (FIRST PKT)))
        (NOT (SEQP PKT))
        (LESSP EDGE NEXT)
        (NOT (PMAPP PKT)))
```

This formula can be simplified, using the abbreviations NOT and OR, to:$$$$

```
      (IMPLIES
           (AND (LESSP (REACH (REDUCE (QUOTE WITH)
                                      (QUOTE (1QUOTE NULL))
                                      (JOIN (FLATTEN PKT.IN) PKT)))
                       (REACH M))
                (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                           (QUOTE (1QUOTE NULL))
                                           (FLATTEN PKT.IN)))
                            NEXT))
                (FOLLOWS QUEUE
                         (UPPER (REDUCE (QUOTE WITH)
                                        (QUOTE (1QUOTE NULL))
                                        (FLATTEN PKT.IN))
                                (ADD1 NEXT)))
                (ALL (FAPPLY (QUOTE MPAIRP)
                             (FLATTEN PKT.IN)))
                (IN 0 (DOMAIN (FLATTEN PKT.IN)))
                (LESSP 0 NEXT)
                (CONSISTENT (QUOTE DOM)
                            (QUOTE EQUAL)
                            (JOIN (FLATTEN PKT.IN) PKT))
                (ALL (FAPPLY (QUOTE CONTIG)
                             (APR PKT.IN PKT)))
                (PMAPP QUEUE)
                (NUMBERP NEXT)
                (NUMBERP EDGE)
                (EQUAL M
                       (REDUCE (QUOTE WITH)
                               QUEUE
```

```
                              (UPPER (LOWER PKT EDGE) NEXT)))
                 (NOT (LESSP (DOM (LST PKT)) NEXT))
                 (NOT (LESSP NEXT (DOM (FIRST PKT))))
                 (SEQP PKT)
                 (NOT (LESSP EDGE NEXT)))
            (NOT (PMAPP PKT))),
```

which simplifies, applying REDUCE.JOIN, APPLY1.CONTIG, LST.APR, and NLST.APR, and opening up the functions EQUAL, LESSP, CONTIG, FAPPLY, and ALL, to:$$

```
      (IMPLIES
        (AND (LESSP (REACH (REDUCE (QUOTE WITH)
                                   (REDUCE (QUOTE WITH)
                                           (QUOTE (1QUOTE NULL))
                                           (FLATTEN PKT.IN))
                                   PKT))
                    (REACH (REDUCE (QUOTE WITH)
                                   QUEUE
                                   (UPPER (LOWER PKT EDGE) NEXT))))
             (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                        (QUOTE (1QUOTE NULL))
                                        (FLATTEN PKT.IN)))
                         NEXT))
             (FOLLOWS QUEUE
                      (UPPER (REDUCE (QUOTE WITH)
                                     (QUOTE (1QUOTE NULL))
                                     (FLATTEN PKT.IN))
                             (ADD1 NEXT)))
             (ALL (FAPPLY (QUOTE MPAIRP)
                          (FLATTEN PKT.IN)))
             (IN 0 (DOMAIN (FLATTEN PKT.IN)))
             (NOT (EQUAL NEXT 0))
             (CONSISTENT (QUOTE DOM)
                         (QUOTE EQUAL)
                         (JOIN (FLATTEN PKT.IN) PKT))
             (CONSECP (DOMAIN PKT))
             (ALL (FAPPLY (QUOTE CONTIG) PKT.IN))
             (PMAPP QUEUE)
             (NUMBERP NEXT)
             (NUMBERP EDGE)
             (NOT (LESSP (DOM (LST PKT)) NEXT))
             (NOT (LESSP NEXT (DOM (FIRST PKT))))
             (SEQP PKT)
```

```
              (NOT (LESSP EDGE NEXT)))
         (NOT (PMAPP PKT))),

which again simplifies, using linear arithmetic, applying the
lemmas IN.DOMAIN.REDUCE.WITH, LESSP.REACH.REDUCE.WITH.2,
DOM.FIRST.REDUCE.UPPER.LOWER, FOLLOWS.REDUCE.CONSISTENT.JOIN,
FOLLOWS.REDUCE.WITH.SAME, CONSISTENT.DOM.EQUAL.JOIN.REDUCE,
FOLLOWS.UPPER, FOLLOWS.REDUCE.REDUCE.WITH, FOLLOWS.TRANS,
FOLLOWS.LOWER, FOLLOWS.SAME, PMAPP.REDUCE.WITH, ALL.FAPPLY.MPAIRP,
and LESSP.REACH.REACH.2, and expanding the functions PMAPP and
FOLLOWS, to:

      T.

Q.E.D.

156478 conses
188.374 seconds
21.06 seconds, garbage collection time
[579 cns / 1.2 s + 0.0 gc + 0.0 io (= 1  1)]
[429 cns / 1.0 s + 0.0 gc + 0.0 io (= 1  1)]


----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#7.4.2.2.2'

     (OR (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                    (NULL)
                                    (JOIN (FLATTEN PKT.IN) PKT)))
                     (REACH M)))
         (NOT (FOLLOWS QUEUE
                       (UPPER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (FLATTEN PKT.IN))
                              (ADD1 NEXT))))
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.IN)))))
```

```
          (IN 0 (DOMAIN (FLATTEN PKT.IN)))
          (NOT (EQUAL NEXT 0))
          (NOT (CONSISTENT (QUOTE DOM)
                           (QUOTE EQUAL)
                           (JOIN (FLATTEN PKT.IN) PKT)))
          (NOT (ALL (FAPPLY (QUOTE CONTIG)
                            (APR PKT.IN PKT))))
          (NOT (PMAPP QUEUE))
          (NOT (NUMBERP EDGE))
          (NOT (EQUAL M
                      (REDUCE (QUOTE WITH)
                              QUEUE
                              (UPPER (LOWER PKT EDGE) NEXT))))
          (LESSP NEXT (DOM (FIRST PKT)))
          (NOT (SEQP PKT))
          (NOT (PMAPP PKT)))
```

This conjecture can be simplified, using the abbreviations NOT and OR, to:$$$

```
      (IMPLIES
           (AND (LESSP (REACH (REDUCE (QUOTE WITH)
                                      (QUOTE (1QUOTE NULL))
                                      (JOIN (FLATTEN PKT.IN) PKT)))
                       (REACH M))
                (FOLLOWS QUEUE
                         (UPPER (REDUCE (QUOTE WITH)
                                        (QUOTE (1QUOTE NULL))
                                        (FLATTEN PKT.IN))
                                (ADD1 NEXT)))
                (ALL (FAPPLY (QUOTE MPAIRP)
                             (FLATTEN PKT.IN)))
                (NOT (IN 0 (DOMAIN (FLATTEN PKT.IN))))
                (EQUAL NEXT 0)
                (CONSISTENT (QUOTE DOM)
                            (QUOTE EQUAL)
                            (JOIN (FLATTEN PKT.IN) PKT))
                (ALL (FAPPLY (QUOTE CONTIG)
                             (APR PKT.IN PKT)))
                (PMAPP QUEUE)
                (NUMBERP EDGE)
                (EQUAL M
                       (REDUCE (QUOTE WITH)
                               QUEUE
```

```
                              (UPPER (LOWER PKT EDGE) NEXT)))
               (NOT (LESSP NEXT (DOM (FIRST PKT))))
               (SEQP PKT))
          (NOT (PMAPP PKT))).
```

This simplifies, using linear arithmetic, applying the lemmas REDUCE.JOIN, PMAPP.LOWER, SEQP.LOWER, FIRST.LOWER, UPPER.NOT.LESSP, APPLY1.CONTIG, LST.APR, and NLST.APR, and opening up CONTIG, FAPPLY, ALL, EQUAL, and LESSP, to:$

```
     (IMPLIES
          (AND (LESSP (REACH (REDUCE (QUOTE WITH)
                                     (REDUCE (QUOTE WITH)
                                             (QUOTE (1QUOTE NULL))
                                             (FLATTEN PKT.IN))
                                     PKT))
                      (REACH (REDUCE (QUOTE WITH)
                                     QUEUE
                                     (LOWER PKT EDGE))))
               (FOLLOWS QUEUE
                        (UPPER (REDUCE (QUOTE WITH)
                                       (QUOTE (1QUOTE NULL))
                                       (FLATTEN PKT.IN))
                               1))
               (ALL (FAPPLY (QUOTE MPAIRP)
                            (FLATTEN PKT.IN)))
               (NOT (IN 0 (DOMAIN (FLATTEN PKT.IN))))
               (CONSISTENT (QUOTE DOM)
                           (QUOTE EQUAL)
                           (JOIN (FLATTEN PKT.IN) PKT))
               (CONSECP (DOMAIN PKT))
               (ALL (FAPPLY (QUOTE CONTIG) PKT.IN))
               (PMAPP QUEUE)
               (NUMBERP EDGE)
               (EQUAL (DOM (FIRST PKT)) 0)
               (SEQP PKT))
          (NOT (PMAPP PKT))),
```

which we again simplify, using linear arithmetic, applying PMAPP.REDUCE.WITH, DOM.FIRST.REDUCE.WITH.ZERO, FIRST.LOWER, SEQP.LOWER, PMAPP.LOWER, FOLLOWS.REDUCE.WITH.LOWER.REDUCE, and LESSP.REACH.REACH.2, and expanding the definitions of PMAPP and LESSP, to:

```
     T.

Q.E.D.

86244 conses
100.039 seconds
13.959 seconds, garbage collection time
[495 cns / 1.2 s + 0.0 gc + 0.0 io (= 1  1)]
[271 cns / .6 s + 0.0 gc + 0.0 io (= 1  1)]


----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#7.4.2.4.1.1'

     (OR (LESSP 0 (REACH M))
         (NOT (ALL (FAPPLY (QUOTE CONTIG)
                           (APR PKT.IN PKT))))
         (NOT (PMAPP QUEUE))
         (NOT (NUMBERP NEXT))
         (NOT (NUMBERP EDGE))
         (NOT (EQUAL M
                     (REDUCE (QUOTE WITH)
                             QUEUE
                             (UPPER (LOWER PKT EDGE) NEXT))))
         (LESSP (DOM (LST PKT)) NEXT)
         (LESSP NEXT (DOM (FIRST PKT)))
         (NOT (SEQP PKT))
         (LESSP EDGE NEXT)
         (NOT (PMAPP PKT)))

This formula can be simplified, using the abbreviations NOT and OR,
to:$

      (IMPLIES (AND (NOT (LESSP 0 (REACH M)))
                    (ALL (FAPPLY (QUOTE CONTIG)
                                 (APR PKT.IN PKT)))
                    (PMAPP QUEUE)
                    (NUMBERP NEXT)
                    (NUMBERP EDGE)
                    (EQUAL M
                           (REDUCE (QUOTE WITH)
                                   QUEUE
                                   (UPPER (LOWER PKT EDGE) NEXT)))
                    (NOT (LESSP (DOM (LST PKT)) NEXT))
                    (NOT (LESSP NEXT (DOM (FIRST PKT))))
                    (SEQP PKT)
                    (NOT (LESSP EDGE NEXT)))
               (NOT (PMAPP PKT))),
```

which simplifies, applying PMAPP.REDUCE.WITH, LESSP.ZERO.REACH, SEQP.REDUCE.WITH, APPLY1.CONTIG, LST.APR, and NLST.APR, and opening up the definitions of CONTIG, FAPPLY, ALL, and PMAPP, to the conjecture:

```
      (IMPLIES (AND (NOT (SEQP QUEUE))
                    (NOT (SEQP (UPPER (LOWER PKT EDGE) NEXT)))
                    (CONSECP (DOMAIN PKT))
                    (ALL (FAPPLY (QUOTE CONTIG) PKT.IN))
                    (EQUAL QUEUE (QUOTE (1QUOTE NULL)))
                    (NUMBERP NEXT)
                    (NUMBERP EDGE)
                    (NOT (LESSP (DOM (LST PKT)) NEXT))
                    (NOT (LESSP NEXT (DOM (FIRST PKT))))
                    (SEQP PKT)
                    (NOT (LESSP EDGE NEXT)))
               (NOT (PMAPP PKT))),
```

which we again simplify, using linear arithmetic, applying SEQP.UPPER.LOWER, and expanding the definition of SEQP, to:

```
      T.
```

Q.E.D.

```
38639 conses
43.14 seconds
3.449 seconds, garbage collection time
[387 cns / .9 s + 0.0 gc + 0.0 io (= 1  1)]
```

```
++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#7.4.2.4.1.2'

     (OR (EQUAL (FLATTEN (APR SINK (RANGE (CONSEC M))))
                (RANGE (LOWER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (JOIN (FLATTEN PKT.IN) PKT))
                              (SUB1 (REACH M)))))
         (NOT (EQUAL (FLATTEN SINK)
                     (RANGE (LOWER (REDUCE (QUOTE WITH)
                                           (NULL)
                                           (FLATTEN PKT.IN))
                                   (SUB1 NEXT)))))
         (NOT (LESSP 0 NEXT))
         (NOT (IN 0 (DOMAIN (FLATTEN PKT.IN))))
         (NOT (FOLLOWS QUEUE
                       (UPPER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (FLATTEN PKT.IN))
                              (ADD1 NEXT))))
         (LESSP (REACH (REDUCE (QUOTE WITH)
                               (NULL)
                               (FLATTEN PKT.IN)))
                NEXT)
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.IN))))
         (NOT (CONSISTENT (QUOTE DOM)
                          (QUOTE EQUAL)
                          (JOIN (FLATTEN PKT.IN) PKT)))
         (NOT (ALL (FAPPLY (QUOTE CONTIG)
                           (APR PKT.IN PKT))))
         (NOT (PMAPP QUEUE))
         (NOT (NUMBERP NEXT))
         (NOT (NUMBERP EDGE))
         (NOT (EQUAL M
                     (REDUCE (QUOTE WITH)
                             QUEUE
                             (UPPER (LOWER PKT EDGE) NEXT)))))
     (LESSP (DOM (LST PKT)) NEXT)
     (LESSP NEXT (DOM (FIRST PKT)))
     (NOT (SEQP PKT))
     (LESSP EDGE NEXT)
     (NOT (PMAPP PKT)))
```

$ This conjecture can be simplified, using the abbreviations NOT and OR, to:$$$$$$

```
 (IMPLIES
  (AND
   (NOT
      (EQUAL (FLATTEN (APR SINK (RANGE (CONSEC M))))
             (RANGE (LOWER (REDUCE (QUOTE WITH)
                                   (QUOTE (1QUOTE NULL))
                                   (JOIN (FLATTEN PKT.IN) PKT))
                          (SUB1 (REACH M))))))
   (EQUAL (FLATTEN SINK)
          (RANGE (LOWER (REDUCE (QUOTE WITH)
                                (QUOTE (1QUOTE NULL))
                                (FLATTEN PKT.IN))
                       (SUB1 NEXT))))
   (LESSP 0 NEXT)
   (IN 0 (DOMAIN (FLATTEN PKT.IN)))
   (FOLLOWS QUEUE
            (UPPER (REDUCE (QUOTE WITH)
                           (QUOTE (1QUOTE NULL))
                           (FLATTEN PKT.IN))
                   (ADD1 NEXT)))
   (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                              (QUOTE (1QUOTE NULL))
                              (FLATTEN PKT.IN)))
               NEXT))
   (ALL (FAPPLY (QUOTE MPAIRP)
                (FLATTEN PKT.IN)))
   (CONSISTENT (QUOTE DOM)
               (QUOTE EQUAL)
               (JOIN (FLATTEN PKT.IN) PKT))
   (ALL (FAPPLY (QUOTE CONTIG)
                (APR PKT.IN PKT)))
   (PMAPP QUEUE)
   (NUMBERP NEXT)
   (NUMBERP EDGE)
   (EQUAL M
               (REDUCE (QUOTE WITH)
                       QUEUE
                       (UPPER (LOWER PKT EDGE) NEXT)))
        (NOT (LESSP (DOM (LST PKT)) NEXT))
        (NOT (LESSP NEXT (DOM (FIRST PKT))))
        (SEQP PKT)
        (NOT (LESSP EDGE NEXT)))
       (NOT (PMAPP PKT))).
```

This simplifies, applying the lemmas LST.APR, NLST.APR, REDUCE.JOIN, and APPLY1.CONTIG, and opening up FLATTEN, EQUAL, LESSP, CONTIG, FAPPLY, and ALL, to:$$$

```
      (IMPLIES
       (AND
        (NOT
         (EQUAL
          (JOIN
            (FLATTEN SINK)
            (RANGE (CONSEC (REDUCE (QUOTE WITH)
                                   QUEUE
                                   (UPPER (LOWER PKT EDGE) NEXT)))))
          (RANGE
           (LOWER
            (REDUCE (QUOTE WITH)
                    (REDUCE (QUOTE WITH)
                            (QUOTE (1QUOTE NULL))
                            (FLATTEN PKT.IN))
                    PKT)
            (SUB1
                 (REACH (REDUCE (QUOTE WITH)
                                QUEUE
                                (UPPER (LOWER PKT EDGE) NEXT))))))))
        (EQUAL (FLATTEN SINK)
               (RANGE (LOWER (REDUCE (QUOTE WITH)
                                     (QUOTE (1QUOTE NULL))
                                     (FLATTEN PKT.IN))
                             (SUB1 NEXT))))
        (NOT (EQUAL NEXT 0))
        (IN 0 (DOMAIN (FLATTEN PKT.IN)))
        (FOLLOWS QUEUE
                 (UPPER (REDUCE (QUOTE WITH)
                                (QUOTE (1QUOTE NULL))
                                (FLATTEN PKT.IN))
```

```
                         (ADD1 NEXT)))
        (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                   (QUOTE (1QUOTE NULL))
                                   (FLATTEN PKT.IN)))
                    NEXT))
        (ALL (FAPPLY (QUOTE MPAIRP)
                     (FLATTEN PKT.IN)))
        (CONSISTENT (QUOTE DOM)
                    (QUOTE EQUAL)
                    (JOIN (FLATTEN PKT.IN) PKT))
        (CONSECP (DOMAIN PKT))
        (ALL (FAPPLY (QUOTE CONTIG) PKT.IN))
        (PMAPP QUEUE)
        (NUMBERP NEXT)
        (NUMBERP EDGE)
        (NOT (LESSP (DOM (LST PKT)) NEXT))
        (NOT (LESSP NEXT (DOM (FIRST PKT))))
        (SEQP PKT)
        (NOT (LESSP EDGE NEXT)))
       (NOT (PMAPP PKT))),

which we again simplify, using linear arithmetic, applying
IN.DOMAIN.LESSP.FIRST.2, PMAPP.REDUCE.WITH,
DOM.FIRST.REDUCE.UPPER.LOWER, FOLLOWS.REDUCE.WITH.UPPER.REDUCE,
LOWER.SUB1.REACH.4, CONSISTENT.DOM.EQUAL.JOIN.REDUCE,
FOLLOWS.REDUCE.WITH.SAME, IN.DOMAIN.REDUCE.WITH,
LOWER.REDUCE.WITH.CONSISTENT, RANGE.JOIN, and EQUAL.JOIN.JOIN, and
expanding the definition of PMAPP, to:

      T.

Q.E.D.

221233 conses
279.384 seconds
35.323 seconds, garbage collection time
[506 cns / 1.2 s + 0.0 gc + 0.0 io (= 2  1)]
[270 cns / .3 s + 0.0 gc + 0.0 io (= 1  1)]
[380 cns / .4 s + 0.0 gc + 0.0 io (= 0  1)]
[356 cns / .8 s + 0.0 gc + 0.0 io (= 1  1)]
[499 cns / 1.3 s + 0.0 gc + 0.0 io (= 1  1)]


----------------------------------------
```

```
++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#7.4.2.4.2.2'

     (OR (EQUAL (FLATTEN (APR SINK (RANGE (CONSEC M))))
                (RANGE (LOWER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (JOIN (FLATTEN PKT.IN) PKT))
                              (SUB1 (REACH M)))))
         (NOT (EQUAL NEXT 0))
         (NOT (EQUAL SINK (NULL)))
         (NOT (FOLLOWS QUEUE
                       (UPPER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (FLATTEN PKT.IN))
                              (ADD1 NEXT)))))
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.IN))))
         (NOT (CONSISTENT (QUOTE DOM)
                          (QUOTE EQUAL)
                          (JOIN (FLATTEN PKT.IN) PKT)))
         (NOT (ALL (FAPPLY (QUOTE CONTIG)
                           (APR PKT.IN PKT))))
         (NOT (PMAPP QUEUE))
         (NOT (NUMBERP EDGE))
         (NOT (EQUAL M
                     (REDUCE (QUOTE WITH)
                             QUEUE
                             (UPPER (LOWER PKT EDGE) NEXT))))
         (LESSP NEXT (DOM (FIRST PKT)))
         (NOT (SEQP PKT))
         (NOT (PMAPP PKT)))

This formula can be simplified, using the abbreviations NOT and OR,
to:$$

      (IMPLIES
       (AND
        (NOT
        (EQUAL (FLATTEN (APR SINK (RANGE (CONSEC M))))
               (RANGE (LOWER (REDUCE (QUOTE WITH)
                                     (QUOTE (1QUOTE NULL))
                                     (JOIN (FLATTEN PKT.IN) PKT))
                             (SUB1 (REACH M))))))
     (EQUAL NEXT 0)
     (EQUAL SINK (QUOTE (1QUOTE NULL)))
     (FOLLOWS QUEUE
              (UPPER (REDUCE (QUOTE WITH)
                             (QUOTE (1QUOTE NULL))
                             (FLATTEN PKT.IN))
                     (ADD1 NEXT)))
     (ALL (FAPPLY (QUOTE MPAIRP)
                  (FLATTEN PKT.IN)))
     (CONSISTENT (QUOTE DOM)
                 (QUOTE EQUAL)
                 (JOIN (FLATTEN PKT.IN) PKT))
     (ALL (FAPPLY (QUOTE CONTIG)
                  (APR PKT.IN PKT)))
     (PMAPP QUEUE)
     (NUMBERP EDGE)
     (EQUAL M
            (REDUCE (QUOTE WITH)
                    QUEUE
                    (UPPER (LOWER PKT EDGE) NEXT)))
     (NOT (LESSP NEXT (DOM (FIRST PKT))))
     (SEQP PKT))
    (NOT (PMAPP PKT))),
```

which we simplify, using linear arithmetic, applying PMAPP.LOWER, SEQP.LOWER, FIRST.LOWER, UPPER.NOT.LESSP, JOIN.NULL, PSEQP.RANGE, LST.APR, NLST.APR, REDUCE.JOIN, and APPLY1.CONTIG, and opening up FLATTEN, CONTIG, FAPPLY, ALL, EQUAL, and LESSP, to:$

```
   (IMPLIES
    (AND
     (NOT
      (EQUAL
       (RANGE (CONSEC (REDUCE (QUOTE WITH)
                              QUEUE
                              (LOWER PKT EDGE))))
       (RANGE (LOWER (REDUCE (QUOTE WITH)
                             (REDUCE (QUOTE WITH)
                                     (QUOTE (1QUOTE NULL))
```

```
                                                      (FLATTEN PKT.IN))
                                             PKT)
                                    (SUB1 (REACH (REDUCE (QUOTE WITH)
                                                         QUEUE
                                                         (LOWER PKT EDGE))))))))
                (FOLLOWS QUEUE
                         (UPPER (REDUCE (QUOTE WITH)
                                        (QUOTE (1QUOTE NULL))
                                        (FLATTEN PKT.IN))
                                1))
                (ALL (FAPPLY (QUOTE MPAIRP)
                             (FLATTEN PKT.IN)))
                (CONSISTENT (QUOTE DOM)
                            (QUOTE EQUAL)
                            (JOIN (FLATTEN PKT.IN) PKT))
                (CONSECP (DOMAIN PKT))
                (ALL (FAPPLY (QUOTE CONTIG) PKT.IN))
                (PMAPP QUEUE)
                (NUMBERP EDGE)
                (EQUAL (DOM (FIRST PKT)) 0)
                (SEQP PKT))
               (NOT (PMAPP PKT))),

which we again simplify, using linear arithmetic, applying
PMAPP.REDUCE.WITH, SEQP.REDUCE.WITH, DOM.FIRST.REDUCE.WITH.ZERO,
FIRST.LOWER, SEQP.LOWER, PMAPP.LOWER,
FOLLOWS.REDUCE.WITH.LOWER.REDUCE, and LOWER.SUB1.REACH.ZERO, and
expanding the definitions of PMAPP, EQUAL, and LESSP, to:

      T.

Q.E.D.

79937 conses
88.235 seconds
10.478 seconds, garbage collection time
[334 cns / .8 s + 0.0 gc + 0.0 io (= 1  1)]
[186 cns / .2 s + 0.0 gc + 0.0 io (= 0  1)]
[304 cns / .3 s + 0.0 gc + 0.0 io (= 0  1)]


----------------------------------------
```

```
+++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#7.4.2.4.4.1'

     (OR (NOT (EQUAL NEXT 0))
         (IN 0
             (DOMAIN (JOIN (FLATTEN PKT.IN) PKT)))
         (LESSP NEXT (DOM (FIRST PKT)))
         (NOT (SEQP PKT))
         (NOT (PMAPP PKT)))

This conjecture can be simplified, using the abbreviations IN.JOIN,
NOT, OR, and DOMAIN.JOIN, to:$

      (IMPLIES (AND (EQUAL NEXT 0)
                    (NOT (IN 0 (DOMAIN (FLATTEN PKT.IN))))
                    (NOT (IN 0 (DOMAIN PKT)))
                    (NOT (LESSP NEXT (DOM (FIRST PKT))))
                    (SEQP PKT))
               (NOT (PMAPP PKT))),

which simplifies, rewriting with the lemma IN.ZERO.DOMAIN, and
expanding the functions EQUAL and LESSP, to:

      T.

Q.E.D.

10150 conses
11.509 seconds
3.519 seconds, garbage collection time
[200 cns / .5 s + 0.0 gc + 0.0 io (= 1  1)]


----------------------------------------





+++++++++++++++++++++++++++++++++++++++++
```

Proof of VC 'RECEIVER#8.4.2.1'

```
     (OR (FOLLOWS QUEUE
                  (UPPER (REDUCE (QUOTE WITH)
                                 (NULL)
                                 (JOIN (FLATTEN PKT.IN) PKT))
                         (ADD1 NEXT)))
         (NOT (FOLLOWS QUEUE
                       (UPPER (REDUCE (QUOTE WITH)
                                      (NULL)
                                      (FLATTEN PKT.IN))
                              (ADD1 NEXT))))
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.IN))))
         (NOT (CONSISTENT (QUOTE DOM)
                          (QUOTE EQUAL)
                          (JOIN (FLATTEN PKT.IN) PKT)))
         (NOT (PMAPP QUEUE))
         (NOT (PMAPP PKT)))
```

This conjecture can be simplified, using the abbreviations NOT and OR, to the new conjecture:$

```
      (IMPLIES
       (AND
           (NOT (FOLLOWS QUEUE
                         (UPPER (REDUCE (QUOTE WITH)
                                        (QUOTE (1QUOTE NULL))
                                        (JOIN (FLATTEN PKT.IN) PKT))
                                (ADD1 NEXT))))
           (FOLLOWS QUEUE
                    (UPPER (REDUCE (QUOTE WITH)
                                   (QUOTE (1QUOTE NULL))
                                   (FLATTEN PKT.IN))
                           (ADD1 NEXT)))
           (ALL (FAPPLY (QUOTE MPAIRP)
                        (FLATTEN PKT.IN)))
           (CONSISTENT (QUOTE DOM)
                       (QUOTE EQUAL)
                       (JOIN (FLATTEN PKT.IN) PKT))
           (PMAPP QUEUE))
       (NOT (PMAPP PKT))).
```

```
This simplifies, applying REDUCE.JOIN, FOLLOWS.UPPER.UPPER,
ALL.FAPPLY.MPAIRP, PMAPP.REDUCE.WITH,
CONSISTENT.DOM.EQUAL.JOIN.REDUCE, FOLLOWS.REDUCE.WITH.SAME,
FOLLOWS.REDUCE.REDUCE.WITH, FOLLOWS.REDUCE.WITH.3, and
FOLLOWS.TRANS, and expanding the functions FOLLOWS and PMAPP, to:

      T.

Q.E.D.

31903 conses
35.162 seconds
3.529 seconds, garbage collection time
[216 cns / .3 s + 0.0 gc + 0.0 io (= 0  1)]


----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#8.4.2.2.1'

     (OR (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                    (NULL)
                                    (JOIN (FLATTEN PKT.IN) PKT)))
                        NEXT))
         (LESSP (REACH (REDUCE (QUOTE WITH)
                               (NULL)
                               (FLATTEN PKT.IN)))
                NEXT)
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.IN))))
         (NOT (IN 0 (DOMAIN (FLATTEN PKT.IN))))
         (NOT (LESSP 0 NEXT))
         (NOT (NUMBERP NEXT))
         (NOT (PMAPP PKT)))

This conjecture can be simplified, using the abbreviations NOT and
OR, to:$
```

```
(IMPLIES
      (AND (LESSP (REACH (REDUCE (QUOTE WITH)
                                 (QUOTE (1QUOTE NULL))
                                 (JOIN (FLATTEN PKT.IN) PKT)))
                  NEXT)
           (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                      (QUOTE (1QUOTE NULL))
                                      (FLATTEN PKT.IN)))
                       NEXT))
           (ALL (FAPPLY (QUOTE MPAIRP)
                        (FLATTEN PKT.IN)))
           (IN 0 (DOMAIN (FLATTEN PKT.IN)))
           (LESSP 0 NEXT)
           (NUMBERP NEXT))
      (NOT (PMAPP PKT))).
```

This simplifies, applying the lemma REDUCE.JOIN, and opening up EQUAL and LESSP, to:$

```
(IMPLIES
      (AND (LESSP (REACH (REDUCE (QUOTE WITH)
                                 (REDUCE (QUOTE WITH)
                                         (QUOTE (1QUOTE NULL))
                                         (FLATTEN PKT.IN))
                                 PKT))
                  NEXT)
           (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                      (QUOTE (1QUOTE NULL))
                                      (FLATTEN PKT.IN)))
                       NEXT))
           (ALL (FAPPLY (QUOTE MPAIRP)
                        (FLATTEN PKT.IN)))
           (IN 0 (DOMAIN (FLATTEN PKT.IN)))
           (NOT (EQUAL NEXT 0))
           (NUMBERP NEXT))
      (NOT (PMAPP PKT))),
```

which we again simplify, using linear arithmetic, applying LESSP.REACH.REDUCE.WITH.2, IN.DOMAIN.REDUCE.WITH, PMAPP.REDUCE.WITH, and ALL.FAPPLY.MPAIRP, and expanding the definition of PMAPP, to:

```
T.
```

Q.E.D.

```
40315 conses
41.37 seconds
6.885 seconds, garbage collection time

-------------------------------------------




+++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#8.4.2.2.2'

     (OR (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                                    (NULL)
                                    (JOIN (FLATTEN PKT.IN) PKT)))
                     NEXT))
         (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                           (FLATTEN PKT.IN))))
         (IN 0 (DOMAIN (FLATTEN PKT.IN)))
         (NOT (EQUAL NEXT 0))
         (NOT (PMAPP PKT)))

This formula can be simplified, using the abbreviations NOT and OR,
to:

      (IMPLIES
           (AND (LESSP (REACH (REDUCE (QUOTE WITH)
                                      (QUOTE (1QUOTE NULL))
                                      (JOIN (FLATTEN PKT.IN) PKT)))
                       NEXT)
                (ALL (FAPPLY (QUOTE MPAIRP)
                             (FLATTEN PKT.IN)))
                (NOT (IN 0 (DOMAIN (FLATTEN PKT.IN))))
                (EQUAL NEXT 0))
           (NOT (PMAPP PKT))).

This simplifies, using linear arithmetic, to:

      T.
```

```
Q.E.D.

948 conses
1.575 seconds
0.0 seconds, garbage collection time

----------------------------------------




++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#8.4.2.4.1.2'

    (OR (EQUAL (FLATTEN SINK)
               (RANGE (LOWER (REDUCE (QUOTE WITH)
                                     (NULL)
                                     (JOIN (FLATTEN PKT.IN) PKT))
                              (SUB1 NEXT))))
        (NOT (EQUAL (FLATTEN SINK)
                    (RANGE (LOWER (REDUCE (QUOTE WITH)
                                          (NULL)
                                          (FLATTEN PKT.IN))
                                  (SUB1 NEXT)))))
        (NOT (LESSP 0 NEXT))
        (NOT (IN 0 (DOMAIN (FLATTEN PKT.IN))))
        (LESSP (REACH (REDUCE (QUOTE WITH)
                              (NULL)
                              (FLATTEN PKT.IN)))
               NEXT)
        (NOT (ALL (FAPPLY (QUOTE MPAIRP)
                          (FLATTEN PKT.IN))))
        (NOT (CONSISTENT (QUOTE DOM)
                         (QUOTE EQUAL)
                         (JOIN (FLATTEN PKT.IN) PKT)))
        (NOT (NUMBERP NEXT))
        (NOT (PMAPP PKT)))

This conjecture can be simplified, using the abbreviations NOT and
OR, to the conjecture:$
```

```
(IMPLIES
 (AND
  (NOT
     (EQUAL (FLATTEN SINK)
            (RANGE (LOWER (REDUCE (QUOTE WITH)
                                  (QUOTE (1QUOTE NULL))
                                  (JOIN (FLATTEN PKT.IN) PKT))
                          (SUB1 NEXT)))))
  (EQUAL (FLATTEN SINK)
         (RANGE (LOWER (REDUCE (QUOTE WITH)
                               (QUOTE (1QUOTE NULL))
                               (FLATTEN PKT.IN))
                       (SUB1 NEXT))))
  (LESSP 0 NEXT)
  (IN 0 (DOMAIN (FLATTEN PKT.IN)))
  (NOT (LESSP (REACH (REDUCE (QUOTE WITH)
                             (QUOTE (1QUOTE NULL))
                             (FLATTEN PKT.IN)))
              NEXT))
  (ALL (FAPPLY (QUOTE MPAIRP)
               (FLATTEN PKT.IN)))
  (CONSISTENT (QUOTE DOM)
              (QUOTE EQUAL)
              (JOIN (FLATTEN PKT.IN) PKT))
  (NUMBERP NEXT))
 (NOT (PMAPP PKT))),
```

```
which we simplify, using linear arithmetic, applying REDUCE.JOIN,
PMAPP.REDUCE.WITH, CONSISTENT.DOM.EQUAL.JOIN.REDUCE,
FOLLOWS.REDUCE.WITH.SAME, IN.DOMAIN.REDUCE.WITH, and
LOWER.REDUCE.WITH.CONSISTENT, and expanding the definitions of
PMAPP and LESSP, to:

      T.

Q.E.D.

29349 conses
31.8 seconds
3.468 seconds, garbage collection time
[256 cns / .3 s + 0.0 gc + 0.0 io (= 0  1)]

----------------------------------------
```

```
++++++++++++++++++++++++++++++++++++++++

Proof of VC  'RECEIVER#8.4.2.4.2.1'

     (OR (NOT (IN 0
                  (DOMAIN (JOIN (FLATTEN PKT.IN) PKT))))
         (NOT (EQUAL PKT (NULL)))
         (IN 0 (DOMAIN (FLATTEN PKT.IN))))

This conjecture can be simplified, using the abbreviations NOT, OR,
and DOMAIN.JOIN, to the conjecture:

      (IMPLIES (AND (IN 0
                        (JOIN (DOMAIN (FLATTEN PKT.IN))
                              (DOMAIN PKT)))
                    (EQUAL PKT (QUOTE (1QUOTE NULL))))
               (IN 0 (DOMAIN (FLATTEN PKT.IN)))),

which we simplify, expanding the definitions of DOMAIN, SEQP, and
JOIN, to:

      T.

Q.E.D.

6272 conses
7.071 seconds
0.0 seconds, garbage collection time

----------------------------------------
```

REFERENCES

[Boyer 79]	R. S. Boyer and J. S. Moore. <u>A Computational Logic</u>. Academic Press, New York, 1979.
[DiVito 82]	B. L. DiVito. <u>Verification of Communications Protocols and Abstract Process Models</u>. PhD thesis, University of Texas at Austin, 1982. Technical Report 25, Institute for Computing Science.
[Postel 80]	J. Postel, editor. DoD Standard Transmission Control Protocol. <u>ACM SIGCOMM</u> 10(4), October, 1980.