```
Map
--------------------------------------------------------------------------------

Definition (noninterfering f)

  forall

    f : I -> O,
    (=I), (=O),

  f is (=I,=O)-noninterfering,
  f ∊ NI(=I,=O),
  iff

    forall l . forall i, i' . i =Il i' => (f i) =Ol (f i').

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


Definition (silence-preserving f)

  forall

    f : I -> O,
    (=I), (=O),

  f is (=I,=O)-silence-preserving,
  f ∊ PS(=I,=O),
  iff

    forall l . forall i . i =Il ● => (f i) =Ol ●.

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


Theorem (map-compose):

  forall

    p ∊ IProc I' O ,
    f : I -> I',
    g : O -> O',
    (=I), (=I'), (=O), (=O'),

  if

    p ∊ NI(=I',=O)
    f ∊ NI(=I,=I') ∩ PS(=I,=I'), and,
    g ∊ NI(=I,=I')

  then

    (map f g p) ∊ NI(=I,=O').

Proof.

  Pick p0, f, g, (=I), (=I'), (=O), (=O') satisfying the above assumptions.

  (note: p0 is p in the above theorem statement.
         calling it p0 here eases notation throughout the proof).

  Pick s0 such that
```

  (map f g p0) --s0-▶.

Pick l.

***

To show: there exists a relation R such that
  ⟨s0,map f g p0⟩ ∊ R
and
  R is a l-(=I)-(=O')-simulation.

***

Pick
  R = { ⟨s,map f g p⟩ | exists sP . (sP ≼l p) AND ((map f g sP) --s--▶) }.

(here, ≼ is a shorthand for ≼(=I')(=O))

***

To prove:
  ⟨s0, map f g p0⟩ ∊ R

(we'll prove that R is a simulation in a moment).

Set
  s = s0,
  p = p0,
and construct
  sP
such that
  (map f g sP) --s--▶.
from the proof of the derivation of
  (map f g p0) --s0--▶

Then
  ⟨s,map f g p⟩ ∊ R.

Thus,
  ⟨s0,map f g p0⟩ ∊ R.

***

To prove:
  R is a l-(=I)-(=O')-simulation.

We prove that
R satisfies pt. 1) through 4) of Def IV.2.

case 1):

  Pick
    ⟨?i.s,(map f g p)⟩ ∊ R
  such that
    i =Il ●.

  To show:
    ⟨s,(map f g p)⟩ ∊ R.

  Since
    ⟨?i.s,(map f g p)⟩ ∊ R,
  we have for some sP that
    sP ≼l p, and
    (map f g sP) --?i.s--▶.

  Since

```
    (map f g sP) --?i.s--►,
  we get by definition of map that,
  for some sP',
    sP = ?(f i).sP', and
    (map f g sP') --s--►.

  Since
    f ∊ NI(=I,=I') ∩ PS(=I,=I'),
  we get
    f ∊ PS(=I,=I').

  Since
    f ∊ PS(=I,=I'), and
    i =Il ●,
  we get
    (f i) =I'l ●.

  Since
    sP ≼l p, and
    (f i) =I'l ●,
  we get by Def IV.2 1) that
    sP' ≼l p.

  Since
    (map f g sP') --s--►, and
    sP' ≼l p,
  we get by definition of R that
    ⟨s',(map f g p)⟩ ∊ R.

case 2):

  Pick
    ⟨s,(map f g p)⟩ ∊ R.

  To show:
  forall
    i =Il ●,
  there exists
    pM'
  such that
    (map f g p) ~~i~~► pM', and
    ⟨s,pM'⟩ ∊ R.

  Pick
    i =Il ●.

  Since
    ⟨s,(map f g p)⟩ ∊ R,
  we have for some sP that
    sP ≼l p, and
    (map f g sP) --s--►.

  Since
    f ∊ NI(=I,=I') ∩ PS(=I,=I'),
  we get
    f ∊ PS(=I,=I').

  Since
    f ∊ PS(=I,=I'), and
    i =Il ●,
  we get
    (f i) =I'l ●.

  Since
    sP ≼l p, and
```

```
    (f i) =I'l ●,
  we get by Def IV.2 2) that
  there exists
    p'
  such that
    p ~~(f i)~~▶ p', and
    sP ≼l p'.

  Let
    pM' = (map f g p').
  Then
    (map f g p) ~~i~~▶ pM'.

  Since
    (map f g sP) --s--▶,
    sP ≼l p', and
    (map f g p) ~~i~~▶ pM',
  we get by definition of R that
    ⟨s,pM'⟩ ∊ R.

case 3):

  Pick
    ⟨?i.s,(map f g p)⟩ ∊ R.

  To show:
  forall
    i' =Il i,
  there exists
    pM'
  such that
    (map f g p) ~~i'~~▶ pM', and
    ⟨s,pM'⟩ ∊ R.

  Pick
    i' =Il i.

  Since
    ⟨?i.s,(map f g p)⟩ ∊ R,
  we have for some sP that
    sP ≼l p, and
    (map f g sP) --?i.s--▶.

  Since
    (map f g sP) --?i.s--▶,
  we get by definition of map that,
  for some sP',
    sP = ?(f i).sP', and
    (map f g sP') --s--▶.

  Since
    f ∊ NI(=I,=I') ∩ PS(=I,=I'),
  we get
    f ∊ NI(=I,=I').

  Since
    f ∊ NI(=I,=I'), and
    i' =I'l i,
  we get
    (f i') =I'l (f i).

  Since
    sP ≼l p,
    sP = ?(f i).sP', and
    (f i') =I'l (f i),
```

```
  we get by Def IV.2 3) that
  there exists
    p'
  such that
    p ~~(f i')~~▶ p', and
    sP' ≼l p'.

  Let
    pM' = (map f g p').
  Then
    (map f g p) ~~i'~~▶ pM'.

  Since
    (map f g sP') --s--▶,
    sP' ≼l p', and
    (map f g p) ~~i'~~▶ pM'.
  we get by definition of R that
    ⟨s,pM'⟩ ∈ R.

case 4):

  Pick
    ⟨!ó.s,(map f g p)⟩ ∈ R.

  To show:
  exists
    ó' =Il ó,
  and
    pM'
  such that
    (map f g p) ——ó'—▶ pM', and
    ⟨s,pM'⟩ ∈ R.

  Since
    ⟨!ó.s,(map f g p)⟩ ∈ R,
  we have for some sP that
    sP ≼l p, and
    (map f g sP) --!ó.s--▶.

  Since
    (map f g sP) --!ó.s--▶,
  we get by definition of map that,
  for some o and sP',
    ó = g o,
    sP = !o.sP', and
    (map f g sP') --s--▶.

  Since
    sP ≼l p, and
    sP = !o.sP',
  we get by Def IV.2 4) that
  there exist
    o' =Ol o, and
    p',
  such that
    p ——o'——▶ p', and
    sP' ≼l p'.

  Since
    g ∈ NI(=O,=O'), and
    o' =I'l o,
  we get
    (g o') =O'l (g o).

  Let
```

```
      pM' = (map f g p').
    Then
      (map f g p) ——(g o')——► pM'.

    Since
      (map f g sP') --s--►,
      sP' ≼l p',
      (g o') =O'l (g o),
      pM' = (map f g p'), and
      (map f g p) ——(g o')——► pM',
    we get by definition of R that
      ⟨s,pM'⟩ ∊ R.

  Thus
    R is a l-(=I)-(=O')-simulation.

  Thus,
  forall l,
  exists an l-(=I)-(=O')-simulation R such that
    ⟨s0,map f g p0⟩ ∊ R.

  Thus
    (map f g p0) ∊ NI(=I,=O').

Qed.

Sta
--------------------------------------------------------------------------------

Definition (noninterfering f)

  forall

    f : I -> V -> O,
    (=I), (=V), (=O),

  f is (=I,=V,=O)-noninterfering,
  f ∊ NI(=I,=V,=O),
  iff

    forall l .
      forall i, i' . i =Il i' =>
      forall v, v' . v =Vl v' =>
      (f i v) =Ol (f i' v').

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


Definition (equivalence-preserving f)

  forall

    f : I -> V -> V,
    (=I), (=V)

  f is (=I,=V)-equivalence-preserving,
  f ∊ PE(=I,=V),
  iff

    forall l .
      forall i . i =Il ● =>
      forall v .
      (f v) =Vl v.

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

```
Let

  eqpair'(=A,=B)   l = { ⟨⟨a,b⟩,⟨a',b'⟩⟩ | a =Al a' ∧ b =Bl b' }
  eqpair'●L(=A,=B) l = { ⟨●,⟨a,b⟩⟩ | a =Al ● }
  eqpair'●R(=A,=B) l = { ⟨⟨a,b⟩,●⟩ | b =Bl ● }
  eqpair'●LR(=A,=B)l = { ⟨⟨a,b⟩,●⟩ | a =Al ● ∧ b =Bl ● }

  RTC(R) is the reflexive transitive closure of R.

  eqpair(=A,=B)    l = RTC(eqpair'(=A,=B) l)
  eqpair●L(=A,=B)  l = RTC(eqpair'(=A,=B) l ∪ eqpair'●L(=A,=B) l)
  eqpair●R(=A,=B)  l = RTC(eqpair'(=A,=B) l ∪ eqpair'●R(=A,=B) l)
  eqpair●LR(=A,=B) l = RTC(eqpair'(=A,=B) l ∪ eqpair'●LR(=A,=B) l)
  eqpair●(=A,=B)   l = RTC(eqpair'(=A,=B) l ∪ eqpair'●L(=A,=B) l ∪ eqpair'●R(=A,=B)
l)

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


Theorem (sta-compose):

  forall

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

  if

    p ∊ NI(=V*I,=O)
    f ∊ NI(=I,=V,=V) ∩ PE(=I,=V), and
    g ∊ NI(=O,=V,=V)

  then forall v,

    (sta f g v p) ∊ NI(=I,=V*O),

  where

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

Proof.

  Pick p0, v0, f, (=I), (=V), (=O), satisfying the above assumptions.

  (note: p0 is p in the above theorem statement.
         calling it p0 here eases notation throughout the proof).

  Pick s0 such that
    sta f g v0 p0 --s0-▶.

  Pick l.

  Let (=V*I) = eqpair●R(=V,=I).

  ***

  To show: there exists a relation R such that
    ⟨s0,sta f g v0 p0⟩ ∊ R
  and
    R is a l-(=V*I)-(=O)-simulation.
```

***

Pick
R = { ⟨s,sta f g v p⟩ | exists sP, vS . (sP ≼l p) AND (vS =Vl v) AND (sta f g vS sP --s--►) }.

(here, ≼ is a shorthand for ≼(=V*I)(=O))

***

To prove:
⟨s0, sta f g v p0⟩ ∈ R

(we'll prove that R is a simulation in a moment).

Set
s = s0,
p = p0,
v = v0,
and construct
sP
such that
sta f g v sP --s--►
from the proof of the derivation of
sta f g v0 p0 --s0--►.

Then
⟨s,sta f g v p⟩ ∈ R.

Thus,
⟨s0,sta f g v0 p0⟩ ∈ R.

***

To prove:
R is a l-(=V*I)-(=O)-simulation.

We prove that
R satisfies pt. 1) through 4) of Def IV.2.

case 1):

Pick
⟨?i.s,(sta f g v p)⟩ ∈ R
such that
i =Il ●.

To show:
⟨s,(sta f g v p)⟩ ∈ R.

Since
⟨?i.s,(sta f g v p)⟩ ∈ R
we have for some sP and vS =Vl v that
sP ≼l p, and
sta f g vS sP --?i.s--►.

Since
sta f g vS sP --?i.s--►,
we get by definition of staI that,
for some sP',
sP = ?⟨(f i vS),i⟩.sP', and
sta f g (f i vS) sP' --s--►.

Since
i =Il ●,
we get by definition of (=V*I) that

```
    ⟨(f i vS),i⟩ =V*Il ●.

  Since
    sP ≼l p, and
    ⟨(f i vS),i⟩ =V*Il ●.
  we get by Def IV.2 1) that
    sP' ≼l p.

  Since
    f ∊ NI(=I,=V,=V) ∩ PE(=I,=V),
  we get
    f ∊ PE(=I,=V).

  Since
    f ∊ PE(=I,=V), and
    i =Il ●,
  we get
    (f i vS) =Vl vS.

  Since
    vS =Vl v, and
    (f i vS) =Vl vS.
  we get by transitivity of (=Vl) that
    (f i vS) =Vl v.

  Since
    sta f g (f i vS) sP' --s--►,
    sP' ≼l p, and
    (f i vS) =Vl v,
  we get by definition of R that
    ⟨s',(sta f g v p)⟩ ∊ R.

case 2):

  Pick
    ⟨s,(sta f g v p)⟩ ∊ R.

  To show:
  forall
    i =Il ●,
  there exists
    pS'
  such that
    (sta f g v p) ~~i~~► pS', and
    ⟨s,pS'⟩ ∊ R.

  Pick
    i =Il ●.

  Since
    ⟨s,(sta f g v p)⟩ ∊ R,
  we have for some sP and vS =Vl v that
    sP ≼l p, and
    sta f g vS sP --s--►.

  Since
    i =Il ●,
  we get by definition of (=V*I) that
    ⟨(f i v),i⟩ =V*Il ●.

  Since
    sP ≼l p, and
    ⟨(f i v),i⟩ =V*Il ●.
  we get by Def IV.2 2) that
  there exists
```

```
    p'
  such that
    p ~~⟨(f i v),i⟩~~► p', and
    sP ≼l p'.

  Since
    f ∊ NI(=I,=V,=V) ∩ PE(=I,=V),
  we get
    f ∊ PE(=I,=V).

  Since
    f ∊ PE(=I,=V),
  we get
    (f i v) =Vl v.

  Since
    (f i v) =Vl v, and
    v =Vl vS,
  we get by transitivity of (=V) that
    (f i v) =Vl vS.

  Let
    pS' = (sta f g (f i v) p').
  Then
    (sta f g v p) ~~i~~► pS'.

  Since
    sta f g vS sP --s--►,
    sP ≼l p', and
    (f i v) =Vl vS.
  we get by definition of R that
    ⟨s,(sta f g (f i v) p')⟩ ∊ R.

case 3):

  Pick
    ⟨?i.s,(sta f g v p)⟩ ∊ R.

  To show:
  forall
    i' =Il i,
  there exists
    pS'
  such that
    (sta f g v p) ~~i'~~► pS', and
    ⟨s,pS'⟩ ∊ R.

  Since
    ⟨?i.s,(sta f g v p)⟩ ∊ R,
  we have for some sP and vS =Vl v that
    sP ≼l p, and
    sta f g vS sP --?i.s--►.

  Since
    sta f g vS sP --?i.s--►,
  we get by definition of staI that,
  for some sP',
    sP = ?⟨(f i vS),i⟩.sP', and
    sta f g (f i vS) sP' --s--►.

  Since
    f ∊ NI(=I,=V,=V) ∩ PS(=I,=V),
  we get
    f ∊ NI(=I,=V,=V).
```

```
  Since
    f ∊ NI(=I,=V,=V),
    i' =Il i, and
    v =Vl vS,
  we get
    (f i vS) =Vl (f i' v).

  Since
    i' =Il i, and
    (f i vS) =Vl (f i' v),
  we get by definition of (=V*I) that
    ⟨(f i vS),i⟩ =V*I ⟨(f i' v),i'⟩.

  Since
    sP ≼l p,
    sP = ?⟨(f i vS),i⟩.sP', and
    ⟨(f i vS),i⟩ =V*I ⟨(f i' v),i'⟩,
  we get by Def IV.2 3) that
  there exists
    p'
  such that
    p ~~⟨(f i' v),i'⟩~~► p', and
    sP' ≼l p'.

  Let
    pS' = (sta f g (f i' v) p').
  Then
    (sta f g v p) ~~i'~~► pS'.

  Since
    sta f g (f i vS) sP' --s--►,
    sP' ≼l p',
    (f i vS) =Vl (f i' v),
    pS' = (sta f g (f i' v) p'), and
    (sta f g v p) ~~i'~~► pS',
  we get by definition of R that
    ⟨s,pS'⟩ ∊ R.

case 4):

  Pick
    ⟨!⟨vO,o⟩.s,(sta f g v p)⟩ ∊ R.

  To show:
  exists
    ⟨vO',o'⟩ =V*Ol ⟨vO,o⟩,
  and
    pS'
  such that
    (sta f g v p) ——⟨vO',o'⟩—► pS', and
    ⟨s,pS'⟩ ∊ R.

  Since
    ⟨!⟨vO,o⟩.s,(sta f g v p)⟩ ∊ R,
  we have for some sP and vS =Vl v that
    sP ≼l p, and
    sta f g vS sP --!⟨vO,o⟩.s--►.

  Since
    sta f g vS sP --!⟨vO,o⟩.s--►,
  we get by definition of sta that
    vO = g o vS,
  and for some sP',
    sP = !o.sP', and
    sta f g vS sP' --s--►.
```

```
    Since
      sP ≼l p, and
      sP = !o.sP',
    we get by Def IV.2 4) that
    there exist
      oP =Ol o, and
      p',
    such that
      p ——oP——► p', and
      sP' ≼l p'.

    Let
      o'  = oP, and
      vO' = g o' v.

    Since
      oP =Ol o, and
      oP =   o',
    we get by transitivity of (=Ol) that
      o' =Ol o.

    Since
      vO = g o vS,
      vO'= g o' v,
      vS =Vl v, and
      g  ∊ NI(=O,=V,=V),
    we get
      vO'=Vl vO.

    Since
      o' =Ol o, and
      vO'=Vl vO,
    we get by definition of (=V*O) that
      ⟨vO',o'⟩ =V*Ol ⟨vO,o⟩.

    Since
      p ——oP——► p', and
      o' = oP,
    we get
      p ——o'——► p'.

    Let
      pS' = (sta f g v p').
    Then, since
      p ——o'——► p', and
      vO' = g o' v.
    we get
      (sta f g v p) ——⟨vO',o'⟩——► pS'.

    Since
      sta f g v sP' --s--►,
      sP' ≼l p',
      ⟨vO',o'⟩ =V*Ol ⟨vO,o⟩,
      pS' = (sta f g v p'), and
      (sta f g v p) ——⟨vO',o'⟩——► pS',
    we get by definition of R that
      ⟨s,pS'⟩ ∊ R.

  Thus
    R is a l-(=I)-(=V*O)-simulation.

  Thus,
  forall l,
  exists an l-(=I)-(=V*O)-simulation R such that
    ⟨s0,sta f g v0 p0⟩ ∊ R.
```

```
  Thus
    (sta f g v0 p0) ∊ NI(=I,=V*O).

Qed.

Swi
--------------------------------------------------------------------------------

Definition (oblivious observers)

  forall

    (=V),

  l is oblivious to v under (=V),
  O(v,=V),
  iff
    v =V ●.

  l is oblivious under (=V,
  O(=V),
  iff
    forall v . O(v,=V).

End Definition

Definition (fully aware observers)

  forall

    (=X),

  l is aware of x under (=X),
  A(x,=X),
  iff
    forall ẋ . x =Xl ẋ => x = ẋ.

  l is aware under (=X),
  A(=X),
  iff
    forall x . A(x,=X).

Definition

Remark

  While obliviousness and awareness are mutually exclusive, the
  negation of one does not imply the other. (An observer may be able
  to distinguish one value from another (thus not being oblivious to
  it), without observing it fully (thus not being fully aware of it)).

End Remark

Definition (oblivious to a process)

  forall

    p ∊ IProc I O,
    (=O),

  l is oblivious to p under (=O), l ∊ O(p,=O), iff
    forall i . p ~~i~▶ p' => l ∊ O(p',=O), and
    forall o . p ——o—▶ p' => l ∊ O(p',=O) ∧ o =Ol ●.

End Definition
```

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

Let

  eqmaybe'(=V) l = { ⟨Just v,Just v'⟩ | v =Vl v' } ∪ { ⟨Just v,●⟩ | v =Vl ● }
  eqmaybe'(L)  l | l ∈ L     = ∅
                 | otherwise = { ⟨Nothing,●⟩ }

  eqmaybe(L,=V) l = RTC(eqmaybe'(=V) l ∪ eqmaybe'(L))

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

Theorem (swi-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,

    (swiI 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.

  Pick p0, b0, (=I), (=O), (=Bool), satisfying the above assumptions.

  (note: p0 is p in the above theorem statement.
         calling it p0 here eases notation throughout the proof).

  Pick s0 such that

    (swi b0 p0) --s0-►.

  Pick l.

  Let

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

  ***

  To show: there exists a relation R such that
    ⟨s0,swi b0 p0⟩ ∈ R, and
    R is a l-(=Bool*I)-(=MaybeO)-simulation.

  ***

  Two cases to consider for l.

  Case l ∉ A(True,=Bool) :
```

```
Pick
  R = { ⟨s,swi b p⟩ | s ∊ Stream (Bool*I) ((=MaybeO)l ●) }.

***

To prove:
  ⟨s0,swi b0 p0⟩ ∊ R.

Since
  l ∉ A(True,=Bool),
we get by definition of (=MaybeO) that
  Nothing  (=MaybeO)l ●,
and, forall o =Ol ●,
  (Just o) (=MaybeO)l ●.

Since
  l ∉ A(True,=Bool),
we get
  l ∊ O(p,=Bool*O).

Since
  l ∊ O(p,=Bool*O),
  (Just o) (=MaybeO)l ● , forall o =Ol ●, and
  Nothing  (=MaybeO)l ●,
we get by definition of (=Bool*O) and (=MaybeO) that
  s0 ∊ Stream (Bool*I) ((=MaybeO)l ●).

Set
  s  = s0,
  b  = b0,
  p  = p0.

Then
  ⟨s,swi b p⟩ ∊ R.

Thus,
  ⟨s0,swi b0 p0⟩ ∊ R.

***

To prove:

  R is a l-(=Bool*I)-(=MaybeO)-simulation.

We prove that
R satisfies pt. 1) through 4) of Def IV.2.

case 1):

  Pick
    ⟨?⟨bI,i⟩.s,(swi b p)⟩ ∊ R
  such that
    ⟨bI,i⟩ =Il ●.

  To show:
    ⟨s,(swi b p)⟩ ∊ R.

  Since
    ?⟨bI,i⟩.s ∊ Stream (Bool*I) ((=MaybeO)l ●),
  we get
    s ∊ Stream (Bool*I) ((=MaybeO)l ●).

  Since
    s ∊ Stream (Bool*I) ((=MaybeO)l ●),
```

```
  we get by definition of R that
    ⟨s,(swi b p)⟩ ∊ R.

Case 2):

  Pick
    ⟨s,(swi b p)⟩ ∊ R.

  To show:
  forall
    ⟨b,i⟩ (=Bool*I)l ●,
  there exists
    pS'
  such that
    (swi b p) ~~⟨b,i⟩~~▶ pS', and
    ⟨s,pS'⟩ ∊ R.

  Pick
    ⟨b,i⟩ (=Bool*I)l ●.

  Since p is interactive,
  we get by rule (Swi-In) that
  there exists a b', p' such that
    (swi b p) ~~⟨b,i⟩~▶ (swi b' p').

  Let
    pS' = (swi b' p').
  Then
    (swi b p) ~~⟨b,i⟩~▶ pS'.

  Since
    s ∊ Stream (Bool*I) ((=MaybeO)l ●),
    pS' = (swi b' p'),
    (swi b p) ~~⟨b,i⟩~▶ pS', and
    ⟨b,i⟩ (=Bool*I)l ●,
  we get by definition of R that
    ⟨s,pS'⟩ ∊ R.

Case 3):

  Pick
    ⟨?⟨b,i⟩.s,(swi b p)⟩ ∊ R.

  To show:
  forall
    ⟨b',i'⟩ (=Bool*I)l ⟨b,i⟩,
  there exists
    pS'
  such that
    (swi b p) ~~⟨b',i'⟩~~▶ pS', and
    ⟨s,pS'⟩ ∊ R.

  Pick
    ⟨b',i'⟩ (=Bool*I)l ⟨b,i⟩.

  Since p is interactive,
  we get by rule (Swi-In) that
  there exists a b', p' such that
    (swi b p) ~~⟨b',i'⟩~▶ (swi b' p').

  Let
    pS' = (swi b' p').
  Then
    (swi b p) ~~⟨b',i'⟩~▶ pS'.
```

```
  Since
    s ∊ Stream (Bool*I) ((=MaybeO)l ●),
    pS' = (swi b' p'),
    (swi b p) ~~⟨b',i'⟩~▶ pS', and
    ⟨b',i'⟩ (=Bool*I)l ⟨b,i⟩,
  we get by definition of R that
    ⟨s,pS'⟩ ∊ R.

Case 4):

  Let
    X = Maybe O.

  Pick
    ⟨!x.s,(swi b p)⟩ ∊ R.
  To show:
  exists
    x' (=MaybeO)l x,
  and
    pS'
  such that
    (swi b p) ——x'—▶ pS', and
    ⟨s,pS'⟩ ∊ R.

  By definition of R,
    x (=MaybeO)l ●.

  Case on b.

  Case b=True:

    Since
      l ∊ O(p,=Bool*O),
    and since p is interactive,
    we get that there exists some
      ⟨b',o'⟩ (=Bool*O)l ●
    such that
      p ——⟨b',o'⟩—▶ p'.

    Since
      ⟨b',o'⟩ (=Bool*O)l ●,
    we get by definition of (=Bool*O) that
      o' =Ol ●.

    Since
      p ——⟨b',o'⟩—▶ p',
    we get by rule (Swi-Out) that
      (swi b p) ——Just o'—▶ (swi (b ⊕ b') p').

    Since
      ⟨b',o'⟩ (=Bool*O)l ●,
    we get by definition of (=Bool*O) that
      o' =Ol ●.

    Since
      o' =Ol ●,
    we get by definition of (=MaybeO) that
      Just o' (=MaybeO)l ●.

    Let
      x' = Just o'.

    Since
      x' = Just o'.
      Just o' (=MaybeO)l ●,
      x (=MaybeO)l ●.
```

```
      we get by transitivity that
        x (=MaybeO)l x'.

      Let
        pS' = (swi (b ⊕ b') p').
      Then
        (swi b p) ——x'—▶ pS'.

      Since
        s ∊ Stream (Bool*I) ((=MaybeO)l ●),
        pS' = (swi (b ⊕ b') p'),
        (swi b p) ——x'—▶ pS', and
        x (=MaybeO)l x'.
      we get by definition of R that
        ⟨s,pS'⟩ ∊ R.

    Case b=False:

      we get by rule (Swi-_Out●) that
        (swi b p) ——Nothing—▶ (swi b p).

      Since
        l ∉ A(True,=Bool),
      we get by definition of (=MaybeO) that
        Nothing (=MaybeO)l ●.

      Let
        x' = Nothing.

      Since
        x' = Nothing,
        Nothing (=MaybeO)l ●,
        x (=MaybeO)l ●.
      we get by transitivity that
        x (=MaybeO)l x'.

      Let
        pS' = (swi b p).
      Then
        (swi b p) ——x'—▶ pS'.

      Since
        s ∊ Stream (Bool*I) ((=MaybeO)l ●),
        pS' = (swi b p),
        (swi b p) ——x'—▶ pS', and
        x (=MaybeO)l x'.
      we get by definition of R that
        ⟨s,pS'⟩ ∊ R.

Case True ∊ A(l,=Bool) :

  Pick
    R = { ⟨s,swi b p⟩ | exists sP, bS . (sP ≼l p) AND (bS (=Bool)l b) AND (swi bS sP
--s--▶) }.
  (here, ≼ is a shorthand for ≼(=Bool*I)(=MaybeO) )

  ***

  To prove:
    ⟨s0, swi b0 p0⟩ ∊ R
  (we'll prove that R is a simulation in a moment).

  Set
    s = s0,
    p = p0,
```

```
  b = b0,
and construct
  sP
such that
  (swi b sP) --s--▶
from the proof of the derivation of
  (swi b0 p0) --s0--▶.

Then
  ⟨s,swi b p⟩ ∊ R.

Thus,
  ⟨s0,swi b0 p0⟩ ∊ R.

***

To prove:
  R is a l-(=Bool*I)-(=MaybeO)-simulation.

We prove that
R satisfies pt. 1) through 4) of Def IV.2.

case 1):

  Pick
    ⟨?⟨bI,i⟩.s,(swi b p)⟩ ∊ R
  such that
    ⟨bI,i⟩ (=Bool*I)l ●.

  To show:
    ⟨s,(swi b p)⟩ ∊ R.

  Since
    ⟨?⟨bI,i⟩.s,(swi b p)⟩ ∊ R
  we have for some sP and bS (=Bool)l b that
    sP ≼l p, and
    (swi bS sP) --?⟨bI,i⟩.s--▶.

  Since
    (swi bS sP) --?⟨bI,i⟩.s--▶,
  we get by definition of swi that,
  for some sP',
    sP = ?i.sP', and
    (swi (bS ⊕ bI) sP') --s--▶.

  Since
    ⟨bI,i⟩ (=Bool*I)l ●, and
    True ∊ A(l,=Bool),
  we get by definition of (=Bool*I) that
    bI = False.

  Thus, by definition of ⊕,
    b  ⊕ bI = b, and
    bS ⊕ bI = bS.

  Since
    bS ⊕ bI = bS, and
    (swi (bS ⊕ bI) sP') --s--▶.
  we get
    (swi bS sP') --s--▶.

  Since
    ⟨bI,i⟩ (=Bool*I)l ●,
  we get by definition of (=Bool*I) that
    i =Il ●.
```

```
  Since
    sP ⩽l p, and
    sP = ?i.sP', and
    i =Il ●,
  we get by Def IV.2 1) that
    sP' ⩽l p.

  Since
    (swi bS sP') --s--►.
    sP' ⩽l p, and
    bS (=Bool)l b,
  we get by definition of R that
    ⟨s,swi b p⟩ ∈ R.

case 2):

  Pick
    ⟨s,(swi b p)⟩ ∈ R.

  To show:
  forall
    ⟨bI,i⟩ (=Bool*I)l ●
  there exists
    pS'
  such that
    (swi b p) ~~⟨bI,i⟩~~► pS', and
    ⟨s,pS'⟩ ∈ R.

  Since
    ⟨s,(swi b p)⟩ ∈ R,
  we have for some sP and bS (=Bool)l b that
    sP ⩽l p, and
    (swi bS sP) --s--►.

  Pick
    ⟨bI,i⟩ =Il ●.

  Since
    ⟨bI,i⟩ (=Bool*I)l ●,
  we get by definition of (=Bool*I) that
    i =Il ●.

  Since
    sP ⩽l p, and
    i =Il ●,
  we get by Def IV.2 2) that
  there exists
    p'
  such that
    p ~~i~~► p', and
    sP ⩽l p'.

  Since
    ⟨bI,i⟩ (=Bool*I)l ●, and
    True ∈ A(l,=Bool),
  we get by definition of (=Bool*I) that
    bI = False.

  Thus, by definition of ⊕,
    b ⊕ bI = b.

  Since
    p ~~i~~► p', and
    b ⊕ bI = b,
```

```
  we get by (Swi-In) that
    (swi b p) ~~⟨bI,i⟩~~► (swi b p').

  Let
    pS' = (swi b p').
  Then
    (swi b p) ~~⟨bI,i⟩~~► pS'.

  Since
    (swi bS sP) --s--►.
    sP ≼l p',
    bS (=Bool)l b,
    pS' = (swi b p'),
    ⟨bI,i⟩ (=Bool*I)l ●, and
    (swi b p) ~~⟨bI,i⟩~~► pS',
  we get by definition of R that
    ⟨s,pS'⟩ ∈ R.

case 3):

  Pick
    ⟨?⟨bI,i⟩.s,(swi b p)⟩ ∈ R.

  To show:
  forall
    ⟨bI',i'⟩ (=Bool*I)l ⟨bI,i⟩,
  there exists
    pS'
  such that
    (swi b p) ~~⟨bI',i'⟩~~► pS', and
    ⟨s,pS'⟩ ∈ R.

  Since
    ⟨?⟨bI,i⟩.s,(swi b p)⟩ ∈ R,
  we have for some
    sP and
    bS (=Bool*I)l b
  that
    sP ≼l p, and
    (swi bS sP) --?⟨bI,i⟩.s--►.

  Since
    (swi b sP) --?⟨bI,i⟩.s--►,
  we get by definition of swi that,
  for some sP',
    sP = ?i.sP', and
    (swi (bS ⊕ bI) sP') --s--►.

  Pick
    ⟨bI',i'⟩ (=Bool*I)l ⟨bI',i'⟩.

  Since
    ⟨bI',i'⟩ (=Bool*I)l ⟨bI',i'⟩,
  we get by definition of (=Bool*I) that
    bI' (=Bool)l bI, and
    i' =Il i.

  Since
    sP ≼l p,
    sP = ?i.sP', and
    i' =Il i,
  we get by Def IV.2 3) that
  there exists
    p'
  such that
```

```
    p ~~i'~~▶ p', and
    sP' ≼l p'.

  Since
    b   (=Bool)l bS,
    bI' (=Bool)l bI, and
    True ∊ A(l,=Bool),
  we get
    (b ⊕ bI') (=Bool)l (bS ⊕ bI).

  Let
    pS' = (swi (b ⊕ bI') p').
  Then
    (swi b p) ~~⟨bI',i'⟩~~▶ pS'.

  Since
    (swi (bS ⊕ bI) sP') --s--▶,
    sP' ≼l p',
    (b ⊕ bI') (=Bool)l (bS ⊕ bI),
    pS' = (swi (b ⊕ bI') p'),
    (swi b p) ~~⟨bI',i'⟩~~▶ pS', and
    ⟨bI',i'⟩ (=Bool*I)l ⟨bI',i'⟩,
  we get by definition of R that
    ⟨s,pS'⟩ ∊ R.

case 4):

  Let
    Ó = Maybe O.

  Pick
    ⟨!ó.s,(swi b p)⟩ ∊ R.

  To show:
  exists
    ó' (=MaybeO)l ó,
  and
    pS'
  such that
    (swi b p) ——ó'—▶ pS', and
    ⟨s,pS'⟩ ∊ R.

  Since
    ⟨!ó.s,(swi b p)⟩ ∊ R,
  we have for some
    sP and
    bS (=Bool)l b
  that
    sP ≼l p, and
    (swi bS sP) --!ó.s--▶.

  Case on b.

  Case b = False:

    Since
      b = False,
    we get
      (swi b p) ——ó'—▶ (swi b p), and
      ó' = Nothing.

    Since
      True ∊ A(l,=Bool),
      bS (=Bool)l b, and
      b  = False
```

```
  we get
    bS = False.

  Since
    bS = False,
  we get
    (swi bS sP) ——ó—▶ (swi bS sP) --s--▶, and
    ó = Nothing.

  Since
    ó  = Nothing, and
    ó' = Nothing,
  we have
    ó' (=Bool)l ó.

  Let
    pS' = (swi b p).
  Then
    (swi b p) ——ó'—▶ pS'.

  Since
    (swi bS sP) --s--▶,
    sP ≼l p,
    ó' (=Bool)l ó,
    pS' = (swi b p), and
    (swi b p) ——ó'—▶ pS',
  we get by definition of R that
    ⟨s,pS'⟩ ∊ R.

Case b = True:

  Since
    True ∊ A(l,=Bool),
    bS (=Bool)l b, and
    b  = True
  we get
    bS = True.

  Since
    bS = True, and
    (swi bS sP) --!ó.s--▶,
  we get for some o, bO and sP' that
    ó = Just o,
    sP = !⟨bO,o⟩.sP', and
    (swi bS sP) ——ó—▶ (swi (bS ⊕ bO) sP') --s--▶.

  Since
    sP ≼l p, and
    sP = !⟨bO,o⟩.sP',
  we get by Def IV.2 4) that
  there exist
    ⟨bO',o'⟩ (=Bool*O)l ⟨bO,o⟩, and
    p',
  such that
    p ——⟨bO',o'⟩——▶ p', and
    sP' ≼l p'.

  Since
    ⟨bO',o'⟩ (=Bool*O)l ⟨bO,o⟩
  we get by definition of (=Bool*O) that
    bO'(=Bool)l bO, and
    o' =Ol o.

  Let
    ó' = Just o'.
  Then, by definition of (=MaybeO),
```

```
        since
          ó  = Just o, and
          o' =Ol o,
        we get
          ó' (=MaybeO)l ó.

        Since
          b  = True,
          p ——⟨bO',o'⟩——► p', and
          ó' = Just o',
        we get by (Swi-Out) that
          (swi b p) ——ó'—► (swi (b ⊕ bO') p').

        Since
          b  = True,
          bS = True,
          bO'(=Bool)l bO, and
          True ∊ A(l,=Bool),
        we get that
          (bS ⊕ bO) (=Bool)l (b ⊕ bO').

        Let
          pS' = (swi (b ⊕ bO') p').
        Then, since
          (swi b p) ——ó'—► (swi (b ⊕ bO') p'),
        we get
          (swi b p) ——ó'—► pS'.

        Since
          (swi (bS ⊕ bO) sP') --s--►,
          sP' ≼l p',
          ó' (=MaybeO)l ó,
          pS' = (swi (b ⊕ bO') p'),
          (swi b p) ——ó'—► pS', and
          (bS ⊕ bO) (=Bool)l (b ⊕ bO').
        we get by definition of R that
          ⟨s,pS'⟩ ∊ R.

    Thus
      R is a l-(=Bool*I)-(=MaybeO)-simulation.

    Thus,
    forall l,
    exists an l-(=Bool*I)-(=MaybeO)-simulation R such that
      ⟨s0,swi b0 p0⟩ ∊ R.

    Thus
      (swi b0 p0) ∊ NI(=Bool*I,=MaybeO).

Qed.

Maybe
--------------------------------------------------------------------------------

Let

  eqmaybe'    l = { ⟨Nothing,●⟩ }
  eqmaybe(=V) l = RTC(eqmaybe'(=V) l ∪ eqmaybe')

(note the difference between eqmaybe(=V) and eqmaybe(L,=V))

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


Theorem:
```

```
  forall

    p ∊ IProc I O ,
    (=I), (=O),

  if

    p ∊ NI(=I,=O) ,

  then

    (maybe p) ∊ NI(=MaybeI,=I),

  where

    (=MaybeI) = eqmaybe(=I).

Proof.

  Pick p0, (=I), (=O) satisfying the above assumptions.

  Pick s0 such that
    (maybe p0) --s0-►.

  Pick l.

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

  ***

  To show: there exists a relation R such that
    ⟨s0,maybe p0⟩ ∊ R
  and
    R is a l-(=MaybeI)-(=I)-simulation.

  Pick
    R = { ⟨s,maybe p⟩ | exists sP . sP ≼l p AND (maybe sP) --s-► }.

  (here, ≼ is a shorthand for ≼(=MaybeI)(=O) )

  ***

  To prove:
    ⟨s0,maybe p0⟩ ∊ R.

  Set
    s  = s0,
    p  = p0,
  and construct
    sP
  such that
    (maybe sP) --s--►
  from the proof of the derivation of
    (maybe p0) --s0--►.

  Then
    ⟨s,maybe p⟩ ∊ R.

  Thus,
    ⟨s0,maybe p0⟩ ∊ R.

  ***

  To prove:
```

```
  R is a l-(=MaybeI)-(=O)-simulation.

We prove that
R satisfies pt. 1) through 4) of Def IV.2.

Let
  Í = Maybe I.
(note the accent)

case 1)

  Pick
    ⟨?í.s,(maybe p)⟩ ∊ R
  such that
    í (=MaybeI)l ●.

  To show:
    ⟨s,(maybe p)⟩ ∊ R.

  Since
    ⟨s,(maybe p)⟩ ∊ R,
  we get for some sP that
    (maybe sP) --?í.s--► and
    sP ≼l p.

  Case on í.

  Case í = Nothing:

    Since
      (maybe sP) --?í.s--►, and
      í = Nothing,
    we get by definition of (Maybe-In●) that
      (maybe sP) --s--►.

    Since
      (maybe sP) --s--► and
      sP ≼l p,
    we get
      ⟨s,(maybe p)⟩ ∊ R.

  Case í = Just i, for some i:

    Since
      (maybe sP) --?í.s--►,
    we get by definition of (Maybe-In) that,
    for some sP',
      sP = ?i.sP', and
      (maybe sP') --s--►.

    Since
      í (=MaybeI)l ●,
    we get by definition of (=MaybeI) that
      i =Il ●.

    Since
      sP ≼l p,
      sP = ?i.sP', and
      i =Il ●,
    we get by 1) that
      sP' ≼l p.

    Since
      (maybe sP') --s--► and
      sP' ≼l p,
    we get
```

```
      ⟨s,(maybe p)⟩ ∈ R.

  case 2)

    Pick
      ⟨s,(maybe p)⟩ ∈ R.

    To show:
    forall
      í (=MaybeI)l ●,
    it holds that, for some pL',
      (maybe p) ~~í~► pM' and
      ⟨s,pM'⟩ ∈ R.

    Since
      ⟨s,(maybe p)⟩ ∈ R,
    we get
      (maybe sP) --s--► and
      sP ≼l p.

    Pick
      í (=MaybeI)l ●.

    Case on í.

    Case í = Nothing:

      Since
        (maybe sP) --s--►, and
        í = Nothing,
      we get by definition of (Maybe-In●) that
        (maybe sP) ~~í~~► (maybe sP).

      Since
        (maybe sP) ~~í~~► (maybe sP), and
        (maybe sP) --s--►,
      we get
        (maybe sP) --?í.s--►.

      Since
        í = Nothing,
      we get by definition of (Maybe-In●) that
        (maybe p) ~~í~► (maybe p).

      Let
        pM' = (maybe p).
      Then
        (maybe p) ~~í~► pM'.

      Since
        (maybe sP) --?í.s--►,
        sP ≼l p,
        pM' = (maybe p),
        (maybe p) ~~í~► pM', and
        í (=MaybeI)l ●,
      we get
        ⟨?s,pM'⟩ ∈ R.

    Case í = Just i, for some i:

      Since
        í (=MaybeI)l ●,
      we get by definition of (=MaybeI) that
        i =Il ●.
```

```
    Since
      sP ≼l p, and
      i =Il ●,
    we get by 2) for some p' that
      p ~~i~▶ p', and
      sP ≼l p'.

    Since
      p ~~i~▶ p', and
      í = Just i,
    we get by definition of (Maybe-In) that
      (maybe p) ~~í~~▶ (maybe p').

    Set
      pM' = (maybe p').
    Then
      (maybe p) ~~í~▶ pM'.

    Since
      (maybe sP) --s--▶,
      sP ≼l p',
      pM' = (maybe p'),
      (maybe p) ~~í~▶ pM', and
      í (=MaybeI)l ●,
    we get
      ⟨s,pM'⟩ ∊ R.

case 3)

  Pick
    ⟨?í.s,(maybe p)⟩ ∊ R

  To show:
  forall
    í' (=MaybeI)l í,
  it holds that, for some pM',
    (maybe p) ~~í'~~▶ pM' and
    ⟨s,pM'⟩ ∊ R.

  Since
    ⟨?í.s,(maybe p)⟩ ∊ R,
  we get
    (maybe sP) --(?í.s)--▶ and
    sP ≼l p.

  Pick
    í' (=MaybeI)l í.

  Case on ⟨í,í'⟩.

  Case í = Nothing, í' = Nothing:

    Since
      (maybe sP) --?í.s--▶, and
      í = Nothing,
    we get by definition of (Maybe-In●) that
      (maybe sP) ~~í~~▶ (maybe sP).

    Since
      (maybe sP) ~~í~~▶ (maybe sP), and
      (maybe sP) --?í.s--▶,
    we get
      (maybe sP) --s--▶.

    Since
```

```
    í' = Nothing,
  we get by definition of (Maybe-In●) that
    (maybe sP) ~~í'~~▶ (maybe sP).

  Let
    pM' = (maybe p).
  Since
    (maybe sP) ~~í'~~▶ (maybe sP)
  we get
    (maybe sP) ~~í'~~▶ pM'.

  Since
    (maybe sP) --s--▶,
    sP ≼l p,
    pM' = (maybe p),
    (maybe sP) ~~í'~~▶ pM', and
    í' (=MaybeI)l í,
  we get
    ⟨s,pM'⟩ ∈ R.

Case í = Nothing, í' = Just i':

  Since
    (maybe sP) --?í.s--▶, and
    í = Nothing,
  we get by definition of (Maybe-In●) that
    (maybe sP) ~~í~~▶ (maybe sP).

  Since
    (maybe sP) ~~í~~▶ (maybe sP), and
    (maybe sP) --?í.s--▶,
  we get
    (maybe sP) --s--▶.

  By definition of (=MaybeI), we have
    Nothing (=MaybeI)l ●.

  Since
    í' (=MaybeI)l í,
    í = Nothing, and
    Nothing (=MaybeI)l ●,
  we get by transitivity that
    í' (=MaybeI)l ●.

  Since
    í' (=MaybeI)l ●, and
    í' = Just i'
  we get by definition of (=MaybeI) that
    i' =Il ●.

  Since
    sP ≼l p, and
    i' =Il ●,
  we get by 2) for some p' that
    p ~~i'~▶ p', and
    sP ≼l p'.

  Since
    p ~~i'~▶ p', and
    í' = Just i',
  we get by definition of (Maybe-In) that
    (maybe p) ~~í'~~▶ (maybe p').

  Set
    pM' = (maybe p').
  Then
```

```
    (maybe p) ~~í'~► pM'.

  Since
    (maybe sP) --s--►,
    sP ⩿l p',
    pM' = (maybe p'),
    (maybe p) ~~í'~► pM', and
    í' (=MaybeI)l í,
  we get
    ⟨s,pM'⟩ ∊ R.

Case í = Just i,  í' = Nothing:

  Since
    (maybe sP) --?í.s--►,
  we get by definition of (Maybe-In) that,
  for some sP',
    sP = ?i.sP', and
    (maybe sP') --s--►.

  By definition of (=MaybeI), we have
    Nothing (=MaybeI)l ●.

  Since
    í' (=MaybeI)l í,
    í' = Nothing, and
    Nothing (=MaybeI)l ●,
  we get by transitivity that
    í (=MaybeI)l ●.

  Since
    í (=MaybeI)l ●,
  we get by definition of (=MaybeI) that
    i =Il ●.

  Since
    sP ⩿l p,
    sP = ?i.sP', and
    i =Il ●,
  we get by 1) that
    sP' ⩿l p.

  Since
    í' = Nothing,
  we get by rule (Maybe-In●) that
    (maybe p) ~~í'~~► (maybe p).

  Let
    pM' = (maybe p).

  Since
    (maybe p) ~~í'~~► (maybe p), and
    pM' = (maybe p),
  we get
    (maybe p) ~~í'~~► pM'.

  Since
    (maybe sP') --s--► and
    sP' ⩿l p,
    pM' = (maybe p),
    (maybe p) ~~í'~~► pM', and
    í' (=MaybeI)l í,
  we get
    ⟨s,pM'⟩ ∊ R.

Case í = Just i,  í' = Just i':
```

```
    Since
      (maybe sP) --?í.s--►, and
      í = Just i,
    we get by definition of (Maybe-In) that,
    for some sP',
      sP = ?i.sP', and
      (maybe sP') --s--►.

    Since
      í  = Just i,
      í' = Just i', and
      í' (=MaybeI)l í,
    we get by definition of (=MaybeI) that
      i =Il i'.

    Since
      sP ≼l p,
      sP = ?i.sP', and
      i' =Il i,
    we get by 3) that, for some p',
      p ~~i'~~► p', and
      sP' ≼l p'.

    Since
      p ~~i'~~► p', and
      í' = Just i',
    we get by definition of (Maybe-In) that
      (maybe p) ~~í'~~► (maybe p').

    Let
      pM' = (maybe p').

    Since
      (maybe p) ~~í'~~► (maybe p'), and
      pM' = (maybe p'),
    we get
      (maybe p) ~~í'~~► pM'.

    Since
      (maybe sP') --s--► and
      sP' ≼l p',
      pM' = (maybe p'),
      (maybe p) ~~í'~~► pM', and
      í' (=MaybeI)l í,
    we get
      ⟨s,pM'⟩ ∊ R.

case 4):

  Pick
    ⟨!o.s,(maybe p)⟩ ∊ R

  To show:
  there exists
    o' =Ol
  such that
    (maybe p) ——o'—► pM' and
    ⟨s,pM'⟩ ∊ R.

  Since
    ⟨!o.s',(maybe p)⟩ ∊ R,
  we get
    (maybe sP) --(!o.s')--► and
    sP ≼l p.
```

```
    Since
      (maybe sP) --(!o.s')--►,
    we get for some sP' that
      sP = !o.sP' and
      (maybe sP') --s--►.

    Since
      sP ⩽l p,
    we get by 4) for some o' and p' that
      o' =Ol o,
      p ——o'—► p', and
      sP' ⩽l p'.

    Since
      p ——o'—► p',
    we get by definition of (Maybe-Out) that
      (maybe p) ——o'—► (maybe p').

    Set
      pM' = (maybe p').
    Since
      (maybe p) ——o'—► (maybe p'),
    we get
      (maybe p) ——o'—► pM'.

    Since
      (maybe sP') --s'-►,
      sP' ⩽l p',
      pM' = (maybe p'),
      (maybe p) ——o'—► pM', and
      o' =Ol o,
    we get
      ⟨s,pM'⟩ ∊ R.

  Thus,
    R is a l-(=MaybeI)-(=O)-simulation.

  Thus,
  for all l,
  there exists an l-(=MaybeI)-(=O)-simulation R such that
    ⟨s0,maybe p0⟩ ∊ R.

  Thus
    (maybe p0) ∊ NI(=MaybeI,=O).

Qed.

Loop
--------------------------------------------------------------------------------

Theorem:

  forall

    p ∊ IProc I I ,
    (=I),

  if

    p ∊ NI(=I,=I) ,

  then

    (loop p) ∊ NI(=I,=I).
```

```
Proof.

  Pick p0, (=I) satisfying the above assumptions.

  Pick s0 such that
    (loop p0) --s0-►.

  Pick l.

  ***

  To show: there exists a relation R such that
    ⟨s0,loop p0⟩ ∊ R
  and
    R is a l-(=I)-(=I)-simulation.

  Pick
    R = { ⟨s,loop p⟩ | exists sP . sP ≼l p AND (loop sP) --s-► }.

  (here, ≼ is a shorthand for ≼(=I)(=I) )

  ***

  To prove:
    ⟨s0,loop p0⟩ ∊ R.

  Set
    s  = s0,
    p  = p0,
  and construct
    sP
  such that
    (loop sP) --s--►
  from the proof of the derivation of
    (loop p0) --s0--►.

  Then
    ⟨s,loop p⟩ ∊ R.

  Thus,
    ⟨s0,loop p0⟩ ∊ R.

  ***

  To prove:
    R is a l-(=I)-(=I)-simulation.

  We prove that
  R satisfies pt. 1) through 4) of Def IV.2.

  case 1)

    Pick
      ⟨?i.s,(loop p)⟩ ∊ R
    such that
      i =Il ●.

    To show:
      ⟨s,(loop p)⟩ ∊ R.

    Since
      ⟨s,(loop p)⟩ ∊ R,
    we get for some sP that
      (loop sP) --?i.s--► and
      sP ≼l p.
```

```
  Since
    (loop sP) --?i.s--►,
  we get by definition of (Loop-In) that,
  for some sP',
    sP = ?i.sP', and
    (loop sP') --s--►.

  Since
    sP ≼l p,
    sP = ?i.sP', and
    i =Il ●,
  we get by 1) that
    sP' ≼l p.

  Since
    (loop sP') --s--► and
    sP' ≼l p,
  we get
    ⟨s,(loop p)⟩ ∊ R.

case 2)

  Pick
    ⟨s,(loop p)⟩ ∊ R.

  To show:
  forall
    i =Il ●,
  it holds that, for some pL',
    (loop p) ~~i~► pL' and
    ⟨s,pL'⟩ ∊ R.

  Since
    ⟨s,(loop p)⟩ ∊ R,
  we get
    (loop sP) --s--► and
    sP ≼l p.

  Pick
    i =Il ●.

  Since
    sP ≼l p,
  we get by 2) for some p' that
    p ~~i~► p', and
    sP ≼l p'.

  Set
    pL' = (loop p').

  Since
    (loop sP) --s--► and
    sP ≼l p',
  we get
    ⟨s,pL'⟩ ∊ R.

case 3)

  Pick
    ⟨?i.s',(loop p)⟩ ∊ R

  To show:
  forall
    i' =Il i,
```

```
  it holds that, for some pL',
    (loop p) ~~i'~► pL' and
    ⟨s',pL'⟩ ∊ R.

  Since
    ⟨?i.s',(loop p)⟩ ∊ R,
  we get
    (loop sP) --(?i.s')--► and
    sP ≼l p.

  Since
    (loop sP) --(?i.s')--►,
  we get for some sP' that
    sP = ?i.sP' and
    (loop sP') --s'--►.

  Pick
    i' =Il i.

  Since
    sP ≼l p,
  we get by 3) for some p' that
    p ~~i'~► p', and
    sP' ≼l p'.

  Set
    pL' = (loop p').

  Since
    (loop sP') --s'--► and
    sP' ≼l p',
  we get
  ⟨s',pL'⟩ ∊ R.

case 4):

  Pick
    ⟨!i.s',(loop p)⟩ ∊ R

  To show:
  there exists
    i' =Il i
  such that
    (loop p) ——i'—► pL' and
    ⟨s',pL'⟩ ∊ R.

  Since
    ⟨!i.s',(loop p)⟩ ∊ R,
  we get
    (loop sP) --(!i.s')-► and
    sP ≼l p.

  Since
    (loop sP) --(!i.s')-►,
  we get for some sP' that
    sP = !i.?i.sP' and
    (loop sP') --s'-►.

  Since
    sP ≼l p,
  we get by 4) for some i' and p' that
    i' =Il i,
    p ——i'—► p', and
    ?i.sP' ≼l p'.
```

```
    Since
      ?i.sP' ≼l p' and
      i' =Il i,
    we get by 3) for some p'' that
      p ~~i'~► p'', and
      sP' ≼l p''.

    Set
      pL' = (loop p'').

    Since
      (loop sP') --s'-► and
      sP' ≼l p'',
    we get
      ⟨s',pL'⟩ ∈ R.

  Thus,
    R is a l-(=I)-(=I)-simulation.

  Thus,
  for all l,
  there exists an l-(=I)-(=I)-simulation R such that
    ⟨s0,loop p0⟩ ∈ R.

  Thus
    (loop p0) ∈ NI(=I,=I).

Qed.

Par
--------------------------------------------------------------------------------

Theorem:

  forall

    pL : IProc I OL ,
    pR : IProc I OR ,
    (=I), (=OL), (=OR),

  if

    pL ∈ NI(=I,=OL) ,
    pR ∈ NI(=I,=OR) ,

  then

    (par pL pR) ∈ NI(=I,=O),

  where

    (=O) = eqpair●LR(=OL,=OR).

Proof.

  Pick pL0, pR0, (=I), (=OL), (=OR) satisfying the above assumptions.

  Set
    (=O) = eqpair●LR(=OL,=OR).

  Pick s0 such that
    par pL0 pR0 --s0-►.

  Pick l.

  ***
```

```
To show: there exists a relation R such that
  ⟨s0,par pL0 pR0⟩ ∊ R
and
  R is a l-(=I)-(=O)-simulation.

Pick
  R = { ⟨s,par pL pR⟩ | exists sPL, sPR .
                          sL ≼l pL,
                          sR ≼l pR, and
                          (par sL sR) --s--►   }.

(here, ≼ is a shorthand for ≼(=I)(=OL) and ≼(=I)(=OR) respectively )

***

To prove:
  ⟨s0,par pL0 pR0⟩ ∊ R.

Set
  s  = s0,
  pL = pL0,
  pR = pR0,
and construct
  sL, sR
such that
  (par sL sR) --s--►
from the proof of the derivation of
  (par pL0 pR0) --s0--►.

Then
  ⟨s,par pL pR⟩ ∊ R.

Thus,
  ⟨s0,par pL0 pR0⟩ ∊ R.

***

To prove:
  R is a l-(=I)-(=O)-simulation.

We prove that
R satisfies pt. 1) through 4) of Def IV.2.

case 1)

  Pick
    ⟨?i.s,(par pL pR)⟩ ∊ R
  such that
    i =Il ●.

  To show:
    ⟨s,(par pL pR)⟩ ∊ R.

  Since
    ⟨?i.s,(par pL pR)⟩ ∊ R,
  we get
    (par sL sR) --?i.s--►,
    sL ≼l pL, and
    sR ≼l pR.

  Since
    (par sL sR) --?i.s--►,
  we get by definition of (Par-In) that,
  for some sL' and sR',
```

```
    sL = ?i.sL',
    sR = ?i.sR', and
    (par sL' sR') --s--►.

  Since
    sL ≼l pL,
  we get by Def IV.2 1) that
    sL' ≼l pL.

  Since
    sR ≼l pR,
  we get by Dev IV.2 1) that
    sR' ≼l pR.

  Since
    (par sL' sR') --s--►.
    sL' ≼l pL, and
    sR' ≼l pR,
  we get
    ⟨s,(par pL pR)⟩ ∈ R.

case 2)

  Pick
    ⟨s,(par pL pR)⟩ ∈ R

  To show:
  forall
    i =Yl ●,
  it holds that, for some pP',
    (par pL pR) ~~i~► pP' and
    ⟨s,pP'⟩ ∈ R.

  Since
    ⟨s,(par pL pR)⟩ ∈ R,
  we get
    (par sL sR) --s--►,
    sL ≼l pL, and
    sR ≼l pR.

  Pick
    i =Il ●.

  Since
    sL ≼l pL,
  we get by Def IV.2 2) for some pL' that
    pL ~~i~► pL', and
    sL ≼l pL'.

  Since
    sR ≼l pR,
  we get by 2) for some pR' that
    pR ~~i~► pR', and
    sR ≼l pR'.

  Set
    pP' = (par pL' pR').

  By (PAR-IN),
    (par pL pR) ~~i~► pP'.

  Since
    (par sL sR) --s--►,
    sL ≼l pL', and
    sR ≼l pR',
```

```
  we get
    ⟨s,pP'⟩ ∊ R.

case 3)

  Pick
    ⟨?i.s',(par pL pR)⟩ ∊ R

  To show:
  forall
    i' =Yl i,
  it holds that, for some pP',
    (par pL pR) ~~i'~► pP' and
    ⟨s',pP'⟩ ∊ R.

  Since
    ⟨?i.s',(par pL pR)⟩ ∊ R,
  we get
    (par sL sR) --(?i.s')--►,
    sL ≼l pL, and
    sR ≼l pR.

  Since
    (par sL sR) --(?i.s')--►,
  we get for some sL' and sR' that
    sL = ?i.sL',
    sR = ?i.sR', and
    (par sL' sR') --s'--►.

  Pick
    i' =Il i.

  Since
    sL ≼l pL,
    we get by Def IV.2 3) for some pL' that
    pL ~~i'~► pL', and
    sL' ≼l pL'.

  Since
    sR ≼l pR,
    we get by Def IV.2 3) for some pR' that
    pR ~~i'~► pR', and
    sR' ≼l pR'.

  Set
    pP' = (par pL' pR').

  By (PAR-IN),
    (par pL pR) ~~i~► pP'.

  Since
    (par sL' sR') --s'--►,
    sL' ≼l pL', and
    sR' ≼l pR',
  we get
    ⟨s',pZ'⟩ ∊ R.

case 4):

  Pick
    ⟨!o.s',(par pL pR)⟩ ∊ R

  To show:
  there exists
    o' =Ol o
```

```
  such that, for some pP',
    (par pL pR) ——o'—▶ pP' and
    ⟨s',pP'⟩ ∊ R.

  Since
    ⟨!o.s',(par pL pR)⟩ ∊ R,
  we get
    par sL sR --(!o.s')--▶,
    sL ≼l pL, and
    sR ≼l pR.

  Let
    ⟨oL,oR⟩ = o.

  Since
    (par sL sR) --(!o.s')--▶ and
    ⟨oL,oR⟩ = o,
  we get for some sL' and sR' that
    sL = !oL.sL',
    sR = !oR.sR' and
    (par sL' sR') --s'--▶.

  Since
    sL ≼l pL,
  we get by Def IV.2 4) for some oL' and pL' that
    oL' =OLl oL,
    pL ——oL'—▶ pL', and
    sL' ≼l pL'.

  Since
    sR ≼l pR,
  we get by Def IV.2 4) for some oR' and pR' that
    oR' =ORl oR,
    pR ——oR'—▶ pR', and
    sR' ≼l pR'.

  Let
    o' = ⟨oL',oR'⟩.

  Since
    oL =OLl oL' and
    oR =ORl oR',
  we get by definition of eqpair that
    o' =Ol o.

  Set
    pP' = (par pL' pR').

  By (PAR-OUT),
    (par pL pR) ——o—▶ pP'.

  Since
    (par sL' sR') --s'--▶,
    sL' ≼l pL', and
    sR' ≼l pR',
  we get
   ⟨s',pP'⟩ ∊ R.

Thus
  R is a l-(=I)-(=O)-simulation.

Thus,
for all l,
there exists an l-(=I)-(=O)-simulation R such that
  ⟨s0,par pL0 pR0⟩ ∊ R.
```

Thus
```
(par pL0 pR0) ∈ NI(=I,=O).
```

Qed.