```
Message transformation
--------------------------------------------------------------------------------

Implementation of "mapI".

  mapI f p = map f id p.

  (the type of id instantiates to (O -> O) )

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


Corollary (mapI-compose):

  forall

    p : IProc X O ,
    f : I -> X ,
    (=I), (=X), (=O),

  if

    p ∈ NI(=X,=O) , and
    f ∈ NI(=I,=X) ∩ PS(=I,=X),

  then

    (mapI f p) ∈ NI(=I,=O).

Proof.

  By definition of id,
    id ∈ NI(=O,=O).

  Set
    g    = id,
    Y    = O, and
    (=Y) = (=O).

  Then by Theorem (map-compose),
    map f id p ∈ NI(=I,=Y).

  Thus, by definition of (=O),
    map f id p ∈ NI(=I,=O).

  Thus, by definition of "mapI",
    mapI f p ∈ NI(=I,=O).

Qed.

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


Implementation of "mapO".

  mapO g p = map id g p.

  (the type of id instantiates to (I -> I) )

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


Corollary (mapO-compose):
```

```
  forall

    p : IProc I O ,
    g : O -> Y ,
    (=I), (=O), (=Y),

  if

    p ∈ NI(=I,=O) ,
    g ∈ NI(=O,=Y) ,

  then

    (mapO g p) ∈ NI(=I,=Y).

Proof.

  By definition of id,
    id ∈ PS(=I,=X), and
    id ∈ NI(=I,=X).

  Set
    f    = id,
    X    = I, and
    (=X) = (=I).

  Then by Theorem (map-compose),
    map id g p ∈ NI(=I,=Y).

  Thus, by definition of "mapO",
    mapO g p ∈ NI(=I,=Y).

Qed.

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


Message filtering
--------------------------------------------------------------------------------

Definition

  forall
    f : A -> B
    A' ⊆ A
  let
    f|A' = { ⟨a',f a'⟩ | a' ∈ A' }

End Definition

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


Implementation of "filter"

  filter f g p = map fM gM (maybe p),
    where
      fM i | (f i)     = Just i
           | otherwise = Nothing
      gM i | (g o)     = Just o
           | otherwise = Nothing.


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


Corollary (filter-compose):

  forall
```

```
    p : IProc I O ,
    f : I -> Bool ,
    g : O -> Bool ,
    (=I), (=O),

  if

    p ∈ NI(=I,=O),
    f|(≠I●) ∈ NI(=I,=Bool), and
    g|(≠O●) ∈ NI(=O,=Bool)

  then

    (filter f g p) ∈ NI(=I,=MaybeO),

  where

    (=MaybeO) = eqmaybe(=O), and
    (=Bool)l = { ⟨True,True⟩, ⟨False,False⟩, ⟨●,●⟩}

Proof.

  Let
    (=MaybeI) = eqmaybe(=I).

  Since
    p ∈ NI(=I,=O),
  we get by definition of (=MaybeI) that
    (maybe p) ∈ NI(=MaybeI,=O).

  Let
    fM i | (f i)     = Just i
         | otherwise = Nothing.

  Since
    f|(≠I●) ∈ NI(=I,=Bool),
  we get by definition of fM and (=MaybeI) that
    fM|(≠I●) ∈ NI(=I,=MaybeI).
  By definition of (=MaybeI), we have that
  forall l,
    Nothing (=MaybeI)l ●, and
    i =Il ● => Just i (=MaybeI)l ●.
  Thus, we get
    fM ∈ NI(=I,=MaybeI).

  Since
    i =Il ● => Just i (=MaybeI)l ●,
  we have
    fM ∈ PS(=I,=MaybeI).

  Let
    gM o | (g i)     = Just o
         | otherwise = Nothing.

  Since
    g|(≠O●) ∈ NI(=O,=Bool),
  we get by definition of gM and (=MaybeO) that
    gM|(≠O●) ∈ NI(=O,=MaybeO)
  By definition of (=MaybeO), we have that
  forall l,
    Nothing (=MaybeO)l ●, and
    o =Ol ● => Just o (=MaybeO)l ●.
  Thus, we get
    gM ∈ NI(=O,=MaybeO).

  Since
    fM ∈ NI(=I,=MaybeI),
    fM ∈ PS(=I,=MaybeI),
```

```
    gM ∈ NI(=O,=MaybeO), and
    (maybe p) ∈ NI(=MaybeI,=O),
  we get that
    map fM gM (maybe p) ∈ NI(=I,=MaybeO).

  Since
    filter f g p = map fM gM (maybe p), and
    map fM gM (maybe p) ∈ NI(=I,=MaybeO),
  we get
    filter f g p ∈ NI(=I,=MaybeO).

Qed.
```

- -

Implementation of "filterI"

```
  filterI f p = mapI fM (maybe p),
    where
      fM i | (f i)     = Just i
           | otherwise = Nothing
```

- -

Corollary (filterI-compose):

```
  forall

    p : IProc I O ,
    f : I -> Bool ,
    (=I), (=O),

  if

    p ∈ NI(=I,=O), and
    f|(≠I●) ∈ NI(=I,=Bool)

  then

    (filterI f p) ∈ NI(=I,=O),

  where

    (=Bool)l = { ⟨True,True⟩, ⟨False,False⟩, ⟨●,●⟩}
```

Proof.

```
  Let
    (=MaybeI) = eqmaybe(=I).

  Since
    p ∈ NI(=I,=O),
  we get by definition of (=MaybeI) that
    (maybe p) ∈ NI(=MaybeI,=O).

  Let
    fM i | (f i)     = Just i
         | otherwise = Nothing.

  Since
    f|(≠I●) ∈ NI(=I,=Bool),
  we get by definition of fM and (=MaybeI) that
    fM|(≠I●) ∈ NI(=I,=MaybeI).
  By definition of (=MaybeI), we have that
  forall l,
    Nothing (=MaybeI)l ●, and
    i =Il ● => Just i (=MaybeI)l ●.
```

```
  Thus, we get
    fM ∈ NI(=I,=MaybeI).

  Since
    i =Il ● => Just i (=MaybeI)l ●,
  we have
    fM ∈ PS(=I,=MaybeI).

  Since
    fM ∈ NI(=I,=MaybeI),
    fM ∈ PS(=I,=MaybeI), and
    (maybe p) ∈ NI(=MaybeI,=O),
  we get that
    mapI fM (maybe p) ∈ NI(=I,=O).

  Since
    filterI f p = mapI fM (maybe p), and
    map fM (maybe p) ∈ NI(=I,=O),
  we get
    filter f g p ∈ NI(=I,=O).

Qed.

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


Implementation of "filterO"

  filterO g p = mapO gM p,
    where
      gM i | (g o)     = Just o
           | otherwise = Nothing.


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


Corollary (filterO-compose):

  forall

    p : IProc I O ,
    g : O -> Bool ,
    (=I), (=O),

  if

    p ∈ NI(=I,=O), and
    g|(≠O●) ∈ NI(=O,=Bool)

  then

    (filterO g p) ∈ NI(=I,=MaybeO),

  where

    (=MaybeO) = eqmaybe(=O), and
    (=Bool)l = { ⟨True,True⟩, ⟨False,False⟩, ⟨●,●⟩}

Proof.

  Let
    gM o | (g i)     = Just o
         | otherwise = Nothing.

  Since
    g|(≠O●) ∈ NI(=O,=Bool),
  we get by definition of gM and (=MaybeO) that
    gM|(≠O●) ∈ NI(=O,=MaybeO)
```

```
  By definition of (=MaybeO), we have that
  forall l,
    Nothing (=MaybeO)l ●, and
    o =Ol ● => Just o (=MaybeO)l ●.
  Thus, we get
    gM ∈ NI(=O,=MaybeO).

  Since
    gM ∈ NI(=O,=MaybeO),
    p  ∈ NI(=I,=O),
  we get that
    mapO gM p ∈ NI(=I,=MaybeO).

  Since
    filterO g p = mapO gM p, and
    mapO gM p ∈ NI(=I,=MaybeO),
  we get
    filterO g p ∈ NI(=I,=MaybeO).

Qed.
```

- -

```
Implementation of "source".

  source p = mapI f (maybe p),
    where
      f _ = Nothing.
```

- -

```
Corollary (source-compose):

  forall

    p : IProc Í O ,
    (=I), (=O),

  it holds that

    (source p) ∈ NI(=I,=O).

Proof.

  Let
    (=Í)l = { ⟨í,í⟩ | í ∈ Í }.

  Then by Theorem (observe all input),
    p ∈ NI(=Í,=O).

  Let
    (=MaybeÍ) = eqmaybe(=Í).

  Then, by Theorem (maybe-compose),
    (maybe p) ∈ NI(=MaybeÍ,=O).

  Let
    f _ = Nothing.

  Then, since
    forall i, i' . i =Il i' => f i = f i' = Nothing,
  we get that
    f ∈ NI(=I,=MaybeÍ).

  Since
    forall l . Nothing (=MaybeÍ)l ●,
  we get
```

```
    f ∈ PS(=I,=Maybeİ).

  Since
    (maybe p) ∈ NI(=MaybeÍ,=O),
    f ∈ NI(=I,=MaybeÍ), and
    f ∈ PS(=I,=MaybeÍ),
  we get by Corollary (mapI-compose) that
    mapI f (maybe p) ∈ NI(=I,=O).

  Since
    source p = mapI f (maybe p), and
    mapI f (maybe p) ∈ NI(=I,=O),
  we get
    source p ∈ NI(=I,=O).

Qed.

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


Process State
--------------------------------------------------------------------------------

Implementation of "staI".

  staI f v p = sta f (curry snd) v p.

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


Corollary (staI-compose):

  forall

    p ∈ IProc (V*I) O ,
    f : I -> V -> V,
    (=I), (=V), (=O),

  if

    p ∈ NI(=V*I,=O)
    f ∈ NI(=I,=V,=V) ∩ PE(=I,=V),

  then forall v,

    (staI f v p) ∈ NI(=I,=V*O),

  where

    (=V*I) = eqpair●R(=V,=I)
    (=V*O) = eqpair(=V,=O)

Proof.

  Let
    g = curry snd.

  By definition of g,
    g ∈ NI(=O,=V,=V).

  Then by Theorem (sta-compose),
    sta f g v p ∈ NI(=I,=V*O).

Qed.

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


Implementation of "staO".
```

```
  staI f v p = sta (curry snd) f

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


Corollary (staO-compose):

  forall

    p ∈ IProc (V*I) O ,
    g : O -> V -> V,
    (=I), (=V), (=O),

  if

    p ∈ NI(=V*I,=O), and
    g ∈ NI(=O,=V,=V),

  then forall v,

    (staO f v p) ∈ NI(=I,=V*O),

  where

    (=V*I) = eqpair●R(=V,=I)
    (=V*O) = eqpair(=V,=O)

Proof.

  Let
    f = curry snd.

  By definition of g,
    f ∈ NI(=I,=V,=V) ∩ PE(=I,=V).

  Then by Theorem (sta-compose),
    sta f g v p ∈ NI(=I,=V*O).

Qed.

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


Process Switching
--------------------------------------------------------------------------------

Implementation of "swiI".

  swiI b p = swi b (mapO f p),
    where
      f o = ⟨False,o⟩.

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


Corollary (swiI-compose):

  forall

    p ∈ IProc I O ,
    (=I), (=O), (=Bool),

  if

    p ∈ NI(=I,=O), and
    forall l . l ∉ A(True,=Bool) => l ∈ O(p,=O)

  then forall b,
```

```
    (swiI b p) ∈ NI(=Bool*I,=MaybeO),

  where

    (=Bool*I) = eqpair●LR(=Bool,=I)
    (=MaybeO) = eqmaybe(A(True,=Bool),=O).

Proof.

  Let
    f o = ⟨False,o⟩.

  Then
    swiI b p = swi b (mapO f p).

  Let
    (=Bool*O) = eqpair●R (=Bool,=O)

  By definition of f,
    f ∈ NI(=O,=Bool*O)

  By definition of f and (=Bool*O),
    forall l . l ∉ A(True,=Bool) => l ∈ O(p,=Bool*O)

  Thus by Theorem (swi-compose),
    swi b (mapO f p) ∈ NI(=Bool*I,=MaybeO).

  Thus, by definition of "swiI",
    swiI f v p ∈ NI(=Bool*I,=MaybeO).

Qed.

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


Implementation of "swiO".

  swiO b p = mapI f (swi b p),
    where
      f i = ⟨False,i⟩.

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


Corollary (swiO-compose):

  forall

    p ∈ IProc I (Bool*O) ,
    (=I), (=O), (=Bool),

  if

    p ∈ NI(=I,=O), and
    forall l . l ∉ A(True,=Bool) => l ∈ O(p,=Bool*O)

  then forall b,

    (swiO b p) ∈ NI(=I,=MaybeO),

  where

    (=Bool*O) = eqpair●R(=Bool,=O)
    (=MaybeO) = eqmaybe(A(True,=Bool),=O).

Proof.

  Let
```

```
    f i = ⟨False,i⟩.

  Then
    swiO b p = mapI f (swi b p).

  Let
    (=Bool*I) = eqpair●LR (=Bool,=I)

  By definition of f,
    f ∈ NI(=I,=Bool*I) ∩ PS(=I,=Bool*I).

  Thus by Theorem (swi-compose),
    mapI f (swi b p) ∈ NI(=I,=MaybeO).

  Thus, by definition of "swiO",
    swiO b p ∈ NI(=I,=MaybeO).

Qed.

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


Implementation of "swiC".

  swiC b p =
    mapO snd (
    sta f g b (
    mapI (onsnd snd) (
    swi b p
    )))
   where
    f ⟨bI,i⟩       True  = False
    f ⟨bI,i⟩       False = bI
    g ⟨bO,Nothing⟩ bS    = False
    g ⟨bO,Just o⟩  bS    = True

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


Corollary (swiC-compose):

  forall

    p ∈ IProc I (Bool*O) ,
    (=I), (=O), (=Bool),

  if

    p ∈ NI(=I,=Bool*O), and
    forall l . l ∉ A(True,=Bool) => l ∈ O(p,=Bool*O)

  then forall b,

    (swiC b p) ∈ NI(=Bool*I,=MaybeO),

  where

    (=Bool*I) = eqpair●LR(=Bool,=I)
    (=Bool*O) = eqpair●R (=Bool,=O)
    (=MaybeO) = eqmaybe(A(True,=Bool),=O).

Proof.

  Let
    f ⟨bI,i⟩  True  = False
    f ⟨bI,i⟩  False = bI
    g Nothing bS    = False
    g Just o  bS    = True.
```

```
By definition of f and g,
  f ∈ NI(=Bool*I,=Bool,=Bool),
  f ∈ PE(=Bool*I,=Bool,=Bool), and
  g ∈ NI(=MaybeO,=Bool,=Bool).

Then
  swiC b p =
    mapO snd (
    sta f g b (
    mapI (onsnd snd) (
    swi b p
    ))).

Let
  (=Bool*I) = eqpair●LR(=Bool,=I)
  (=Bool*O) = eqpair●R (=Bool,=O)
  (=MaybeO) = eqmaybe(A(True,=Bool),=O).

Since
  p ∈ NI(=I,=Bool*O), and
  forall l . l ∉ A(True,=Bool) => l ∈ O(p,=Bool*O),
we get by Theorem (swi-compose) that
  swi b p
  ∈ NI(=Bool*I,=MaybeO).

By definition of snd,
  snd ∈ NI(=Bool*I,=I), and
  snd ∈ PS(=Bool*I,=I).

Since
  snd ∈ NI(=Bool*I,=I), and
  snd ∈ PS(=Bool*I,=I),
we get by definition of onsnd that
  (onsnd snd) ∈ NI(=Bool*(Bool*I),=Bool*I), and
  (onsnd snd) ∈ PS(=Bool*(Bool*I),=Bool*I).
where
  (=Bool*(Bool*I)) = eqpair●LR(=Bool,=Bool*I).

By this, and by
  swi b p
  ∈ NI(=Bool*I,=MaybeO),
we get by Corollary (mapI-compose) that
  mapI (onsnd snd) (
  swi b p
  )
  ∈ NI(=Bool*(Bool*I),=MaybeO).

By this, and since
  f ∈ NI(=Bool*I,=Bool,=Bool),
  f ∈ PE(=Bool*I,=Bool,=Bool), and
  g ∈ NI(=MaybeO,=Bool,=Bool),
we get by Theorem (sta-compose) that
  sta f g b (
  mapI (onsnd snd) (
  swi b p
  ))
  ∈ NI(=Bool*I,=Bool*MaybeO),
where
  (=Bool*MaybeO) = eqpair●R(=Bool,=MaybeO).

By this, and since
  snd ∈ NI(=Bool*MaybeO,=MaybeO),
we get by Corollary (mapO-compose) that
  mapO snd (
  sta f g b (
  mapI (onsnd snd) (
  swi b p
  )))
  ∈ NI(=Bool*I,=MaybeO).
```

```
  Since
    swiC b p =
      mapO snd (
      sta f g b (
      mapI (onsnd snd) (
      swi b p
      ))).
  and
    mapO snd (
    sta f g b (
    mapI (onsnd snd) (
    swi b p
    )))
    ∈ NI(=Bool*I,=MaybeO),
  we get
    swiC b p
    ∈ NI(=Bool*I,=MaybeO).

Qed.

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


Process Composition
-----------------------------------------------------------------------------------

Implementation of "uni".

  uni p q =
    mapI topair (
      loop (
        par (
          mapI snd p
        )(
          mapI fst q
        )
      )
    )

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


Corollary (uni-compose)

  forall

    p, q ∈ IProc I I ,
    (=I),

  if

    p, q ∈ NI(=I,=I),

  then

    (uni p q) ∈ NI(=I,=I*I),

  where

    (=I*I) = eqpair●LR(=A,=B).

Proof.

  Since
    snd ∈ NI(=I*I,=I),
    snd ∈ PS(=I*I,=I), and
    p   ∈ NI(=I,=I),
  we get by Corollary (mapI-compose) that
```

```
  mapI snd p ∈ NI(=I*I,=I).

Since
  fst ∈ NI(=I*I,=I),
  fst ∈ PS(=I*I,=I), and
  q   ∈ NI(=I,=I),
we get by Corollary (mapI-compose) that
  mapI fst q ∈ NI(=I*I,=I).

Since
  mapI snd p ∈ NI(=I*I,=I), and
  mapI fst q ∈ NI(=I*I,=I),
we get by Theorem (par-compose) that
  par (
    mapI snd p
  )(
    mapI fst q
  )
  ∈ NI(=I*I,=I*I).

Since
  par (
    mapI snd p
  )(
    mapI fst q
  )
  ∈ NI(=I*I,=I*I),
we get by Theorem (loop-compose) that
  loop (
    par (
      mapI snd p
    )(
      mapI fst q
    )
  )
  ∈ NI(=I*I,=I*I).

Since
  topair ∈ NI(=I,=I*I),
  topair ∈ PS(=I,=I*I), and
  loop (
    par (
      mapI snd p
    )(
      mapI fst q
    )
  )
  ∈ NI(=I*I,=I*I),
we get by Corollary (mapI-compose) that
  mapI topair (
    loop (
      par (
        mapI snd p
      )(
        mapI fst q
      )
    )
  )
  ∈ NI(=I,=I*I).

Since
  uni p q =
  mapI topair (
    loop (
      par (
        mapI snd p
      )(
        mapI fst q
      )
```

```
      )
    )
    ∈ NI(=I,=I*I),
  we get
    uni p q ∈ NI(=I,=I*I).

Qed.
```

- -

Implementation of "seq".

```
  seq p q = source q , IF O is uninhabited. OTHERWISE, for some o ∈ O,

  seq p q =
    mapO (fromRight o ○ fromRight Right o) (
      mapO snd (
        mapI Left (
          uni (                            (Either I (Either V O)) (Either I (Either V O))
            mapO Right ○ Left(
              mapI left  (
                maybe p
              )
            )
          )(
            mapO Right ○ Right(
              mapI right (
                maybe (
                  mapI left (
                    maybe q
                  )
                )
              )
            )
          )
        )
      )
    )
```

Here,

```
  fromLeft : A -> Either A B -> A
  fromRight: B -> Either A B -> B
  left     : Either A B -> Maybe A
  right    : Either A B -> Maybe B
```

- -

Let

```
  eqeither'(=A,=B) l = { ⟨Left  a,Left a' ⟩ | a =Al a' }
                     ∪ { ⟨Right b,Right b'⟩ | b =Bl b' }
  eqeither●'(=A,=B)l = { ⟨Left  a,●⟩ | a =Al ● }
                     ∪ { ⟨Right b,●⟩ | b =Bl ● }
  eqeither(=A,=B)  l = RTC( eqeither'(=A,=B) ∪ eqeither●'(=A,=B) )
```

- -

Corollary (seq-compose)

```
  forall

    p ∈ IProc I V ,
    q ∈ IProc V O ,
    (=I), (=V), (=O),

  if
```

```
    p ∈ NI(=I,=V),
    q ∈ NI(=V,=O),

  then

    (seq p q) ∈ NI(=I,=O).

Proof.

  Let
    (=mI) = eqmaybe(=I).

  Since
    (=mI) = eqmaybe(=I), and
    p
    ∈ IProc I V
    ∈ NI(=I,=V)
  we get by Theorem (maybe-compose) that
    maybe p
    ∈ IProc (Maybe I) V.
    ∈ NI(=mI,=V)

  Let
    (=V+O) = eqeither(=V,=O), and
    (=I+V+O) = eqeither(=I,=V+O).

  By definition of left and (=I+V+O),
    left ∈ NI(=I+V+O,=mI).

  By definition of left, (=I+V+O) and (=mI),
    left ∈ PS(=I+V+O,=mI).

  Since
    left ∈ NI(=I+V+O,=mI),
    left ∈ PS(=I+V+O,=mI), and
    maybe p
    ∈ IProc (Maybe I) V.
    ∈ NI(=mI,=V)
  we get by Corollary (mapI-compose) that
    mapI left (maybe p)
    ∈ IProc (Either I (Either V O)) V.
    ∈ NI(=I+V+O,=V)

  By definition of Left and (=V+O),
    Left ∈ NI(=V,=V+O).
    Left : V -> {Left v | v ∈ V}

  By definition of (=V+O) and (=I+V+O),
    Right ∈ NI(=V+O,=I+V+O).
    Right : V+O -> {Right vo | vo ∈ V+O}

  Thus
    Right ○ Left ∈ NI(=V,=I+V+O).
    Right ○ Left : V -> {Right Left v | v ∈ V}

  Since
    Right ○ Left ∈ NI(=V,=I+V+O), and
    mapI left (maybe p)
    ∈ IProc (Either I (Either V O)) V,
    ∈ NI(=I+V+O,=V)
  we get by Corollary (mapO-compose) that
    mapO Right ○ Left (mapI left (maybe p))
    ∈ IProc (Either I (Either V O)) (Either I (Either V O))
    ∈ IProc (Either I (Either V O)) {Right Left v | v ∈ V}
    ∈ NI(=I+V+O,=I+V+O).

  By analogous reasoning,
    mapO Right ○ Right (mapI right (maybe (mapI left (maybe q))))
```

```
  ∈ IProc (Either I (Either V O)) (Either I (Either V O)).
  ∈ IProc (Either I (Either V O)) {Right Right o | o ∈ O}
  ∈ NI(=I+V+O,=I+V+O).

Let
  pU = mapO Right ○ Left (mapI left (maybe p))
  qU = mapO Right ○ Right (mapI right (maybe (mapI left (maybe q)))).

Let
  (=(I+V+O)^2) = eqpair●LR(=I+V+O,=I+V+O)

Since
  pU
  ∈ IProc (Either I (Either V O)) (Either I (Either V O))
  ∈ NI(=I+V+O,=I+V+O), and
  qU
  ∈ IProc (Either I (Either V O)) (Either I (Either V O))
  ∈ NI(=I+V+O,=I+V+O),
we get
  uni pU qU
  ∈ IProc (Either I (Either V O)) (Either I (Either V O))^2
  ∈ IProc (Either I (Either V O)) {Right Left v | v ∈ V}*{Right Right o | o ∈ O}
  ∈ NI(=I+V+O,=(I+V+O)^2).

By definition of Left and (=I+V+O),
  Left ∈ NI(=I,=I+V+O).

By definition of Left, (=I+V+O) and (=I),
  Left ∈ PS(=I,=I+V+O).

Since
  Left ∈ NI(=I,=I+V+O),
  Left ∈ PS(=I,=I+V+O), and
  uni pU qU
  ∈ IProc (Either I (Either V O)) (Either I (Either V O))^2
  ∈ IProc (Either I (Either V O)) {Right Left v | v ∈ V}*{Right Right o | o ∈ O}
  ∈ NI(=I+V+O,=(I+V+O)^2),
we get by Corollary (mapI-compose) that
  mapI Left (uni pU qU)
  ∈ IProc I (Either I (Either V O))^2
  ∈ IProc I {Right Left v | v ∈ V}*{Right Right o | o ∈ O}
  ∈ NI(=I,=(I+V+O)^2).

By definition of snd, (=(I+V+O)^2) and (=I+V+O),
we get
  snd ∈ NI(=(I+V+O)^2,=I+V+O).

Since
  snd ∈ NI(=(I+V+O)^2,=I+V+O), and
  mapI Left (uni pU qU)
  ∈ IProc I (Either I (Either V O))^2
  ∈ NI(=I,=(I+V+O)^2),
we get
  mapO snd (mapI Left (uni pU qU))
  ∈ IProc I (Either I (Either V O))
  ∈ IProc I {Right Right o | o ∈ O}
  ∈ NI(=I,=I+V+O).

Since
  mapO snd (mapI Left (uni pU qU))
  ∈ IProc I (Either I (Either V O))
  ∈ IProc I {Right Right o | o ∈ O}
  ∈ NI(=I,=I+V+O).

For
  fromRight : {Right o | o ∈ O } -> {Right Right o | o ∈ O} -> {Right o | o ∈ O},
we get by definition of fromRight, (=I+V+O), and (=V+O) that
  fromRight Right o
  : {Right Right o | o ∈ O} -> {Right o | o ∈ O}
```

```
    ∈ NI(=I+V+O,=V+O).

  By similar reasoning,
    fromRight o
    : {Right o | o ∈ O} -> O
    ∈ NI(=V+O,=O).

  Thus,
    fromRight o ○ fromRight Right o
    : {Right Right o | o ∈ O} -> O
    ∈ NI(=I+V+O,=O).

  Since
    fromRight o ○ fromRight Right o
    : {Right Right o | o ∈ O} -> O
    ∈ NI(=I+V+O,=O),
  and
    mapO snd (mapI Left (uni pU qU))
    ∈ IProc I (Either I (Either V O))
    ∈ IProc I {Right Right o | o ∈ O}
    ∈ NI(=I,=I+V+O),
  we get by Corollary (mapO-compose) that
    mapO (fromRight o ○ fromRight Right o) (
      mapO snd (mapI Left (uni pU qU))
    )
    ∈ IProc I O,
    ∈ NI(=I,=O).

  By the above, we get by definition of seq that
    seq p q
    ∈ IProc I O,
    ∈ NI(=I,=O).

Qed.


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

Implementation of "cut".

  cut p q = source q , IF O is uninhabited. OTHERWISE, for some o ∈ O,

  cut p q =
    mapO (fromRight o) (
      mapO snd (
        mapI Left (
          uni (                         IProc (Either I O) (Either I O)
            mapO Left(
              mapI left  (
                maybe p
              )
            )
          )(
            mapO Right
              mapI left (
                maybe q
              )
            )
          )
        )
      )
    )

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

Corollary (cut-compose)

  forall
```

```
    p ∈ IProc I I ,
    q ∈ IProc I O ,
    (=I), (=O),

  if

    p ∈ NI(=I,=I),
    q ∈ NI(=I,=O),

  then

    (cut p q) ∈ NI(=I,=O).

Proof.

  Let
    (=mI) = eqmaybe(=I).

  Since
    (=mI) = eqmaybe(=I), and
    p
    ∈ IProc I I
    ∈ NI(=I,=I)
  we get by Theorem (maybe-compose) that
    maybe p
    ∈ IProc (Maybe I) I
    ∈ NI(=mI,=I).

  Let
    (=I+O) = eqeither(=I,=O).

  By definition of left and (=I+O),
    left ∈ NI(=I+O,=mI).

  By definition of left, (=I+O) and (=mI),
    left ∈ PS(=I+O,=mI).

  Since
    left ∈ NI(=I+O,=mI),
    left ∈ PS(=I+O,=mI), and
    maybe p
    ∈ IProc (Maybe I) I.
    ∈ NI(=mI,=I)
  we get by Corollary (mapI-compose) that
    mapI left (maybe p)
    ∈ IProc (Either I O) I.
    ∈ NI(=I+O,=I).

  By definition of Left and (=V+O),
    Left ∈ NI(=I,=I+O).
    Left : I -> {Left i | i ∈ I}

  Since
    Left ∈ NI(=I,=I+O),
    mapI left (maybe p)
    ∈ IProc (Either I O) I,
    ∈ NI(=I+O,=I)
  we get by Corollary (mapO-compose) that
    mapO Left (mapI left (maybe p))
    ∈ IProc (Either I O) (Either I O),
    ∈ IProc (Either I O) {Left i | i ∈ I}
    ∈ NI(=I+O,=I+O).

  By analogous reasoning,
    mapO Right (mapI left (maybe q))
    ∈ IProc (Either I O) (Either I O),
    ∈ IProc (Either I O) {Right o | o ∈ O}
    ∈ NI(=I+O,=I+O).
```

```
Let
  pU = mapO Left  (mapI left (maybe p))
  qU = mapO Right (mapI left (maybe q)).

Let
  (=(I+O)^2) = eqpair●LR(=I+O,=I+O)

Since
  pU
  ∈ IProc (Either I O) (Either I O)
  ∈ NI(=I+O,=I+O), and
  qU
  ∈ IProc (Either I O) (Either I O)
  ∈ NI(=I+O,=I+O), and
we get
  uni pU qU
  ∈ IProc (Either I O) (Either I O)^2
  ∈ IProc (Either I O) {Left i | i ∈ I}*{Right o | o ∈ O}, and
  ∈ NI(=I+O,=(I+O)^2).

By definition of Left and (=I+O),
  Left ∈ NI(=I,=I+O).

By definition of Left, (=I+O) and (=I),
  Left ∈ PS(=I,=I+O).

Since
  Left ∈ NI(=I,=I+O),
  Left ∈ PS(=I,=I+O),
  uni pU qU
  ∈ IProc (Either I O) (Either I O)^2
  ∈ IProc (Either I O) {Left i | i ∈ I}*{Right o | o ∈ O}, and
  ∈ NI(=I+O,=(I+O)^2).
we get by Corollary (mapI-compose) that
  mapI Left (uni pU qU)
  ∈ IProc I (Either I O)^2
  ∈ IProc I {Left i | i ∈ I}*{Right o | o ∈ O}, and
  ∈ NI(=I,=(I+O)^2).

By definition of snd, (=(I+O)^2) and (=I+O),
we get
  snd ∈ NI(=(I+O)^2,=I+O).

Since
  snd ∈ NI(=(I+O)^2,=I+O), and
  mapI Left (uni pU qU)
  ∈ IProc I (Either I O)^2
  ∈ IProc I {Left i | i ∈ I}*{Right o | o ∈ O}, and
  ∈ NI(=I,=(I+O)^2).
we get
  mapO snd (mapI Left (uni pU qU))
  ∈ IProc I (Either I O)
  ∈ IProc I {Right o | o ∈ O}, and
  ∈ NI(=I,=I+O).

For
  fromRight : O -> {Right o | o ∈ O} -> O,
we get by definition of fromRight, (=I+O), and (=O) that
  fromRight o
  : {Right o | o ∈ O} -> O
  ∈ NI(=I+O,=O).

Since
  fromRight o
  : {Right o | o ∈ O} -> O
  ∈ NI(=I+O,=O),
and
  mapO snd (mapI Left (uni pU qU))
  ∈ IProc I (Either I O)
```

```
    ∈ IProc I {Right o | o ∈ O}, and
    ∈ NI(=I,=I+O),
  we get by Corollary (mapO-compose) that
    mapO (fromRight o) (
      mapO snd (mapI Left (uni pU qU))
    )
    ∈ IProc I O,
    ∈ NI(=I,=O).

  By the above, we get by definition of cut that
    cut p q
    ∈ IProc I O,
    ∈ NI(=I,=O).

Qed.


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

Implementation of "in".

  in p q = source p , IF O is uninhabited. OTHERWISE, for some o ∈ O,

  in p q =
    mapO (fromRight o) (
      mapO fst (
        mapI Left (
          uni (
            mapO Right(
              mapI left  (
                maybe p
              )
            )
          )(
            mapO Left
              mapI right (
                maybe q
              )
            )
          )
        )
      )
    )

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

Corollary (in-compose)

  forall

    p ∈ IProc I O ,
    q ∈ IProc O I ,
    (=I), (=O),

  if

    p ∈ NI(=I,=O),
    q ∈ NI(=O,=1),

  then

    (in p q) ∈ NI(=I,=O).

Proof.

  Let
    (=mI) = eqmaybe(=I).

  Since
```

```
  (=mI) = eqmaybe(=I), and
  p
  ∈ IProc I O
  ∈ NI(=I,=O)
we get by Theorem (maybe-compose) that
  maybe p
  ∈ IProc (Maybe I) O
  ∈ NI(=mI,=O).

Let
  (=I+O) = eqeither(=I,=O).

By definition of left and (=I+O),
  left ∈ NI(=I+O,=mI).

By definition of left, (=I+O) and (=mI),
  left ∈ PS(=I+O,=mI).

Since
  left ∈ NI(=I+O,=mI),
  left ∈ PS(=I+O,=mI), and
  maybe p
  ∈ IProc (Maybe I) O.
  ∈ NI(=mI,=O)
we get by Corollary (mapI-compose) that
  mapI left (maybe p)
  ∈ IProc (Either I O) O.
  ∈ NI(=I+O,=O).

By definition of Right and (=I+O),
  Right ∈ NI(=O,=I+O).
  Right : O -> {Right o | o ∈ O}

Since
  Right ∈ NI(=O,=I+O),
  mapI left (maybe p)
  ∈ IProc (Either I O) O,
  ∈ NI(=I+O,=O)
we get by Corollary (mapO-compose) that
  mapO Right (mapI left (maybe p))
  ∈ IProc (Either I O) (Either I O),
  ∈ IProc (Either I O) {Right o | o ∈ O}
  ∈ NI(=I+O,=I+O).

By analogous reasoning,
  mapO Left (mapI right (maybe p))
  ∈ IProc (Either I O) (Either I O),
  ∈ IProc (Either I O) {Left i | i ∈ I}
  ∈ NI(=I+O,=I+O).

Let
  pU = mapO Right (mapI left  (maybe p))
  qU = mapO Left  (mapI right (maybe q)).

Let
  (=(I+O)^2) = eqpair●LR(=I+O,=I+O)

Since
  pU
  ∈ IProc (Either I O) (Either I O)
  ∈ NI(=I+O,=I+O), and
  qU
  ∈ IProc (Either I O) (Either I O)
  ∈ NI(=I+O,=I+O), and
we get
  uni pU qU
  ∈ IProc (Either I O) (Either I O)^2
  ∈ IProc (Either I O) {Right o | o ∈ O}*{Left i | i ∈ I}, and
  ∈ NI(=I+O,=(I+O)^2).
```

```
  By definition of Left and (=I+O),
    Left ∈ NI(=I,=I+O).

  By definition of Left, (=I+O) and (=I),
    Left ∈ PS(=I,=I+O).

  Since
    Left ∈ NI(=I,=I+O),
    Left ∈ PS(=I,=I+O),
    uni pU qU
    ∈ IProc (Either I O) (Either I O)^2
    ∈ IProc (Either I O) {Right o | o ∈ O}*{Left i | i ∈ I}, and
    ∈ NI(=I+O,=(I+O)^2).
  we get by Corollary (mapI-compose) that
    mapI Left (uni pU qU)
    ∈ IProc I (Either I O)^2
    ∈ IProc I {Right o | o ∈ O}*{Left i | i ∈ I}, and
    ∈ NI(=I,=(I+O)^2).

  By definition of fst, (=(I+O)^2) and (=I+O),
  we get
    snd ∈ NI(=(I+O)^2,=I+O).

  Since
    snd ∈ NI(=(I+O)^2,=I+O), and
    mapI Left (uni pU qU)
    ∈ IProc I (Either I O)^2
    ∈ IProc I {Right o | o ∈ O}*{Left i | i ∈ I}, and
    ∈ NI(=I,=(I+O)^2).
  we get
    mapO fst (mapI Left (uni pU qU))
    ∈ IProc I (Either I O)
    ∈ IProc I {Right o | o ∈ O}, and
    ∈ NI(=I,=I+O).

  For
    fromRight : O -> {Right o | o ∈ O} -> O,
  we get by definition of fromRight, (=I+O), and (=O) that
    fromRight o
    : {Right o | o ∈ O} -> O
    ∈ NI(=I+O,=O).

  Since
    fromRight o
    : {Right o | o ∈ O} -> O
    ∈ NI(=I+O,=O),
  and
    mapO fst (mapI Left (uni pU qU))
    ∈ IProc I (Either I O)
    ∈ IProc I {Right o | o ∈ O}, and
    ∈ NI(=I,=I+O),
  we get by Corollary (mapO-compose) that
    mapO (fromRight o) (
      mapO fst (mapI Left (uni pU qU))
    )
    ∈ IProc I O,
    ∈ NI(=I,=O).

  By the above, we get by definition of cut that
    in p q
    ∈ IProc I O,
    ∈ NI(=I,=O).

Qed.

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