```
sends-rule  THM

⊢  ∀i:Id, k:Knd, l:IdLnk, ds:x:Id fp-> Type, da:a:Knd fp-> Type,
|   f:(tg:Id
|       ×State(ds)→ma-valtype(da; k)→
|       (da(rcv(l; tg))?Void List)) List.
|  source(l) = i
|  ⇒
|  d-single-sends(i; ds; da; k; l; f)
|  realizes es.(∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
|              & (∀e:E.
|                 loc(e) = i
|                 ⇒
|                 (valtype(e) ⊆r ma-valtype(da; kind(e))))
|              & (∀e:E.
|                 isrcv(e)
|                 ⇒
|                 lnk(e) = l
|                 ⇒
|                 (valtype(e) ⊆r ma-valtype(da; kind(e))))
|              & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
|                 ((λl,tg. da(rcv(l; tg))?Top)))
|              & (∀e:E.
|                 loc(e) = i
|                 ⇒
|                 kind(e) = k
|                 ⇒
|                 sends(l;e)
|                 =
|                 tagged-messages(l;λz.(z when e);val(e);f))
| by Auto
|    THEN
|    BackThru
|    Thm* ∀D:Dsys, P:({es:ES| es is an event system of D }→Prop{i'}).
|         D realizes2 es.P(es) ⇒ D realizes es.P(es)
|    {d_DASH_realizes2_DASH_implies_DASH_realizes}
|    THENA
|    Try (Complete Auto)
|
|\
| .....wf .....
| 1. i : Id
| 2. k : Knd
| 3. l : IdLnk
| 4. ds : x:Id fp-> Type
| 5. da : a:Knd fp-> Type
| 6. f :
| |  (tg:Id
| |   ×State(ds)→ma-valtype(da; k)→(da(rcv(l; tg))?Void List)) List
| 7. source(l) = i
| ⊢  (λes.(∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
| |   & (∀e:E. loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
| |   & (∀e:E.
| |      isrcv(e)
| |      ⇒
```

```
| |      lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
| |   & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
| |      ((λl,tg. da(rcv(l; tg))?Top)))
| |   & (∀e:E.
| |      loc(e) = i
| |      ⇒
| |      kind(e) = k
| |      ⇒
| |      sends(l;e) = tagged-messages(l;λz.(z when e);val(e);f)))
| |  ∈ {es:ES
| |     | es is an event system of d-single-sends
| |       (i; ds; da; k; l; f) }
| |    →Prop  by Analyze THEN Analyze -1 THEN Analyze -1 THEN ExRepD
| |
| |\
| | 1. i : Id
| | 2. k : Knd
| | 3. l : IdLnk
| | 4. ds : x:Id fp-> Type
| | 5. da : a:Knd fp-> Type
| | 6. f :
| | |  (tg:Id
| | |   ×State(ds)→ma-valtype(da; k)→
| | |   (da(rcv(l; tg))?Void List)) List
| | 7. source(l) = i
| | 8. es : ES
| | 9. w : World
| | 10. p : FairFifo
| | 11. PossibleWorld(d-single-sends(i; ds; da; k; l; f);w)
| | 12. es = ES(w)
| | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
| | 14. ∀e:E. loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
| | 15. ∀e:E.
| | |   isrcv(e)
| | |   ⇒
| | |   lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
| | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
| | |   ((λl,tg. da(rcv(l; tg))?Top))
| | 17. e : E
| | 18. loc(e) = i
| | 19. kind(e) = k
| | 20. sends(l;e) = sends(l;e)
| | 21. (Msg on l) List
| | 22. (Msg on l) List
| | 23. u : (Msg on l)
| | ⊢  u ∈ Msg((λl,tg. da(rcv(l; tg))?Top))
| | | by SubsumeC {m:Msg| source(mlnk(m)) = i }
| | |
| | 1. i : Id
| | 2. k : Knd
| | 3. l : IdLnk
| | 4. ds : x:Id fp-> Type
| | 5. da : a:Knd fp-> Type
| | 6. f :
| |    (tg:Id
```

```
| |     ×State(ds)→ma-valtype(da; k)→
| |     (da(rcv(l; tg))?Void List)) List
| | 7. source(l) = i
| | 8. es : ES
| | 9. w : World
| | 10. p : FairFifo
| | 11. PossibleWorld(d-single-sends(i; ds; da; k; l; f);w)
| | 12. es = ES(w)
| | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
| | 14. ∀e:E. loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
| | 15. ∀e:E.
| |     isrcv(e)
| |     ⇒
| |     lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
| | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
| |     ((λl,tg. da(rcv(l; tg))?Top))
| | 17. e : E
| | 18. loc(e) = i
| | 19. kind(e) = k
| | 20. sends(l;e) = sends(l;e)
| | 21. (Msg on l) List
| | 22. (Msg on l) List
| | 23. u : (Msg on l)
| | ⊢  u ∈ {m:Msg| source(mlnk(m)) = i }
| |   by Analyze -1 THEN Analyze -1 THEN Analyze THEN HypSubst -1 0
| |
| .\
| . 1. i : Id
| . 2. k : Knd
| . 3. l : IdLnk
| . 4. ds : x:Id fp-> Type
| . 5. da : a:Knd fp-> Type
| . 6. f :
| . |  (tg:Id
| . |   ×State(ds)→ma-valtype(da; k)→
| . |   (da(rcv(l; tg))?Void List)) List
| . 7. source(l) = i
| . 8. es : ES
| . 9. w : World
| . 10. p : FairFifo
| . 11. PossibleWorld(d-single-sends(i; ds; da; k; l; f);w)
| . 12. es = ES(w)
| . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
| . 14. ∀e:E. loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
| . 15. ∀e:E.
| . |   isrcv(e)
| . |   ⇒
| . |   lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
| . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
| . |   ((λl,tg. da(rcv(l; tg))?Top))
| . 17. e : E
| . 18. loc(e) = i
| . 19. kind(e) = k
| . ⊢  tagged-messages(l;λz.(z when e);val(e);f)
| . |  ∈ Msg((λl,tg. da(rcv(l; tg))?Top)) List
```

```
| . | by Unfold `tagged-messages` 0
| . |
| . |\
| . | .....subterm .....
| . | 1. i : Id
| . | 2. k : Knd
| . | 3. l : IdLnk
| . | 4. ds : x:Id fp-> Type
| . | 5. da : a:Knd fp-> Type
| . | 6. f :
| . |    (tg:Id
| . |     ×State(ds)→ma-valtype(da; k)→
| . |     (da(rcv(l; tg))?Void List)) List
| . | 7. source(l) = i
| . | 8. es : ES
| . | 9. w : World
| . | 10. p : FairFifo
| . | 11. PossibleWorld(d-single-sends(i; ds; da; k; l; f);w)
| . | 12. es = ES(w)
| . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
| . | 14. ∀e:E. loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
| . | 15. ∀e:E.
| . |     isrcv(e)
| . |     ⇒
| . |     lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
| . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
| . |     ((λl,tg. da(rcv(l; tg))?Top))
| . | 17. e : E
| . | 18. loc(e) = i
| . | 19. kind(e) = k
| . | 20. x : t:Id×da(rcv(l; t))?Void
| . | ⊢  <l,x> ∈ Msg((λl,tg. da(rcv(l; tg))?Top))
| . |   by Analyze -1 THEN Fold `msg` 0 THEN Reduce 0 THEN DoSubsume
| . |
| . |\
| . | .....subterm .....
| . | 1. i : Id
| . | 2. k : Knd
| . | 3. l : IdLnk
| . | 4. ds : x:Id fp-> Type
| . | 5. da : a:Knd fp-> Type
| . | 6. f :
| . |    (tg:Id
| . |     ×State(ds)→ma-valtype(da; k)→
| . |     (da(rcv(l; tg))?Void List)) List
| . | 7. source(l) = i
| . | 8. es : ES
| . | 9. w : World
| . | 10. p : FairFifo
| . | 11. PossibleWorld(d-single-sends(i; ds; da; k; l; f);w)
| . | 12. es = ES(w)
| . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
| . | 14. ∀e:E. loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
| . | 15. ∀e:E.
| . |     isrcv(e)
```

```
| . |     ⇒
| . |     lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
| . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
| . |     ((λl,tg. da(rcv(l; tg))?Top))
| . | 17. e : E
| . | 18. loc(e) = i
| . | 19. kind(e) = k
| . | ⊢  (λz.(z when e)) ∈ State(ds)
| . |   by Unfold `ma-state` 0 THEN DoSubsume
| . |      THEN
| . |      Subst (loc(e) = i) 0
| . |      THEN
| . |      BackThruSomeHyp
| . |
| . .\
| . . .....subterm .....
| . . 1. i : Id
| . . 2. k : Knd
| . . 3. l : IdLnk
| . . 4. ds : x:Id fp-> Type
| . . 5. da : a:Knd fp-> Type
| . . 6. f :
| . .    (tg:Id
| . .     ×State(ds)→ma-valtype(da; k)→
| . .     (da(rcv(l; tg))?Void List)) List
| . . 7. source(l) = i
| . . 8. es : ES
| . . 9. w : World
| . . 10. p : FairFifo
| . . 11. PossibleWorld(d-single-sends(i; ds; da; k; l; f);w)
| . . 12. es = ES(w)
| . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
| . . 14. ∀e:E. loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
| . . 15. ∀e:E.
| . .     isrcv(e)
| . .     ⇒
| . .     lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
| . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
| . .     ((λl,tg. da(rcv(l; tg))?Top))
| . . 17. e : E
| . . 18. loc(e) = i
| . . 19. kind(e) = k
| . . ⊢  val(e) ∈ ma-valtype(da; k)
| . .   by Thin 15 THEN DoSubsume THEN RevHypSubst -1 0
| . .      THEN
| . .      BackThruSomeHyp
|
.\
. 1. i : Id
. 2. k : Knd
. 3. l : IdLnk
. 4. ds : x:Id fp-> Type
. 5. da : a:Knd fp-> Type
. 6. f :
. |  (tg:Id
```

```
. |   ×State(ds)→ma-valtype(da; k)→(da(rcv(l; tg))?Void List)) List
. 7. source(l) = i
. ⊢  d-single-sends(i; ds; da; k; l; f
. |                ) realizes2 es.(∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. |  & (∀e:E. loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. |  & (∀e:E.
. |     isrcv(e)
. |     ⇒
. |     lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. |  & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. |     ((λl,tg. da(rcv(l; tg))?Top)))
. |  & (∀e:E.
. |     loc(e) = i
. |     ⇒
. |     kind(e) = k
. |     ⇒
. |     sends(l;e) = tagged-messages(l;λz.(z when e);val(e);f))
. | by RepeatFor 3 (Analyze 0 THENA Complete Auto) THEN ES_Reduce
. |    THEN
. |    PW_Reduce -1
. |    THEN
. |    Analyze 0
. |
. |\
. | 1. i : Id
. | 2. k : Knd
. | 3. l : IdLnk
. | 4. ds : x:Id fp-> Type
. | 5. da : a:Knd fp-> Type
. | 6. f :
. | |  (tg:Id
. | |   ×State(ds)→ma-valtype(da; k)→
. | |   (da(rcv(l; tg))?Void List)) List
. | 7. source(l) = i
. | 8. w : World
. | 9. p : FairFifo
. | 10. FairFifo
. | 11. ∀i@0,x:Id.
. | |   vartype(i@0;x) ⊆r if deq-member(IdDeq;x;1of(1of(if eqof(IdDeq)
. | |                                                    (i@0
. | |                                                    ,i)→
. | |                                                    <ds
. | |                                                    ,da
. | |                                                    ,
. | |                                                    ,
. | |                                                    ,
. | |                                                    ,<[<k,l>]
. | |                                                     ,λx.f>
. | |                                                    ,
. | |                                                    ,
. | |                                                    ,•>
. | |                                                  else  fi)))→
. | |                        2of(1of(if eqof(IdDeq)(i@0,i)→
. | |                                   <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | |                                else  fi))
```

```
. | |                        (x)
. | |                    else Top fi
. | 12. ∀i@0:Id, a:Action(i@0).
. | |   ¬isnull(a)
. | |   ⇒
. | |   (valtype(i@0;a) ⊆r if eqof(IdDeq)(i@0,i)→
. | |                         <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | |                      else  fi.da(kind(a)))
. | 13. ∀l@0:IdLnk, tg:Id.
. | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. | |                       <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | |                    else  fi.da(rcv(l@0; tg))
. | 14. ∀i@0,x:Id.
. | |   deq-member(IdDeq;x;1of(1of(2of(2of(if eqof(IdDeq)(i@0,i)→
. | |                                         <ds
. | |                                         ,da
. | |                                         ,
. | |                                         ,
. | |                                         ,
. | |                                         ,<[<k,l>],λx.f>
. | |                                         ,
. | |                                         ,
. | |                                         ,•>
. | |                                      else  fi)))))
. | |   ⇒
. | |   s(i@0;0).x
. | |   =
. | |   2of(1of(2of(2of(if eqof(IdDeq)(i@0,i)→
. | |                      <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | |                   else  fi))))
. | |   (x)
. | 15. ∀i@0:Id, t:ℕ.
. | |   ¬isnull(a(i@0;t))
. | |   ⇒
. | |   (islocal(kind(a(i@0;t)))
. | |    ⇒
. | |    deq-member(IdDeq;act(kind(a(i@0;t)));1of(1of(2of(2of(2of(
. | |                                                  if eqof(IdDeq)
. | |                                                     (i@0
. | |                                                     ,i)→
. | |                                                     <ds
. | |                                                     ,da
. | |                                                     ,
. | |                                                     ,
. | |                                                     ,
. | |                                                     ,<[<k,l>]
. | |                                                      ,λx.f>
. | |                                                     ,
. | |                                                     ,
. | |                                                     ,•>
. | |                                                  else  fi))))))
. | |    ⇒
. | |    2of(1of(2of(2of(2of(if eqof(IdDeq)(i@0,i)→
. | |                           <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | |                        else  fi)))))
```

```
. | |    (act(kind(a(i@0;t)))
. | |    ,λx.s(i@0;t).x
. | |    ,val(a(i@0;t))))
. | |   & (∀x:Id.
. | |      deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<kind(...)
. | |                                                    ,x>;1of(1of(
. | |                                                        2of(2of(
. | |                                                        2of(2of(
. | |                                                        ...)))))))
. | |      ⇒
. | |      s(i@0;t+1).x
. | |      =
. | |      2of(1of(2of(2of(2of(2of(if eqof(IdDeq)(i@0,i)→
. | |                                 <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | |                               else  fi))))))
. | |      (<kind(a(i@0;t)),x>
. | |      ,λx.s(i@0;t).x
. | |      ,val(a(i@0;t))))
. | |   & (∀l@0:IdLnk.
. | |      deq-member(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq);...;...)
. | |      ⇒
. | |      withlnk(l@0;m(i@0;t))
. | |      =
. | |      if source(l@0) = i@0→
. | |         concat(map(λtgf.map(λx.
. | |         <1of(tgf)
. | |         ,x>;2of(tgf)
. | |             ((λx.s(i@0;t).x)
. | |             ,val(a(i@0;t))));2of(1of(2of(2of(2of(2of(2of(
. | |                                       if eqof(IdDeq)(i@0,i)→
. | |                                          <ds
. | |                                          ,da
. | |                                          ,
. | |                                          ,
. | |                                          ,
. | |                                          ,<[<k,l>],λx.f>
. | |                                          ,
. | |                                          ,
. | |                                          ,•>
. | |                                       else  fi)))))))
. | |                                  (<kind(a(i@0;t)),l@0>)))
. | |      else nil fi)
. | |   & (∀x:Id.
. | |      ¬(deq-member(IdDeq;x;1of(1of(2of(2of(2of(2of(2of(2of(
. | |                                      if eqof(IdDeq)(i@0,i)→
. | |                                         <ds
. | |                                         ,da
. | |                                         ,
. | |                                         ,
. | |                                         ,
. | |                                         ,<[<k,l>],λx.f>
. | |                                         ,
. | |                                         ,
. | |                                         ,•>
. | |                                      else  fi)))))))))
```

```
. | |        ⇒
. | |        deq-member(KindDeq;kind(a(i@0;t));2of(
. | |                                          1of(2of(2of(2of(2of(
. | |                                              2of(2of(
. | |                                              if eqof(IdDeq)
. | |                                                 (i@0
. | |                                                 ,i)→
. | |                                                 <ds
. | |                                                 ,da
. | |                                                 ,
. | |                                                 ,
. | |                                                 ,
. | |                                                 ,<[<k,l>],λx.f>
. | |                                                 ,
. | |                                                 ,
. | |                                                 ,·>
. | |                                              else  fi)))))))))
. | |                                          (x)))
. | |      ⇒
. | |      s(i@0;t).x = s(i@0;t+1).x)
. | |   & (∀l@0:IdLnk, tg:Id.
. | |      ¬(deq-member(product-deq(IdLnk;Id;IdLnkDeq;IdDeq);...;...)
. | |        ⇒
. | |        deq-member(KindDeq;kind(a(i@0;t));2of(
. | |                                          1of(2of(2of(2of(2of(
. | |                                              2of(2of(2of(
. | |                                              if eqof(IdDeq)
. | |                                                 (i@0
. | |                                                 ,i)→
. | |                                                 <ds
. | |                                                 ,da
. | |                                                 ,
. | |                                                 ,
. | |                                                 ,
. | |                                                 ,<[<k,l>],λx.f>
. | |                                                 ,
. | |                                                 ,
. | |                                                 ,·>
. | |                                              else  fi)))))))))
. | |                                          (<l@0,tg>)))
. | |      ⇒
. | |      w-tagged(tg; onlnk(l@0;m(i@0;t))) = nil)
. | 16. ∀i@0,a:Id, t:ℕ.
. | |   ∃t':ℕ.
. | |   t≤t'
. | |   & ¬isnull(a(i@0;t')) & kind(a(i@0;t')) = locl(a)
. | |     ∨ ¬a declared in if eqof(IdDeq)(i@0,i)→
. | |                         <ds,da,,,,<[<k,l>],λx.f>,,,·>
. | |                      else  fi
. | |     ∨ unsolvable if eqof(IdDeq)(i@0,i)→
. | |                     <ds,da,,,,<[<k,l>],λx.f>,,,·>
. | |                  else  fi.pre(a,λx.s(i@0;t').x)
. | ⊢  (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. | |  & (∀e:E. loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. | |  & (∀e:E.
```

```
. | |     isrcv(e)
. | |     ⇒
. | |     lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. | |  & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. | |     ((λl,tg. da(rcv(l; tg))?Top)))
. | | by RepeatFor 3 (Thin -1)
. | |    THEN
. | |    AllHyps
. | |    (λh.
. | |     InstHyp [i] h THENA Complete Auto THEN Thin h
. | |     THEN
. | |     Subst ((eqof(IdDeq)(i,i)) ∼ true₂) -1
. | |     THEN
. | |     Try (Unfolds [‘ma-da‘;‘ma-din‘] -1)
. | |     THEN
. | |     Reduce -1
. | |     THEN
. | |     Try
. | |     (BackThru
. | |      Thm* ∀d:EqDecider(A), i:A. (eqof(d)(i,i)) ∼ true₂
. | |      {eqof_eq_btrue}))
. | |    THEN
. | |    SplitAndConcl
. | |
. | |\
. | | 1. i : Id
. | | 2. k : Knd
. | | 3. l : IdLnk
. | | 4. ds : x:Id fp-> Type
. | | 5. da : a:Knd fp-> Type
. | | 6. f :
. | | |  (tg:Id
. | | |   ×State(ds)→ma-valtype(da; k)→
. | | |   (da(rcv(l; tg))?Void List)) List
. | | 7. source(l) = i
. | | 8. w : World
. | | 9. FairFifo
. | | 10. FairFifo
. | | 11. ∀l@0:IdLnk, tg:Id.
. | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. | | |                       <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | | |                    else  fi.da(rcv(l@0; tg))
. | | 12. ∀a:Action(i).
. | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | 13. ∀x:Id.
. | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | | |                   else Top fi
. | | ⊢  ∀x:Id. vartype(i;x) ⊆r ds(x)?Top  by Auto
. | | |
. | | 1. i : Id
. | | 2. k : Knd
. | | 3. l : IdLnk
. | | 4. ds : x:Id fp-> Type
. | | 5. da : a:Knd fp-> Type
. | | 6. f :
```

```
. | | |  (tg:Id
. | | |   ×State(ds)→ma-valtype(da; k)→
. | | |   (da(rcv(l; tg))?Void List)) List
. | | 7. source(l) = i
. | | 8. w : World
. | | 9. FairFifo
. | | 10. FairFifo
. | | 11. ∀l@0:IdLnk, tg:Id.
. | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. | | |                      <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | | |                   else  fi.da(rcv(l@0; tg))
. | | 12. ∀a:Action(i).
. | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | 13. ∀x:Id.
. | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | | |                   else Top fi
. | | 14. x : Id
. | | ⊢  vartype(i;x) ⊆r ds(x)?Top  by RepeatFor 3 (Thin -4)
. | | |
. | | 1. i : Id
. | | 2. k : Knd
. | | 3. l : IdLnk
. | | 4. ds : x:Id fp-> Type
. | | 5. da : a:Knd fp-> Type
. | | 6. (tg:Id
. | | |   ×State(ds)→ma-valtype(da; k)→
. | | |   (da(rcv(l; tg))?Void List)) List
. | | 7. source(l) = i
. | | 8. w : World
. | | 9. ∀a:Action(i). ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | 10. ∀x:Id.
. | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | | |                   else Top fi
. | | 11. x : Id
. | | ⊢  vartype(i;x) ⊆r ds(x)?Top
. | | | by InstHyp [x] -2 THENA (Auto THEN PWTrivial)
. | | |
. | | 1. i : Id
. | | 2. k : Knd
. | | 3. l : IdLnk
. | | 4. ds : x:Id fp-> Type
. | | 5. da : a:Knd fp-> Type
. | | 6. (tg:Id
. | |     ×State(ds)→ma-valtype(da; k)→
. | |     (da(rcv(l; tg))?Void List)) List
. | | 7. source(l) = i
. | | 8. w : World
. | | 9. ∀a:Action(i). ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | 10. ∀x:Id.
. | |     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | |                   else Top fi
. | | 11. x : Id
. | | 12. vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | |                   else Top fi
. | | ⊢  vartype(i;x) ⊆r ds(x)?Top
```

```
. | |   by Repeat (Unfolds [‘fpf-ap‘;‘fpf-cap‘] 0)
. | |      THEN
. | |      Fold ‘fpf-dom‘ -1
. | |      THEN
. | |      NthHypSq -1
. | |      THEN
. | |      RepeatFor 2 (Analyze THEN Try Trivial)
. | |
. | |\
. | | 1. i : Id
. | | 2. k : Knd
. | | 3. l : IdLnk
. | | 4. ds : x:Id fp-> Type
. | | 5. da : a:Knd fp-> Type
. | | 6. f :
. | | |  (tg:Id
. | | |   ×State(ds)→ma-valtype(da; k)→
. | | |   (da(rcv(l; tg))?Void List)) List
. | | 7. source(l) = i
. | | 8. w : World
. | | 9. p : FairFifo
. | | 10. FairFifo
. | | 11. ∀l@0:IdLnk, tg:Id.
. | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. | | |                       <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | | |                    else  fi.da(rcv(l@0; tg))
. | | 12. ∀a:Action(i).
. | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | 13. ∀x:Id.
. | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | | |                   else Top fi
. | | ⊢  ∀e:E. loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
. | | | by Auto
. | | |
. | | 1. i : Id
. | | 2. k : Knd
. | | 3. l : IdLnk
. | | 4. ds : x:Id fp-> Type
. | | 5. da : a:Knd fp-> Type
. | | 6. f :
. | | |  (tg:Id
. | | |   ×State(ds)→ma-valtype(da; k)→
. | | |   (da(rcv(l; tg))?Void List)) List
. | | 7. source(l) = i
. | | 8. w : World
. | | 9. p : FairFifo
. | | 10. FairFifo
. | | 11. ∀l@0:IdLnk, tg:Id.
. | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. | | |                       <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | | |                    else  fi.da(rcv(l@0; tg))
. | | 12. ∀a:Action(i).
. | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | 13. ∀x:Id.
. | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
```

```
. | | |                   else Top fi
. | | 14. e : E
. | | 15. loc(e) = i
. | | ⊢  valtype(e) ⊆r ma-valtype(da; kind(e))  by Thin -5
. | | |
. | | 1. i : Id
. | | 2. k : Knd
. | | 3. l : IdLnk
. | | 4. ds : x:Id fp-> Type
. | | 5. da : a:Knd fp-> Type
. | | 6. (tg:Id
. | | |   ×State(ds)→ma-valtype(da; k)→
. | | |   (da(rcv(l; tg))?Void List)) List
. | | 7. source(l) = i
. | | 8. w : World
. | | 9. p : FairFifo
. | | 10. FairFifo
. | | 11. ∀a:Action(i).
. | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | 12. ∀x:Id.
. | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | | |                   else Top fi
. | | 13. e : E
. | | 14. loc(e) = i
. | | ⊢  valtype(e) ⊆r ma-valtype(da; kind(e))  by Thin -3
. | | |
. | | 1. i : Id
. | | 2. k : Knd
. | | 3. l : IdLnk
. | | 4. ds : x:Id fp-> Type
. | | 5. da : a:Knd fp-> Type
. | | 6. (tg:Id
. | | |   ×State(ds)→ma-valtype(da; k)→
. | | |   (da(rcv(l; tg))?Void List)) List
. | | 7. source(l) = i
. | | 8. w : World
. | | 9. p : FairFifo
. | | 10. FairFifo
. | | 11. ∀a:Action(i).
. | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | 12. e : E
. | | 13. loc(e) = i
. | | ⊢  valtype(e) ⊆r ma-valtype(da; kind(e))
. | | | by InstHyp [act(e)] -3
. | | |
. | | |\
. | | | .....wf .....
. | | | 1. i : Id
. | | | 2. k : Knd
. | | | 3. l : IdLnk
. | | | 4. ds : x:Id fp-> Type
. | | | 5. da : a:Knd fp-> Type
. | | | 6. (tg:Id
. | | |     ×State(ds)→ma-valtype(da; k)→
. | | |     (da(rcv(l; tg))?Void List)) List
```

```
. | | | 7. source(l) = i
. | | | 8. w : World
. | | | 9. FairFifo
. | | | 10. FairFifo
. | | | 11. ∀a:Action(i).
. | | |     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | | 12. e : E
. | | | 13. loc(e) = i
. | | | ⊢  act(e) ∈ Action(i)
. | | |   by DoSubsume THEN Analyze 0 THEN HypSubst -3 -2
. | | |
. | | |\
. | | | .....antecedent .....
. | | | 1. i : Id
. | | | 2. k : Knd
. | | | 3. l : IdLnk
. | | | 4. ds : x:Id fp-> Type
. | | | 5. da : a:Knd fp-> Type
. | | | 6. (tg:Id
. | | |     ×State(ds)→ma-valtype(da; k)→
. | | |     (da(rcv(l; tg))?Void List)) List
. | | | 7. source(l) = i
. | | | 8. w : World
. | | | 9. FairFifo
. | | | 10. FairFifo
. | | | 11. ∀a:Action(i).
. | | |     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | | 12. e : E
. | | | 13. loc(e) = i
. | | | ⊢  ¬isnull(act(e))
. | | |   by Inst
. | | |      Thm* ∀the_w:World, e:E.
. | | |           act(e) ∈ {a:Action(loc(e))| ¬isnull(a) }
. | | |      {w_DASH_act_wf}
. | | |      [w;e]
. | | |      THEN
. | | |      Analyze -1
. | | |
. | | .\
. | | . 1. i : Id
. | | . 2. k : Knd
. | | . 3. l : IdLnk
. | | . 4. ds : x:Id fp-> Type
. | | . 5. da : a:Knd fp-> Type
. | | . 6. (tg:Id
. | | . |   ×State(ds)→ma-valtype(da; k)→
. | | . |   (da(rcv(l; tg))?Void List)) List
. | | . 7. source(l) = i
. | | . 8. w : World
. | | . 9. p : FairFifo
. | | . 10. FairFifo
. | | . 11. ∀a:Action(i).
. | | . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | . 12. e : E
. | | . 13. loc(e) = i
```

```
. | | . 14. valtype(i;act(e)) ⊆r da(kind(act(e)))?Top
. | | . ⊢  valtype(e) ⊆r ma-valtype(da; kind(e))
. | | . | by Fold ‘ma-valtype‘ -1 THEN NthHypEq -1 THEN Analyze
. | | . |
. | | . |\
. | | . | .....subterm .....
. | | . | 1. i : Id
. | | . | 2. k : Knd
. | | . | 3. l : IdLnk
. | | . | 4. ds : x:Id fp-> Type
. | | . | 5. da : a:Knd fp-> Type
. | | . | 6. (tg:Id
. | | . | |   ×State(ds)→ma-valtype(da; k)→
. | | . | |   (da(rcv(l; tg))?Void List)) List
. | | . | 7. source(l) = i
. | | . | 8. w : World
. | | . | 9. p : FairFifo
. | | . | 10. FairFifo
. | | . | 11. ∀a:Action(i).
. | | . | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | . | 12. e : E
. | | . | 13. loc(e) = i
. | | . | 14. valtype(i;act(e)) ⊆r ma-valtype(da; kind(act(e)))
. | | . | ⊢  valtype(e) = valtype(i;act(e))
. | | . | | by Assert (¬isnull(act(e)))
. | | . | |
. | | . | |\
. | | . | | .....assertion .....
. | | . | | 1. i : Id
. | | . | | 2. k : Knd
. | | . | | 3. l : IdLnk
. | | . | | 4. ds : x:Id fp-> Type
. | | . | | 5. da : a:Knd fp-> Type
. | | . | | 6. (tg:Id
. | | . | |     ×State(ds)→ma-valtype(da; k)→
. | | . | |     (da(rcv(l; tg))?Void List)) List
. | | . | | 7. source(l) = i
. | | . | | 8. w : World
. | | . | | 9. FairFifo
. | | . | | 10. FairFifo
. | | . | | 11. ∀a:Action(i).
. | | . | |     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | . | | 12. e : E
. | | . | | 13. loc(e) = i
. | | . | | 14. valtype(i;act(e)) ⊆r ma-valtype(da; kind(act(e)))
. | | . | | ⊢  ¬isnull(act(e))
. | | . | |   by Inst
. | | . | |      Thm* ∀the_w:World, e:E.
. | | . | |           act(e) ∈ {a:Action(loc(e))| ¬isnull(a) }
. | | . | |      {w_DASH_act_wf}
. | | . | |      [w;e]
. | | . | |      THEN
. | | . | |      Analyze -1
. | | . | |
. | | . | .\
```

```
. | | . | . 1. i : Id
. | | . | . 2. k : Knd
. | | . | . 3. l : IdLnk
. | | . | . 4. ds : x:Id fp-> Type
. | | . | . 5. da : a:Knd fp-> Type
. | | . | . 6. (tg:Id
. | | . | .  |   ×State(ds)→ma-valtype(da; k)→
. | | . | .  |   (da(rcv(l; tg))?Void List)) List
. | | . | . 7. source(l) = i
. | | . | . 8. w : World
. | | . | . 9. p : FairFifo
. | | . | . 10. FairFifo
. | | . | . 11. ∀a:Action(i).
. | | . | .  |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | . | . 12. e : E
. | | . | . 13. loc(e) = i
. | | . | . 14. valtype(i;act(e)) ⊆r ma-valtype(da; kind(act(e)))
. | | . | . 15. ¬isnull(act(e))
. | | . | . ⊢  valtype(e) = valtype(i;act(e))
. | | . | .  | by Subst (valtype(e) ∼ valtype(loc(e);act(e))) 0
. | | . | .  |
. | | . | .  |\
. | | . | .  | .....equality .....
. | | . | .  | 1. i : Id
. | | . | .  | 2. k : Knd
. | | . | .  | 3. l : IdLnk
. | | . | .  | 4. ds : x:Id fp-> Type
. | | . | .  | 5. da : a:Knd fp-> Type
. | | . | .  | 6. (tg:Id
. | | . | .  |  |     ×State(ds)→ma-valtype(da; k)→
. | | . | .  |  |     (da(rcv(l; tg))?Void List)) List
. | | . | .  | 7. source(l) = i
. | | . | .  | 8. w : World
. | | . | .  | 9. p : FairFifo
. | | . | .  | 10. FairFifo
. | | . | .  | 11. ∀a:Action(i).
. | | . | .  |  |     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | . | .  | 12. e : E
. | | . | .  | 13. loc(e) = i
. | | . | .  | 14. valtype(i;act(e)) ⊆r ma-valtype(da; kind(act(e)))
. | | . | .  | 15. ¬isnull(act(e))
. | | . | .  | ⊢  valtype(e) ∼ valtype(loc(e);act(e))
. | | . | .  |   by DVar ‘e’ THEN DVar ‘e’
. | | . | .  |      THEN
. | | . | .  |      All (λh.Unfolds [‘w-act‘] h THEN Reduce h)
. | | . | .  |      THEN
. | | . | .  |      Fold ‘w-es‘ 0
. | | . | .  |      THEN
. | | . | .  |      BackThru
. | | . | .  |      Thm* ∀i:Id, w:World, p:FairFifo, t:ℕ.
. | | . | .  |           ¬isnull(a(i;t))
. | | . | .  |           ⇒
. | | . | .  |           (valtype(<i,t>) ∼ valtype(i;a(i;t)))
. | | . | .  |      {es_DASH_valtype_DASH_w_DASH_valtype}
. | | . | .  |
```

```
. | | . | . .\
. | | . | . . 1. i : Id
. | | . | . . 2. k : Knd
. | | . | . . 3. l : IdLnk
. | | . | . . 4. ds : x:Id fp-> Type
. | | . | . . 5. da : a:Knd fp-> Type
. | | . | . . 6. (tg:Id
. | | . | . .      ×State(ds)→ma-valtype(da; k)→
. | | . | . .      (da(rcv(l; tg))?Void List)) List
. | | . | . . 7. source(l) = i
. | | . | . . 8. w : World
. | | . | . . 9. FairFifo
. | | . | . . 10. FairFifo
. | | . | . . 11. ∀a:Action(i).
. | | . | . .     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | . | . . 12. e : E
. | | . | . . 13. loc(e) = i
. | | . | . . 14. valtype(i;act(e)) ⊆r ma-valtype(da; kind(act(e)))
. | | . | . . 15. ¬isnull(act(e))
. | | . | . . ⊢  valtype(loc(e);act(e)) = valtype(i;act(e))
. | | . | . .   by Analyze
. | | . |

. | | . .\
. | | . . .....subterm .....
. | | . . 1. i : Id
. | | . . 2. k : Knd
. | | . . 3. l : IdLnk
. | | . . 4. ds : x:Id fp-> Type
. | | . . 5. da : a:Knd fp-> Type
. | | . . 6. (tg:Id
. | | . . |   ×State(ds)→ma-valtype(da; k)→
. | | . . |   (da(rcv(l; tg))?Void List)) List
. | | . . 7. source(l) = i
. | | . . 8. w : World
. | | . . 9. FairFifo
. | | . . 10. FairFifo
. | | . . 11. ∀a:Action(i).
. | | . . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | . . 12. e : E
. | | . . 13. loc(e) = i
. | | . . 14. valtype(i;act(e)) ⊆r ma-valtype(da; kind(act(e)))
. | | . . ⊢  ma-valtype(da; kind(e)) = ma-valtype(da; kind(act(e)))
. | | . . | by Unfold ‘w-ekind‘ 0
. | | . . |
. | | . . 1. i : Id
. | | . . 2. k : Knd
. | | . . 3. l : IdLnk
. | | . . 4. ds : x:Id fp-> Type
. | | . . 5. da : a:Knd fp-> Type
. | | . . 6. (tg:Id
. | | . . |   ×State(ds)→ma-valtype(da; k)→
. | | . . |   (da(rcv(l; tg))?Void List)) List
. | | . . 7. source(l) = i
. | | . . 8. w : World
. | | . . 9. FairFifo
```

```
. | | . . 10. FairFifo
. | | . . 11. ∀a:Action(i).
. | | . . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | . . 12. e : E
. | | . . 13. loc(e) = i
. | | . . 14. valtype(i;act(e)) ⊆r ma-valtype(da; kind(act(e)))
. | | . . ⊢  ma-valtype(da; kind(act(e)))
. | | . . |  =
. | | . . |  ma-valtype(da; kind(act(e)))  by Auto
. | | . . |
. | | . . 1. i : Id
. | | . . 2. k : Knd
. | | . . 3. l : IdLnk
. | | . . 4. ds : x:Id fp-> Type
. | | . . 5. da : a:Knd fp-> Type
. | | . . 6. (tg:Id
. | | . .     ×State(ds)→ma-valtype(da; k)→
. | | . .     (da(rcv(l; tg))?Void List)) List
. | | . . 7. source(l) = i
. | | . . 8. w : World
. | | . . 9. FairFifo
. | | . . 10. FairFifo
. | | . . 11. ∀a:Action(i).
. | | . .     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | . . 12. e : E
. | | . . 13. loc(e) = i
. | | . . 14. valtype(i;act(e)) ⊆r ma-valtype(da; kind(act(e)))
. | | . . ⊢  ¬isnull(act(e))  by Unfold ‘w-act‘ 0 THEN DVar ‘e’
. | |
. | |\
. | | 1. i : Id
. | | 2. k : Knd
. | | 3. l : IdLnk
. | | 4. ds : x:Id fp-> Type
. | | 5. da : a:Knd fp-> Type
. | | 6. f :
. | | |  (tg:Id
. | | |   ×State(ds)→ma-valtype(da; k)→
. | | |   (da(rcv(l; tg))?Void List)) List
. | | 7. source(l) = i
. | | 8. w : World
. | | 9. p : FairFifo
. | | 10. FairFifo
. | | 11. ∀l@0:IdLnk, tg:Id.
. | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. | | |                       <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | | |                    else  fi.da(rcv(l@0; tg))
. | | 12. ∀a:Action(i).
. | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | 13. ∀x:Id.
. | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | | |                   else Top fi
. | | ⊢  ∀e:E.
. | | |  isrcv(e)
. | | |  ⇒
```

```
. | | |  lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
. | | | by Repeat
. | | |    (Unfolds
. | | |     [‘es-isrcv‘
. | | |     ;‘es-lnk‘
. | | |     ;‘es-tag‘
. | | |     ;‘es-kind‘
. | | |     ;‘es-valtype‘
. | | |     ;‘es-rcvtype‘]
. | | |     0
. | | |     THEN
. | | |     Reduce 0)
. | | |    THEN
. | | |    SplitOnConclITE
. | | |
. | | .....truecase .....
. | | 1. i : Id
. | | 2. k : Knd
. | | 3. l : IdLnk
. | | 4. ds : x:Id fp-> Type
. | | 5. da : a:Knd fp-> Type
. | | 6. f :
. | | |  (tg:Id
. | | |   ×State(ds)→ma-valtype(da; k)→
. | | |   (da(rcv(l; tg))?Void List)) List
. | | 7. source(l) = i
. | | 8. w : World
. | | 9. FairFifo
. | | 10. FairFifo
. | | 11. ∀l@0:IdLnk, tg:Id.
. | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. | | |                       <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | | |                   else  fi.da(rcv(l@0; tg))
. | | 12. ∀a:Action(i).
. | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | 13. ∀x:Id.
. | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | | |                   else Top fi
. | | 14. e : E
. | | 15. isrcv(kind(e))
. | | 16. lnk(kind(e)) = l
. | | 17. isrcv(kind(e))
. | | ⊢  (w.M(lnk(kind(e)),tag(kind(e)))) ⊆r ma-valtype(da; kind(e))
. | | | by InstHyp [lnk(kind(e));tag(kind(e))] -7
. | | |    THEN
. | | |    SplitOnHypITE -1
. | | |    THEN
. | | |    RW0 "deq_property<" -1
. | | |    THEN
. | | |    Unfold ‘ma-da‘ -2
. | | |    THEN
. | | |    Reduce -2
. | | |
. | | |\
. | | | 1. i : Id
```

```
. | | | 2. k : Knd
. | | | 3. l : IdLnk
. | | | 4. ds : x:Id fp-> Type
. | | | 5. da : a:Knd fp-> Type
. | | | 6. f :
. | | | |  (tg:Id
. | | | |   ×State(ds)→ma-valtype(da; k)→
. | | | |   (da(rcv(l; tg))?Void List)) List
. | | | 7. source(l) = i
. | | | 8. w : World
. | | | 9. FairFifo
. | | | 10. FairFifo
. | | | 11. ∀l@0:IdLnk, tg:Id.
. | | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. | | | |                       <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | | | |                    else  fi.da(rcv(l@0; tg))
. | | | 12. ∀a:Action(i).
. | | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | | 13. ∀x:Id.
. | | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | | | |                   else Top fi
. | | | 14. e : E
. | | | 15. isrcv(kind(e))
. | | | 16. lnk(kind(e)) = l
. | | | 17. isrcv(kind(e))
. | | | 18. (w.M(lnk(kind(e)),tag(kind(e)))) ⊆r da(rcv(lnk(kind(e));
. | | | |                                            tag(kind(e))
. | | | |                                            ))?Top
. | | | 19. source(lnk(kind(e))) = i
. | | | ⊢  (w.M(lnk(kind(e)),tag(kind(e)))) ⊆r ma-valtype(da; kind(e))
. | | | | by Unfold `ma-valtype` 0 THEN NthHypEq -2
. | | | |    THEN
. | | | |    Repeat Analyze
. | | | |
. | | | .....subterm .....
. | | | 1. i : Id
. | | | 2. k : Knd
. | | | 3. l : IdLnk
. | | | 4. ds : x:Id fp-> Type
. | | | 5. da : a:Knd fp-> Type
. | | | 6. f :
. | | |    (tg:Id
. | | |     ×State(ds)→ma-valtype(da; k)→
. | | |     (da(rcv(l; tg))?Void List)) List
. | | | 7. source(l) = i
. | | | 8. w : World
. | | | 9. FairFifo
. | | | 10. FairFifo
. | | | 11. ∀l@0:IdLnk, tg:Id.
. | | |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. | | |                       <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | | |                    else  fi.da(rcv(l@0; tg))
. | | | 12. ∀a:Action(i).
. | | |     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | | 13. ∀x:Id.
```

```
. | | |     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | | |                     else Top fi

. | | | 14. e : E
. | | | 15. isrcv(kind(e))
. | | | 16. lnk(kind(e)) = l
. | | | 17. isrcv(kind(e))
. | | | 18. (w.M(lnk(kind(e)),tag(kind(e)))) ⊆r da(rcv(lnk(kind(e));
. | | |                                                 tag(kind(e))
. | | |                                                 ))?Top
. | | | 19. source(lnk(kind(e))) = i
. | | | ⊢  kind(e) = rcv(lnk(kind(e)); tag(kind(e)))
. | | |   by MoveToConcl -3 THEN GenConclAtAddr [1;1;1]
. | | |      THEN
. | | |      Analyze -2
. | | |      THEN
. | | |      Analyze -2
. | | |      THEN
. | | |      Unfolds [‘isrcv‘;‘lnk‘;‘tagof‘] 0
. | | |      THEN
. | | |      Reduce 0
. | | |      THEN
. | | |      Try (Fold ‘rcv‘ 0)
. | | |
. | | .\
. | | . 1. i : Id
. | | . 2. k : Knd
. | | . 3. l : IdLnk
. | | . 4. ds : x:Id fp-> Type
. | | . 5. da : a:Knd fp-> Type
. | | . 6. f :
. | | . |  (tg:Id
. | | . |   ×State(ds)→ma-valtype(da; k)→
. | | . |   (da(rcv(l; tg))?Void List)) List
. | | . 7. source(l) = i
. | | . 8. w : World
. | | . 9. FairFifo
. | | . 10. FairFifo
. | | . 11. ∀l@0:IdLnk, tg:Id.
. | | . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. | | . |                       <ds,da,,,,<[<k,l>],λx.f>,,,·>
. | | . |                     else  fi.da(rcv(l@0; tg))
. | | . 12. ∀a:Action(i).
. | | . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | . 13. ∀x:Id.
. | | . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | | . |                     else Top fi
. | | . 14. e : E
. | | . 15. isrcv(kind(e))
. | | . 16. lnk(kind(e)) = l
. | | . 17. isrcv(kind(e))
. | | . 18. (w.M(lnk(kind(e)),tag(kind(e)))) ⊆r 1of(2of())(rcv
. | | . |   (lnk(kind(e)); tag(kind(e))))?Top
. | | . 19. ¬source(lnk(kind(e))) = i
. | | . ⊢  (w.M(lnk(kind(e)),tag(kind(e)))) ⊆r ma-valtype(da; kind(e))
. | | . | by Analyze -1
```

```
. | | . |
. | | . 1. i : Id
. | | . 2. k : Knd
. | | . 3. l : IdLnk
. | | . 4. ds : x:Id fp-> Type
. | | . 5. da : a:Knd fp-> Type
. | | . 6. f :
. | | .    (tg:Id
. | | .     ×State(ds)→ma-valtype(da; k)→
. | | .     (da(rcv(l; tg))?Void List)) List
. | | . 7. source(l) = i
. | | . 8. w : World
. | | . 9. FairFifo
. | | . 10. FairFifo
. | | . 11. ∀l@0:IdLnk, tg:Id.
. | | .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. | | .                         <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | | .                       else  fi.da(rcv(l@0; tg))
. | | . 12. ∀a:Action(i).
. | | .     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | | . 13. ∀x:Id.
. | | .     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | | .                     else Top fi
. | | . 14. e : E
. | | . 15. isrcv(kind(e))
. | | . 16. lnk(kind(e)) = l
. | | . 17. isrcv(kind(e))
. | | . 18. (w.M(lnk(kind(e)),tag(kind(e)))) ⊆r 1of(2of())(rcv
. | | .     (lnk(kind(e)); tag(kind(e))))?Top
. | | . ⊢  source(lnk(kind(e))) = i
. | | .   by AllHyps (λh.HypSubst' h 0 THEN Complete Auto)
. | |
. | .\
. | . 1. i : Id
. | . 2. k : Knd
. | . 3. l : IdLnk
. | . 4. ds : x:Id fp-> Type
. | . 5. da : a:Knd fp-> Type
. | . 6. f :
. | . |  (tg:Id
. | . |   ×State(ds)→ma-valtype(da; k)→
. | . |   (da(rcv(l; tg))?Void List)) List
. | . 7. source(l) = i
. | . 8. w : World
. | . 9. p : FairFifo
. | . 10. FairFifo
. | . 11. ∀l@0:IdLnk, tg:Id.
. | . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. | . |                       <ds,da,,,,<[<k,l>],λx.f>,,,•>
. | . |                     else  fi.da(rcv(l@0; tg))
. | . 12. ∀a:Action(i).
. | . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | . 13. ∀x:Id.
. | . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | . |                   else Top fi
```

```
. | . ⊢  {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. | . |  ((λl,tg. da(rcv(l; tg))?Top))  by RenameVar ‘l1’ 3
. | . |
. | . 1. i : Id
. | . 2. k : Knd
. | . 3. l1 : IdLnk
. | . 4. ds : x:Id fp-> Type
. | . 5. da : a:Knd fp-> Type
. | . 6. f :
. | . |  (tg:Id
. | . |   ×State(ds)→ma-valtype(da; k)→
. | . |   (da(rcv(l1; tg))?Void List)) List
. | . 7. source(l1) = i
. | . 8. w : World
. | . 9. p : FairFifo
. | . 10. FairFifo
. | . 11. ∀l@0:IdLnk, tg:Id.
. | . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. | . |                       <ds,da,,,,<[<k,l1>],λx.f>,,,•>
. | . |                    else  fi.da(rcv(l@0; tg))
. | . 12. ∀a:Action(i).
. | . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | . 13. ∀x:Id.
. | . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | . |                   else Top fi
. | . ⊢  {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. | . |  ((λl,tg. da(rcv(l; tg))?Top))
. | . | by Repeat (Unfolds [‘Msg‘;‘es-Msg‘] 0 THEN Reduce 0)
. | . |    THEN
. | . |    Analyze 0
. | . |    THEN
. | . |    Try (Unfold ‘mlnk‘ 0)
. | . |    THEN
. | . |    Analyze -1
. | . |    THEN
. | . |    Unfold ‘mlnk‘ -1
. | . |    THEN
. | . |    Analyze -2
. | . |    THEN
. | . |    Analyze -2
. | . |    THEN
. | . |    All Reduce
. | . |    THEN
. | . |    DoSubsume
. | . |
. | . .....subtype_rel .....
. | . 1. i : Id
. | . 2. k : Knd
. | . 3. l1 : IdLnk
. | . 4. ds : x:Id fp-> Type
. | . 5. da : a:Knd fp-> Type
. | . 6. f :
. | .    (tg:Id
. | .     ×State(ds)→ma-valtype(da; k)→
. | .     (da(rcv(l1; tg))?Void List)) List
```

```
. | . 7. source(l1) = i
. | . 8. w : World
. | . 9. FairFifo
. | . 10. FairFifo
. | . 11. ∀l@0:IdLnk, tg:Id.
. | .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. | .                         <ds,da,,,,<[<k,l1>],λx.f>,,,•>
. | .                       else  fi.da(rcv(l@0; tg))
. | . 12. ∀a:Action(i).
. | .     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. | . 13. ∀x:Id.
. | .     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. | .                     else Top fi
. | . 14. l : IdLnk
. | . 15. t : Id
. | . 16. w.M(l,t)
. | . 17. source(l) = i
. | . ⊢  (w.M(l,t)) ⊆r da(rcv(l; t))?Top
. | .   by InstHyp [l;t] -7 THEN SplitOnHypITE -1
. | .      THEN
. | .      RWO "deq_property<" -1
. | .      THEN
. | .      Unfold ‘ma-da‘ -2
. | .      THEN
. | .      Reduce -2
. |
. .\
. . 1. i : Id
. . 2. k : Knd
. . 3. l : IdLnk
. . 4. ds : x:Id fp-> Type
. . 5. da : a:Knd fp-> Type
. . 6. f :
. . |  (tg:Id
. . |   ×State(ds)→ma-valtype(da; k)→
. . |   (da(rcv(l; tg))?Void List)) List
. . 7. source(l) = i
. . 8. w : World
. . 9. p : FairFifo
. . 10. FairFifo
. . 11. ∀i@0,x:Id.
. . |   vartype(i@0;x) ⊆r if deq-member(IdDeq;x;1of(1of(if eqof(IdDeq)
. . |                                                      (i@0
. . |                                                      ,i)→
. . |                                                      <ds
. . |                                                      ,da
. . |                                                      ,
. . |                                                      ,
. . |                                                      ,
. . |                                                      ,<[<k,l>]
. . |                                                       ,λx.f>
. . |                                                      ,
. . |                                                      ,
. . |                                                      ,•>
. . |                                                    else  fi)))→
```

```
. . |                        2of(1of(if eqof(IdDeq)(i@0,i)→
. . |                                  <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . |                                else  fi))
. . |                        (x)
. . |                      else Top fi
. . 12. ∀i@0:Id, a:Action(i@0).
. . |   ¬isnull(a)
. . |   ⇒
. . |   (valtype(i@0;a) ⊆r if eqof(IdDeq)(i@0,i)→
. . |                          <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . |                        else  fi.da(kind(a)))
. . 13. ∀l@0:IdLnk, tg:Id.
. . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . |                       <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . |                     else  fi.da(rcv(l@0; tg))
. . 14. ∀i@0,x:Id.
. . |   deq-member(IdDeq;x;1of(1of(2of(2of(if eqof(IdDeq)(i@0,i)→
. . |                                          <ds
. . |                                          ,da
. . |                                          ,
. . |                                          ,
. . |                                          ,
. . |                                          ,<[<k,l>],λx.f>
. . |                                          ,
. . |                                          ,
. . |                                          ,·>
. . |                                        else  fi)))))
. . |   ⇒
. . |   s(i@0;0).x
. . |   =
. . |   2of(1of(2of(2of(if eqof(IdDeq)(i@0,i)→
. . |                       <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . |                     else  fi))))
. . |   (x)
. . 15. ∀i@0:Id, t:ℕ.
. . |   ¬isnull(a(i@0;t))
. . |   ⇒
. . |   (islocal(kind(a(i@0;t)))
. . |    ⇒
. . |    deq-member(IdDeq;act(kind(a(i@0;t)));1of(1of(2of(2of(2of(
. . |                                                    if eqof(IdDeq)
. . |                                                       (i@0
. . |                                                       ,i)→
. . |                                                       <ds
. . |                                                       ,da
. . |                                                       ,
. . |                                                       ,
. . |                                                       ,
. . |                                                       ,<[<k,l>]
. . |                                                        ,λx.f>
. . |                                                       ,
. . |                                                       ,
. . |                                                       ,·>
. . |                                                    else  fi))))))
. . |    ⇒
```

```
. . |    2of(1of(2of(2of(2of(if eqof(IdDeq)(i@0,i)→
. . |                          <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . |                       else  fi)))))
. . |    (act(kind(a(i@0;t)))
. . |    ,λx.s(i@0;t).x
. . |    ,val(a(i@0;t))))
. . |   & (∀x:Id.
. . |      deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<kind(...)
. . |                                                  ,x>;1of(1of(
. . |                                                      2of(2of(
. . |                                                      2of(2of(
. . |                                                      ...)))))))
. . |      ⇒
. . |      s(i@0;t+1).x
. . |      =
. . |      2of(1of(2of(2of(2of(if eqof(IdDeq)(i@0,i)→
. . |                              <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . |                           else  fi))))))
. . |      (<kind(a(i@0;t)),x>
. . |      ,λx.s(i@0;t).x
. . |      ,val(a(i@0;t))))
. . |   & (∀l@0:IdLnk.
. . |      deq-member(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq);...;...)
. . |      ⇒
. . |      withlnk(l@0;m(i@0;t))
. . |      =
. . |      if source(l@0) = i@0→
. . |         concat(map(λtgf.map(λx.
. . |         <1of(tgf)
. . |         ,x>;2of(tgf)
. . |             ((λx.s(i@0;t).x)
. . |             ,val(a(i@0;t))));2of(1of(2of(2of(2of(2of(2of(
. . |                                      if eqof(IdDeq)(i@0,i)→
. . |                                         <ds
. . |                                         ,da
. . |                                         ,
. . |                                         ,
. . |                                         ,
. . |                                         ,<[<k,l>],λx.f>
. . |                                         ,
. . |                                         ,
. . |                                         ,·>
. . |                                      else  fi)))))))
. . |                                (<kind(a(i@0;t)),l@0>)))
. . |      else nil fi)
. . |   & (∀x:Id.
. . |      ¬(deq-member(IdDeq;x;1of(1of(2of(2of(2of(2of(2of(2of(
. . |                                    if eqof(IdDeq)(i@0,i)→
. . |                                       <ds
. . |                                       ,da
. . |                                       ,
. . |                                       ,
. . |                                       ,
. . |                                       ,<[<k,l>],λx.f>
. . |                                       ,
```

```
. . |                                        ,
. . |                                        ,·>
. . |                                   else  fi)))))))))
. . |       ⇒
. . |       deq-member(KindDeq;kind(a(i@0;t));2of(
. . |                                        1of(2of(2of(2of(2of(
. . |                                            2of(2of(
. . |                                            if eqof(IdDeq)
. . |                                               (i@0
. . |                                               ,i)→
. . |                                               <ds
. . |                                               ,da
. . |                                               ,
. . |                                               ,
. . |                                               ,
. . |                                               ,<[<k,l>],λx.f>
. . |                                               ,
. . |                                               ,
. . |                                               ,·>
. . |                                            else  fi)))))))))
. . |                                        (x)))
. . |     ⇒
. . |     s(i@0;t).x = s(i@0;t+1).x)
. . |   & (∀l@0:IdLnk, tg:Id.
. . |     ¬(deq-member(product-deq(IdLnk;Id;IdLnkDeq;IdDeq);...;...)
. . |       ⇒
. . |       deq-member(KindDeq;kind(a(i@0;t));2of(
. . |                                        1of(2of(2of(2of(2of(
. . |                                            2of(2of(2of(
. . |                                            if eqof(IdDeq)
. . |                                               (i@0
. . |                                               ,i)→
. . |                                               <ds
. . |                                               ,da
. . |                                               ,
. . |                                               ,
. . |                                               ,
. . |                                               ,<[<k,l>],λx.f>
. . |                                               ,
. . |                                               ,
. . |                                               ,·>
. . |                                            else  fi)))))))))
. . |                                        (<l@0,tg>)))
. . |     ⇒
. . |     w-tagged(tg; onlnk(l@0;m(i@0;t))) = nil)
. . 16. ∀i@0,a:Id, t:ℕ.
. . |   ∃t':ℕ.
. . |   t≤t'
. . |   & ¬isnull(a(i@0;t')) & kind(a(i@0;t')) = locl(a)
. . |     ∨ ¬a declared in if eqof(IdDeq)(i@0,i)→
. . |                           <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . |                      else  fi
. . |     ∨ unsolvable if eqof(IdDeq)(i@0,i)→
. . |                        <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . |                 else  fi.pre(a,λx.s(i@0;t').x)
```

```
. . 17. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . |   & (∀e:E.
. . |      loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . |   & (∀e:E.
. . |      isrcv(e)
. . |      ⇒
. . |      lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . |   & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . |      ((λl,tg. da(rcv(l; tg))?Top)))
. . ⊢  ∀e:E.
. . |  loc(e) = i
. . |  ⇒
. . |  kind(e) = k
. . |  ⇒
. . |  sends(l;e) = tagged-messages(l;λz.(z when e);val(e);f)
. . | by let maops
. . |        ≡ [‘ma-ds‘
. . |          ;‘ma-da‘
. . |          ;‘ma-din‘
. . |          ;‘ma-dout‘
. . |          ;‘ma-pre‘
. . |          ;‘ma-npre‘
. . |          ;‘ma-ef‘
. . |          ;‘ma-init‘
. . |          ;‘ma-decla‘] in
. . |    AssertBY ((eqof(IdDeq)(i,i)) ∼ true2)
. . |    (BackThru
. . |     Thm* ∀d:EqDecider(A), i:A. (eqof(d)(i,i)) ∼ true2
. . |     {eqof_eq_btrue})
. . |    THEN
. . |    AllHyps
. . |    (λh.
. . |     InstHyp [i] h THENA Complete Auto
. . |     THEN
. . |     Subst ((eqof(IdDeq)(i,i)) ∼ true2) -1
. . |     THENL
. . |     [Trivial
. . |     ;Thin h THEN Reduce -1
. . |      THEN
. . |      Repeat (Unfolds maops -1 THEN Reduce -1)])
. . |
. . 1. i : Id
. . 2. k : Knd
. . 3. l : IdLnk
. . 4. ds : x:Id fp-> Type
. . 5. da : a:Knd fp-> Type
. . 6. f :
. . |  (tg:Id
. . |   ×State(ds)→ma-valtype(da; k)→
. . |   (da(rcv(l; tg))?Void List)) List
. . 7. source(l) = i
. . 8. w : World
. . 9. p : FairFifo
. . 10. FairFifo
. . 11. ∀l@0:IdLnk, tg:Id.
```

```
. . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . |                        <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . |                    else  fi.da(rcv(l@0; tg))
. . 12. (eqof(IdDeq)(i,i)) ∼ true2
. . 13. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . |   & (∀e:E.
. . |      loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . |   & (∀e:E.
. . |      isrcv(e)
. . |      ⇒
. . |      lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . |   & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . |      ((λl,tg. da(rcv(l; tg))?Top)))
. . 14. ∀a:Id, t:ℕ.
. . |   ∃t':ℕ.
. . |   t≤t'
. . |   & ¬isnull(a(i;t')) & kind(a(i;t')) = locl(a)
. . |     ∨ ¬locl(a) ∈ dom(da)
. . |     ∨ P != (a) ==> ∀v:da(locl(a))?Top. ¬P((λx.s(i;t').x),v)
. . 15. ∀t:ℕ.
. . |   ¬isnull(a(i;t))
. . |   ⇒
. . |   (islocal(kind(a(i;t)))
. . |    ⇒
. . |    deq-member(IdDeq;act(kind(a(i;t)));1of())
. . |    ⇒
. . |    2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . |   & (∀x:Id.
. . |      deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<kind(a(i;t))
. . |                                                    ,x>;1of())
. . |      ⇒
. . |      s(i;t+1).x
. . |      =
. . |      2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . |   & (∀l@0:IdLnk.
. . |      (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . |       (<k,l>
. . |       ,<kind(a(i;t)),l@0>) ∨2 false2)
. . |      ⇒
. . |      withlnk(l@0;m(i;t))
. . |      =
. . |      if source(l@0) = i→
. . |         concat(map(λtgf.map(λx.<1of(tgf),x>;2of(tgf)
. . |                                          ((λx.s(i;t).x)
. . |                                          ,val(a(i;t))));f))
. . |      else nil fi)
. . |   & (∀x:Id.
. . |      ¬(deq-member(IdDeq;x;1of())
. . |        ⇒
. . |        deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . |      ⇒
. . |      s(i;t).x = s(i;t+1).x)
. . |   & (∀l@0:IdLnk, tg:Id.
. . |      ¬(deq-member(product-deq(IdLnk;Id;IdLnkDeq;IdDeq);...;1of(
. . |                                                          ))
```

```
. . |        ⇒
. . |        deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
. . |      ⇒
. . |      w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
. . 16. ∀x:Id. deq-member(IdDeq;x;1of()) ⇒ s(i;0).x = 2of()(x)
. . 17. ∀a:Action(i). ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . 18. ∀x:Id.
. . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. . |                   else Top fi
. . ⊢  ∀e:E.
. . |  loc(e) = i
. . |  ⇒
. . |  kind(e) = k
. . |  ⇒
. . |  sends(l;e) = tagged-messages(l;λz.(z when e);val(e);f)
. . | by Thin -3 THEN Thin -4
. . |
. . 1. i : Id
. . 2. k : Knd
. . 3. l : IdLnk
. . 4. ds : x:Id fp-> Type
. . 5. da : a:Knd fp-> Type
. . 6. f :
. . |  (tg:Id
. . |   ×State(ds)→ma-valtype(da; k)→
. . |   (da(rcv(l; tg))?Void List)) List
. . 7. source(l) = i
. . 8. w : World
. . 9. p : FairFifo
. . 10. FairFifo
. . 11. ∀l@0:IdLnk, tg:Id.
. . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . |                       <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . |                    else  fi.da(rcv(l@0; tg))
. . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . 13. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . |   & (∀e:E.
. . |      loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . |   & (∀e:E.
. . |      isrcv(e)
. . |      ⇒
. . |      lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . |   & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . |      ((λl,tg. da(rcv(l; tg))?Top)))
. . 14. ∀t:ℕ.
. . |   ¬isnull(a(i;t))
. . |   ⇒
. . |   (islocal(kind(a(i;t)))
. . |    ⇒
. . |    deq-member(IdDeq;act(kind(a(i;t)));1of())
. . |    ⇒
. . |    2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . |   & (∀x:Id.
. . |      deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<kind(a(i;t))
. . |                                                    ,x>;1of())
```

```
. . |      ⇒
. . |      s(i;t+1).x
. . |      =
. . |      2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . |   & (∀l@0:IdLnk.
. . |      (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . |       (<k,l>
. . |       ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . |      ⇒
. . |      withlnk(l@0;m(i;t))
. . |      =
. . |      if source(l@0) = i→
. . |         concat(map(λtgf.map(λx.<1of(tgf),x>;2of(tgf)
. . |                                          ((λx.s(i;t).x)
. . |                                          ,val(a(i;t))));f))
. . |      else nil fi)
. . |   & (∀x:Id.
. . |      ¬(deq-member(IdDeq;x;1of())
. . |        ⇒
. . |        deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . |      ⇒
. . |      s(i;t).x = s(i;t+1).x)
. . |   & (∀l@0:IdLnk, tg:Id.
. . |      ¬(deq-member(product-deq(IdLnk;Id;IdLnkDeq;IdDeq);...;1of(
. . |                                                          ))
. . |        ⇒
. . |        deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
. . |      ⇒
. . |      w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
. . 15. ∀a:Action(i). ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . 16. ∀x:Id.
. . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. . |                   else Top fi
. . ⊢  ∀e:E.
. . |  loc(e) = i
. . |  ⇒
. . |  kind(e) = k
. . |  ⇒
. . |  sends(l;e) = tagged-messages(l;λz.(z when e);val(e);f)
. . | by Assert (IdDeq ∈ EqDecider(Id))
. . |
. . |\
. . | .....assertion .....
. . | 1. i : Id
. . | 2. k : Knd
. . | 3. l : IdLnk
. . | 4. ds : x:Id fp-> Type
. . | 5. da : a:Knd fp-> Type
. . | 6. f :
. . |    (tg:Id
. . |     ×State(ds)→ma-valtype(da; k)→
. . |     (da(rcv(l; tg))?Void List)) List
. . | 7. source(l) = i
. . | 8. w : World
. . | 9. p : FairFifo
```

```
. . | 10. FairFifo
. . | 11. ∀l@0:IdLnk, tg:Id.
. . |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . |                          <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . |                       else  fi.da(rcv(l@0; tg))
. . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . | 13. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . |     & (∀e:E.
. . |        loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . |     & (∀e:E.
. . |        isrcv(e)
. . |        ⇒
. . |        lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . |     & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . |        ((λl,tg. da(rcv(l; tg))?Top)))
. . | 14. ∀t:ℕ.
. . |     ¬isnull(a(i;t))
. . |     ⇒
. . |     (islocal(kind(a(i;t)))
. . |      ⇒
. . |      deq-member(IdDeq;act(kind(a(i;t)));1of())
. . |      ⇒
. . |      2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . |     & (∀x:Id.
. . |        deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<kind(...)
. . |                                                     ,x>;1of())
. . |        ⇒
. . |        s(i;t+1).x
. . |        =
. . |        2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . |     & (∀l@0:IdLnk.
. . |        (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . |         (<k,l>
. . |         ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . |        ⇒
. . |        withlnk(l@0;m(i;t))
. . |        =
. . |        if source(l@0) = i→
. . |           concat(map(λtgf.map(λx.<1of(tgf),x>;2of(tgf)
. . |                                              ((λx.s(i;t).x)
. . |                                              ,val(a(i;t))));f))
. . |        else nil fi)
. . |     & (∀x:Id.
. . |        ¬(deq-member(IdDeq;x;1of())
. . |          ⇒
. . |          deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . |        ⇒
. . |        s(i;t).x = s(i;t+1).x)
. . |     & (∀l@0:IdLnk, tg:Id.
. . |        ¬(deq-member(product-deq(IdLnk;Id;IdLnkDeq;...);...;1of(
. . |                                                           ))
. . |          ⇒
. . |          deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
. . |        ⇒
. . |        w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
```

```
. . | 15. ∀a:Action(i).
. . |     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . | 16. ∀x:Id.
. . |     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. . |                      else Top fi
. . | ⊢  IdDeq ∈ EqDecider(Id)
. . |   by SubsumeC EqDecider(Id)
. . |      THEN
. . |      Try
. . |      (BackThru
. . |       Thm* strong-subtype(A;B) ⇒ (EqDecider(B) ⊆r EqDecider(A))
. . |       {strong_DASH_subtype_DASH_deq_DASH_subtype}
. . |       THEN
. . |       All (Unfold `dstype`)
. . |       THEN
. . |       All Reduce
. . |       THEN
. . |       Trivial)
. . |      THEN
. . |      AllHyps (λi.Unfold_Wld i THEN Reduce 0 THEN Complete Auto)
. . |
. . .\
. . . 1. i : Id
. . . 2. k : Knd
. . . 3. l : IdLnk
. . . 4. ds : x:Id fp-> Type
. . . 5. da : a:Knd fp-> Type
. . . 6. f :
. . . |  (tg:Id
. . . |   ×State(ds)→ma-valtype(da; k)→
. . . |   (da(rcv(l; tg))?Void List)) List
. . . 7. source(l) = i
. . . 8. w : World
. . . 9. p : FairFifo
. . . 10. FairFifo
. . . 11. ∀l@0:IdLnk, tg:Id.
. . . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . |                        <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . |                     else  fi.da(rcv(l@0; tg))
. . . 12. (eqof(IdDeq)(i,i)) ∼ true2
. . . 13. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . . |   & (∀e:E.
. . . |      loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . |   & (∀e:E.
. . . |      isrcv(e)
. . . |      ⇒
. . . |      lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . |   & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . |      ((λl,tg. da(rcv(l; tg))?Top)))
. . . 14. ∀t:ℕ.
. . . |   ¬isnull(a(i;t))
. . . |   ⇒
. . . |   (islocal(kind(a(i;t)))
. . . |    ⇒
. . . |    deq-member(IdDeq;act(kind(a(i;t)));1of())
```

```
. . . |    ⇒
. . . |    2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . . |   & (∀x:Id.
. . . |      deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<kind(...)
. . . |                                                    ,x>;1of())
. . . |      ⇒
. . . |      s(i;t+1).x
. . . |      =
. . . |      2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . . |   & (∀l@0:IdLnk.
. . . |      (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . |       (<k,l>
. . . |       ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . |      ⇒
. . . |      withlnk(l@0;m(i;t))
. . . |      =
. . . |      if source(l@0) = i→
. . . |         concat(map(λtgf.map(λx.<1of(tgf),x>;2of(tgf)
. . . |                                           ((λx.s(i;t).x)
. . . |                                           ,val(a(i;t))));f))
. . . |      else nil fi)
. . . |   & (∀x:Id.
. . . |      ¬(deq-member(IdDeq;x;1of())
. . . |        ⇒
. . . |        deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . . |      ⇒
. . . |      s(i;t).x = s(i;t+1).x)
. . . |   & (∀l@0:IdLnk, tg:Id.
. . . |      ¬(deq-member(product-deq(IdLnk;Id;IdLnkDeq;...);...;1of(
. . . |                                                        ))
. . . |        ⇒
. . . |        deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
. . . |      ⇒
. . . |      w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
. . . 15. ∀a:Action(i).
. . . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . 16. ∀x:Id.
. . . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. . . |                  else Top fi
. . . 17. IdDeq ∈ EqDecider(Id)
. . . ⊢  ∀e:E.
. . . |  loc(e) = i
. . . |  ⇒
. . . |  kind(e) = k
. . . |  ⇒
. . . |  sends(l;e) = tagged-messages(l;λz.(z when e);val(e);f)
. . . | by Analyze 0
. . . |
. . . |\
. . . | 1. i : Id
. . . | 2. k : Knd
. . . | 3. l : IdLnk
. . . | 4. ds : x:Id fp-> Type
. . . | 5. da : a:Knd fp-> Type
. . . | 6. f :
```

```
. . . | |  (tg:Id
. . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | |   (da(rcv(l; tg))?Void List)) List
. . . | 7. source(l) = i
. . . | 8. w : World
. . . | 9. p : FairFifo
. . . | 10. FairFifo
. . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | |                        <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | |                    else  fi.da(rcv(l@0; tg))
. . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | 13. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . . | |   & (∀e:E.
. . . | |      loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . | |   & (∀e:E.
. . . | |      isrcv(e)
. . . | |      ⇒
. . . | |      lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . | |   & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | |      ((λl,tg. da(rcv(l; tg))?Top)))
. . . | 14. ∀t:ℕ.
. . . | |   ¬isnull(a(i;t))
. . . | |   ⇒
. . . | |   (islocal(kind(a(i;t)))
. . . | |    ⇒
. . . | |    deq-member(IdDeq;act(kind(a(i;t)));1of())
. . . | |    ⇒
. . . | |    2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . . | |   & (∀x:Id.
. . . | |      deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<kind(...)
. . . | |                                                   ,x>;1of())
. . . | |      ⇒
. . . | |      s(i;t+1).x
. . . | |      =
. . . | |      2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . . | |   & (∀l@0:IdLnk.
. . . | |      (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | |       (<k,l>
. . . | |       ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | |      ⇒
. . . | |      withlnk(l@0;m(i;t))
. . . | |      =
. . . | |      if source(l@0) = i→
. . . | |         concat(map(λtgf.map(λx.<1of(tgf)
. . . | |                                 ,x>;2of(tgf)
. . . | |                                     ((λx.s(i;t).x)
. . . | |                                     ,val(a(i;t))));f))
. . . | |      else nil fi)
. . . | |   & (∀x:Id.
. . . | |      ¬(deq-member(IdDeq;x;1of())
. . . | |        ⇒
. . . | |        deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . . | |      ⇒
. . . | |      s(i;t).x = s(i;t+1).x)
```

```
. . . | |   & (∀l@0:IdLnk, tg:Id.
. . . | |      ¬(deq-member(product-deq(IdLnk;Id;...;IdDeq);...;1of())
. . . | |        ⇒
. . . | |        deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
. . . | |      ⇒
. . . | |      w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
. . . | 15. ∀a:Action(i).
. . . | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | 16. ∀x:Id.
. . . | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. . . | |                  else Top fi
. . . | 17. IdDeq ∈ EqDecider(Id)
. . . | 18. e : E
. . . | ⊢  loc(e) = i
. . . | |  ⇒
. . . | |  kind(e) = k
. . . | |  ⇒
. . . | |  sends(l;e) = tagged-messages(l;λz.(z when e);val(e);f)
. . . | | by AssertBY True
. . . | |    (Auto THEN All (Unfold `dstype`) THEN All Reduce
. . . | |     THEN
. . . | |     Try Trivial)
. . . | |    THEN
. . . | |    RepeatFor 2 (Analyze 0 THENA (Auto THEN D_ES_Subtype))
. . . | |
. . . | 1. i : Id
. . . | 2. k : Knd
. . . | 3. l : IdLnk
. . . | 4. ds : x:Id fp-> Type
. . . | 5. da : a:Knd fp-> Type
. . . | 6. f :
. . . | |  (tg:Id
. . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | |   (da(rcv(l; tg))?Void List)) List
. . . | 7. source(l) = i
. . . | 8. w : World
. . . | 9. p : FairFifo
. . . | 10. FairFifo
. . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | |                      <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . . | |                   else  fi.da(rcv(l@0; tg))
. . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | 13. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . . | |   & (∀e:E.
. . . | |      loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . | |   & (∀e:E.
. . . | |      isrcv(e)
. . . | |      ⇒
. . . | |      lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . | |   & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | |      ((λl,tg. da(rcv(l; tg))?Top)))
. . . | 14. ∀t:ℕ.
. . . | |   ¬isnull(a(i;t))
. . . | |   ⇒
```

```
. . . | |   (islocal(kind(a(i;t)))
. . . | |    ⇒
. . . | |    deq-member(IdDeq;act(kind(a(i;t)));1of())
. . . | |    ⇒
. . . | |    2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . . | |   & (∀x:Id.
. . . | |      deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<kind(...)
. . . | |                                                  ,x>;1of())
. . . | |      ⇒
. . . | |      s(i;t+1).x
. . . | |      =
. . . | |      2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . . | |   & (∀l@0:IdLnk.
. . . | |      (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | |       (<k,l>
. . . | |       ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | |      ⇒
. . . | |      withlnk(l@0;m(i;t))
. . . | |      =
. . . | |      if source(l@0) = i→
. . . | |         concat(map(λtgf.map(λx.<1of(tgf)
. . . | |                                ,x>;2of(tgf)
. . . | |                                    ((λx.s(i;t).x)
. . . | |                                    ,val(a(i;t))));f))
. . . | |      else nil fi)
. . . | |   & (∀x:Id.
. . . | |      ¬(deq-member(IdDeq;x;1of())
. . . | |        ⇒
. . . | |        deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . . | |      ⇒
. . . | |      s(i;t).x = s(i;t+1).x)
. . . | |   & (∀l@0:IdLnk, tg:Id.
. . . | |      ¬(deq-member(product-deq(IdLnk;Id;...;IdDeq);...;1of())
. . . | |        ⇒
. . . | |        deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
. . . | |      ⇒
. . . | |      w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
. . . | 15. ∀a:Action(i).
. . . | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | 16. ∀x:Id.
. . . | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. . . | |                  else Top fi
. . . | 17. IdDeq ∈ EqDecider(Id)
. . . | 18. e : E
. . . | 19. True
. . . | 20. loc(e) = i
. . . | 21. kind(e) = k
. . . | ⊢  sends(l;e) = tagged-messages(l;λz.(z when e);val(e);f)
. . . | | by Analyze -4 THEN Analyze -5 THEN RenameVar ‘j’ -6
. . . | |    THEN
. . . | |    RenameVar ‘t’ -5
. . . | |    THEN
. . . | |    All Reduce
. . . | |    THEN
. . . | |    Decide isnull(a(j;t))
```

```
. . . | |    THEN
. . . | |    Try (Assert False THEN Unhide THEN Complete Auto)
. . . | |    THEN
. . . | |    Thin -1
. . . | |
. . . | 1. i : Id
. . . | 2. k : Knd
. . . | 3. l : IdLnk
. . . | 4. ds : x:Id fp-> Type
. . . | 5. da : a:Knd fp-> Type
. . . | 6. f :
. . . | |  (tg:Id
. . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | |   (da(rcv(l; tg))?Void List)) List
. . . | 7. source(l) = i
. . . | 8. w : World
. . . | 9. p : FairFifo
. . . | 10. FairFifo
. . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | |                        <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | |                    else  fi.da(rcv(l@0; tg))
. . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | 13. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . . | |   & (∀e:E.
. . . | |      loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . | |   & (∀e:E.
. . . | |      isrcv(e)
. . . | |      ⇒
. . . | |      lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . | |   & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | |      ((λl,tg. da(rcv(l; tg))?Top)))
. . . | 14. ∀t:ℕ.
. . . | |   ¬isnull(a(i;t))
. . . | |   ⇒
. . . | |   (islocal(kind(a(i;t)))
. . . | |    ⇒
. . . | |    deq-member(IdDeq;act(kind(a(i;t)));1of())
. . . | |    ⇒
. . . | |    2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . . | |   & (∀x:Id.
. . . | |      deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<kind(...)
. . . | |                                                    ,x>;1of())
. . . | |      ⇒
. . . | |      s(i;t+1).x
. . . | |      =
. . . | |      2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . . | |   & (∀l@0:IdLnk.
. . . | |      (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | |       (<k,l>
. . . | |       ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | |      ⇒
. . . | |      withlnk(l@0;m(i;t))
. . . | |      =
. . . | |      if source(l@0) = i→
```

```
. . . | |         concat(map(λtgf.map(λx.<1of(tgf)
. . . | |                                ,x>;2of(tgf)
. . . | |                                    ((λx.s(i;t).x)
. . . | |                                    ,val(a(i;t))));f))
. . . | |      else nil fi)
. . . | |   & (∀x:Id.
. . . | |      ¬(deq-member(IdDeq;x;1of())
. . . | |        ⇒
. . . | |        deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . . | |      ⇒
. . . | |      s(i;t).x = s(i;t+1).x)
. . . | |   & (∀l@0:IdLnk, tg:Id.
. . . | |      ¬(deq-member(product-deq(IdLnk;Id;...;IdDeq);...;1of())
. . . | |        ⇒
. . . | |        deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
. . . | |      ⇒
. . . | |      w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
. . . | 15. ∀a:Action(i).
. . . | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | 16. ∀x:Id.
. . . | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. . . | |                  else Top fi
. . . | 17. IdDeq ∈ EqDecider(Id)
. . . | 18. j : Id
. . . | 19. t : ℕ
. . . | 20. ¬isnull(a(j;t))
. . . | 21. True
. . . | 22. j = i
. . . | 23. kind(<j,t>) = k
. . . | ⊢  sends(l;<j,t>)
. . . | |  =
. . . | |  tagged-messages(l;λz.(z when <j,t>);val(<j,t>);f)
. . . | | by AssertBY (¬isnull(a(i;t))) (RevHypSubst -2 0)
. . . | |
. . . | 1. i : Id
. . . | 2. k : Knd
. . . | 3. l : IdLnk
. . . | 4. ds : x:Id fp-> Type
. . . | 5. da : a:Knd fp-> Type
. . . | 6. f :
. . . | |  (tg:Id
. . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | |   (da(rcv(l; tg))?Void List)) List
. . . | 7. source(l) = i
. . . | 8. w : World
. . . | 9. p : FairFifo
. . . | 10. FairFifo
. . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | |                      <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | |                   else  fi.da(rcv(l@0; tg))
. . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | 13. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . . | |   & (∀e:E.
. . . | |      loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
```

```
. . . | |   & (∀e:E.
. . . | |      isrcv(e)
. . . | |      ⇒
. . . | |      lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . | |   & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | |      ((λl,tg. da(rcv(l; tg))?Top)))
. . . | 14. ∀t:ℕ.
. . . | |   ¬isnull(a(i;t))
. . . | |   ⇒
. . . | |   (islocal(kind(a(i;t)))
. . . | |    ⇒
. . . | |    deq-member(IdDeq;act(kind(a(i;t)));1of())
. . . | |    ⇒
. . . | |    2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . . | |   & (∀x:Id.
. . . | |      deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<kind(...)
. . . | |                                                  ,x>;1of())
. . . | |      ⇒
. . . | |      s(i;t+1).x
. . . | |      =
. . . | |      2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . . | |   & (∀l@0:IdLnk.
. . . | |      (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | |       (<k,l>
. . . | |       ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | |      ⇒
. . . | |      withlnk(l@0;m(i;t))
. . . | |      =
. . . | |      if source(l@0) = i→
. . . | |         concat(map(λtgf.map(λx.<1of(tgf)
. . . | |                                ,x>;2of(tgf)
. . . | |                                    ((λx.s(i;t).x)
. . . | |                                    ,val(a(i;t))));f))
. . . | |      else nil fi)
. . . | |   & (∀x:Id.
. . . | |      ¬(deq-member(IdDeq;x;1of())
. . . | |        ⇒
. . . | |        deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . . | |      ⇒
. . . | |      s(i;t).x = s(i;t+1).x)
. . . | |   & (∀l@0:IdLnk, tg:Id.
. . . | |      ¬(deq-member(product-deq(IdLnk;Id;...;IdDeq);...;1of())
. . . | |        ⇒
. . . | |        deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
. . . | |      ⇒
. . . | |      w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
. . . | 15. ∀a:Action(i).
. . . | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | 16. ∀x:Id.
. . . | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. . . | |                   else Top fi
. . . | 17. IdDeq ∈ EqDecider(Id)
. . . | 18. j : Id
. . . | 19. t : ℕ
. . . | 20. ¬isnull(a(j;t))
```

```
. . . | 21. True
. . . | 22. j = i

. . . | 23. kind(<j,t>) = k
. . . | 24. ¬isnull(a(i;t))
. . . | ⊢  sends(l;<j,t>)
. . . | |  =
. . . | |  tagged-messages(l;λz.(z when <j,t>);val(<j,t>);f)
. . . | | by Assert ((λx.s(i;t).x) ∈ State(ds))
. . . | |
. . . | |\
. . . | | .....assertion .....
. . . | | 1. i : Id
. . . | | 2. k : Knd
. . . | | 3. l : IdLnk
. . . | | 4. ds : x:Id fp-> Type
. . . | | 5. da : a:Knd fp-> Type
. . . | | 6. f :
. . . | |    (tg:Id
. . . | |     ×State(ds)→ma-valtype(da; k)→
. . . | |     (da(rcv(l; tg))?Void List)) List
. . . | | 7. source(l) = i
. . . | | 8. w : World
. . . | | 9. p : FairFifo
. . . | | 10. FairFifo
. . . | | 11. ∀l@0:IdLnk, tg:Id.
. . . | |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | |                          <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | |                       else  fi.da(rcv(l@0; tg))
. . . | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | | 13. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . . | |     & (∀e:E.
. . . | |        loc(e) = i
. . . | |        ⇒
. . . | |        (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . | |     & (∀e:E.
. . . | |        isrcv(e)
. . . | |        ⇒
. . . | |        lnk(e) = l
. . . | |        ⇒
. . . | |        (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . | |     & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | |        ((λl,tg. da(rcv(l; tg))?Top)))
. . . | | 14. ∀t:ℕ.
. . . | |     ¬isnull(a(i;t))
. . . | |     ⇒
. . . | |     (islocal(kind(a(i;t)))
. . . | |      ⇒
. . . | |      deq-member(IdDeq;act(kind(a(i;t)));1of())
. . . | |      ⇒
. . . | |      2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . . | |     & (∀x:Id.
. . . | |        deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<...
. . . | |                                                    ,x>;1of(
. . . | |                                                        ))
. . . | |        ⇒
```

```
. . . | |       s(i;t+1).x
. . . | |       =
. . . | |       2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . . | |    & (∀l@0:IdLnk.
. . . | |       (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | |        (<k,l>
. . . | |        ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | |       ⇒
. . . | |       withlnk(l@0;m(i;t))
. . . | |       =
. . . | |       if source(l@0) = i→
. . . | |          concat(map(λtgf.map(λx.<1of(tgf)
. . . | |                                  ,x>;2of(tgf)
. . . | |                                      ((λx.s(i;t).x)
. . . | |                                      ,val(a(i;t))));f))
. . . | |       else nil fi)
. . . | |    & (∀x:Id.
. . . | |       ¬(deq-member(IdDeq;x;1of())
. . . | |         ⇒
. . . | |         deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . . | |       ⇒
. . . | |       s(i;t).x = s(i;t+1).x)
. . . | |    & (∀l@0:IdLnk, tg:Id.
. . . | |       ¬(deq-member(product-deq(IdLnk;Id;...;IdDeq);...;1of(
. . . | |                                                        ))
. . . | |         ⇒
. . . | |         deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
. . . | |       ⇒
. . . | |       w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
. . . | 15. ∀a:Action(i).
. . . | |    ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | 16. ∀x:Id.
. . . | |    vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | |                       2of(ds)(x)
. . . | |                    else Top fi
. . . | 17. IdDeq ∈ EqDecider(Id)
. . . | 18. j : Id
. . . | 19. t : ℕ
. . . | 20. ¬isnull(a(j;t))
. . . | 21. True
. . . | 22. j = i
. . . | 23. kind(<j,t>) = k
. . . | 24. ¬isnull(a(i;t))
. . . | ⊢  (λx.s(i;t).x) ∈ State(ds)
. . . |   by Unfold ‘ma-state‘ 0 THEN DoSubsume
. . . |      THEN
. . . |      Try BackThruSomeHyp

. . . |      THEN
. . . |      Try D_ES_Subtype
. . . |
. . . | .\
. . . | . 1. i : Id
. . . | . 2. k : Knd
. . . | . 3. l : IdLnk
. . . | . 4. ds : x:Id fp-> Type
```

```
. . . | . 5. da : a:Knd fp-> Type
. . . | . 6. f :
. . . | . |  (tg:Id
. . . | . |   ×State(ds)→ma-valtype(da; k)→
. . . | . |   (da(rcv(l; tg))?Void List)) List
. . . | . 7. source(l) = i
. . . | . 8. w : World
. . . | . 9. p : FairFifo
. . . | . 10. FairFifo
. . . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . |                       <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . . | . |                    else  fi.da(rcv(l@0; tg))
. . . | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . 13. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . . | . |   & (∀e:E.
. . . | . |      loc(e) = i
. . . | . |      ⇒
. . . | . |      (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . | . |   & (∀e:E.
. . . | . |      isrcv(e)
. . . | . |      ⇒
. . . | . |      lnk(e) = l
. . . | . |      ⇒
. . . | . |      (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . | . |   & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . |      ((λl,tg. da(rcv(l; tg))?Top)))
. . . | . 14. ∀t:ℕ.
. . . | . |   ¬isnull(a(i;t))
. . . | . |   ⇒
. . . | . |   (islocal(kind(a(i;t)))
. . . | . |    ⇒
. . . | . |    deq-member(IdDeq;act(kind(a(i;t)));1of())
. . . | . |    ⇒
. . . | . |    2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . . | . |   & (∀x:Id.
. . . | . |      deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<...
. . . | . |                                                  ,x>;1of(
. . . | . |                                                      ))
. . . | . |      ⇒
. . . | . |      s(i;t+1).x
. . . | . |      =
. . . | . |      2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . . | . |   & (∀l@0:IdLnk.
. . . | . |      (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . |       (<k,l>
. . . | . |       ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . |      ⇒
. . . | . |      withlnk(l@0;m(i;t))
. . . | . |      =
. . . | . |      if source(l@0) = i→
. . . | . |         concat(map(λtgf.map(λx.<1of(tgf)
. . . | . |                                 ,x>;2of(tgf)
. . . | . |                                     ((λx.s(i;t).x)
. . . | . |                                     ,val(a(i;t))));f))
```

```
. . . | . |      else nil fi)
. . . | . |   & (∀x:Id.
. . . | . |      ¬(deq-member(IdDeq;x;1of())
. . . | . |        ⇒
. . . | . |        deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . . | . |      ⇒
. . . | . |      s(i;t).x = s(i;t+1).x)
. . . | . |   & (∀l@0:IdLnk, tg:Id.
. . . | . |      ¬(deq-member(product-deq(IdLnk;Id;...;IdDeq);...;1of(
. . . | . |                                                      ))
. . . | . |        ⇒
. . . | . |        deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
. . . | . |      ⇒
. . . | . |      w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
. . . | . 15. ∀a:Action(i).
. . . | . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . 16. ∀x:Id.
. . . | . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . |                      2of(ds)(x)
. . . | . |                  else Top fi
. . . | . 17. IdDeq ∈ EqDecider(Id)
. . . | . 18. j : Id
. . . | . 19. t : ℕ
. . . | . 20. ¬isnull(a(j;t))
. . . | . 21. True
. . . | . 22. j = i
. . . | . 23. kind(<j,t>) = k
. . . | . 24. ¬isnull(a(i;t))
. . . | . 25. (λx.s(i;t).x) ∈ State(ds)
. . . | . ⊢  sends(l;<j,t>)
. . . | . |  =
. . . | . |  tagged-messages(l;λz.(z when <j,t>);val(<j,t>);f)
. . . | . | by Unfolds [‘w-when‘;‘es-val‘;‘es-LnkTag-deq‘] 0
. . . | . |    THEN
. . . | . |    Reduce 0
. . . | . |    THEN
. . . | . |    Unfold ‘w-eval‘ 0
. . . | . |    THEN
. . . | . |    Unfold ‘w-act‘ 0
. . . | . |    THEN
. . . | . |    Reduce 0
. . . | . |
. . . | . 1. i : Id
. . . | . 2. k : Knd
. . . | . 3. l : IdLnk
. . . | . 4. ds : x:Id fp-> Type
. . . | . 5. da : a:Knd fp-> Type
. . . | . 6. f :
. . . | . |  (tg:Id
. . . | . |   ×State(ds)→ma-valtype(da; k)→
. . . | . |   (da(rcv(l; tg))?Void List)) List
. . . | . 7. source(l) = i
. . . | . 8. w : World
. . . | . 9. p : FairFifo
. . . | . 10. FairFifo
```

```
. . . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . |                       <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . . | . |                   else  fi.da(rcv(l@0; tg))
. . . | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . 13. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . . | . |   & (∀e:E.
. . . | . |      loc(e) = i
. . . | . |      ⇒
. . . | . |      (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . | . |   & (∀e:E.
. . . | . |      isrcv(e)
. . . | . |      ⇒
. . . | . |      lnk(e) = l
. . . | . |      ⇒
. . . | . |      (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . | . |   & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . |      ((λl,tg. da(rcv(l; tg))?Top)))
. . . | . 14. ∀t:ℕ.
. . . | . |   ¬isnull(a(i;t))
. . . | . |   ⇒
. . . | . |   (islocal(kind(a(i;t)))
. . . | . |    ⇒
. . . | . |    deq-member(IdDeq;act(kind(a(i;t)));1of())
. . . | . |    ⇒
. . . | . |    2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . . | . |   & (∀x:Id.
. . . | . |      deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<...
. . . | . |                                                  ,x>;1of(
. . . | . |                                                      ))
. . . | . |      ⇒
. . . | . |      s(i;t+1).x
. . . | . |      =
. . . | . |      2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . . | . |   & (∀l@0:IdLnk.
. . . | . |      (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . |       (<k,l>
. . . | . |       ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . |      ⇒
. . . | . |      withlnk(l@0;m(i;t))
. . . | . |      =
. . . | . |      if source(l@0) = i→
. . . | . |         concat(map(λtgf.map(λx.<1of(tgf)
. . . | . |                                ,x>;2of(tgf)
. . . | . |                                    ((λx.s(i;t).x)
. . . | . |                                    ,val(a(i;t))));f))
. . . | . |      else nil fi)
. . . | . |   & (∀x:Id.
. . . | . |      ¬(deq-member(IdDeq;x;1of())
. . . | . |        ⇒
. . . | . |        deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . . | . |      ⇒
. . . | . |      s(i;t).x = s(i;t+1).x)
. . . | . |   & (∀l@0:IdLnk, tg:Id.
. . . | . |      ¬(deq-member(product-deq(IdLnk;Id;...;IdDeq);...;1of(
```

```
. . . | . |                                                          ))
. . . | . |        ⇒
. . . | . |        deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
. . . | . |      ⇒
. . . | . |      w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
. . . | . 15. ∀a:Action(i).
. . . | . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . 16. ∀x:Id.
. . . | . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . |                      2of(ds)(x)
. . . | . |                  else Top fi
. . . | . 17. IdDeq ∈ EqDecider(Id)
. . . | . 18. j : Id
. . . | . 19. t : ℕ
. . . | . 20. ¬isnull(a(j;t))
. . . | . 21. True
. . . | . 22. j = i
. . . | . 23. kind(<j,t>) = k
. . . | . 24. ¬isnull(a(i;t))
. . . | . 25. (λx.s(i;t).x) ∈ State(ds)
. . . | . ⊢  sends(l;<j,t>)
. . . | . |  =
. . . | . |  tagged-messages(l;λz.s(j;t).z;val(a(j;t));f)
. . . | . | by ExRepD
. . . | . |    THEN
. . . | . |    AllHyps
. . . | . |    (λh.

. . . | . |     InstHyp [t] h THENA Complete Auto
. . . | . |     THEN
. . . | . |     RepeatFor 2 (Analyze -1 THEN Thin -2)
. . . | . |     THEN
. . . | . |     Analyze -1
. . . | . |     THEN
. . . | . |     Thin -1
. . . | . |     THEN
. . . | . |     Thin h)
. . . | . |    THEN
. . . | . |    InstHyp [l] -1
. . . | . |    THENA
. . . | . |    Try PWTrivial
. . . | . |
. . . | . |\
. . . | . | .....antecedent .....
. . . | . | 1. i : Id
. . . | . | 2. k : Knd
. . . | . | 3. l : IdLnk
. . . | . | 4. ds : x:Id fp-> Type
. . . | . | 5. da : a:Knd fp-> Type
. . . | . | 6. f :
. . . | . | |  (tg:Id
. . . | . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . | |   (da(rcv(l; tg))?Void List)) List
. . . | . | 7. source(l) = i
. . . | . | 8. w : World
. . . | . | 9. p : FairFifo
```

```
. . . | . | 10. FairFifo
. . . | . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . | |                      <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . | |                   else  fi.da(rcv(l@0; tg))
. . . | . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . | 14. ∀e:E.
. . . | . | |   loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . | 15. ∀e:E.
. . . | . | |   isrcv(e)
. . . | . | |   ⇒
. . . | . | |   lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . | 17. ∀a:Action(i).
. . . | . | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . | 18. ∀x:Id.
. . . | . | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . | |                      2of(ds)(x)
. . . | . | |                   else Top fi
. . . | . | 19. IdDeq ∈ EqDecider(Id)
. . . | . | 20. j : Id
. . . | . | 21. t : ℕ
. . . | . | 22. ¬isnull(a(j;t))
. . . | . | 23. True
. . . | . | 24. j = i
. . . | . | 25. kind(<j,t>) = k
. . . | . | 26. ¬isnull(a(i;t))
. . . | . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . | 28. ∀l@0:IdLnk.
. . . | . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . | |    (<k,l>
. . . | . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . | |   ⇒
. . . | . | |   withlnk(l@0;m(i;t))
. . . | . | |   =
. . . | . | |   if source(l@0) = i→
. . . | . | |      concat(map(λtgf.map(λx.<1of(tgf)
. . . | . | |                              ,x>;2of(tgf)
. . . | . | |                                  ((λx.s(i;t).x)
. . . | . | |                                  ,val(a(i;t))));f))
. . . | . | |   else nil fi
. . . | . | ⊢  eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . | |  (<k,l>
. . . | . | |  ,<kind(a(i;t)),l>) ∨₂ false₂
. . . | . | | by Unfold ‘w-ekind‘ -4 THEN Unfold ‘w-act‘ -4
. . . | . | |    THEN
. . . | . | |    Reduce -4
. . . | . | |    THEN
. . . | . | |    Subst (kind(a(i;t)) = k) 0
. . . | . | |
. . . | . | |\
. . . | . | | .....equality .....
. . . | . | | 1. i : Id
```

```
. . . | . | | 2. k : Knd
. . . | . | | 3. l : IdLnk
. . . | . | | 4. ds : x:Id fp-> Type
. . . | . | | 5. da : a:Knd fp-> Type
. . . | . | | 6. f :
. . . | . | |    (tg:Id
. . . | . | |     ×State(ds)→ma-valtype(da; k)→
. . . | . | |     (da(rcv(l; tg))?Void List)) List
. . . | . | | 7. source(l) = i
. . . | . | | 8. w : World
. . . | . | | 9. p : FairFifo
. . . | . | | 10. FairFifo
. . . | . | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . | |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . | |                         <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . | |                       else  fi.da(rcv(l@0; tg))
. . . | . | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . | | 14. ∀e:E.
. . . | . | |     loc(e) = i
. . . | . | |     ⇒
. . . | . | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . | | 15. ∀e:E.
. . . | . | |     isrcv(e)
. . . | . | |     ⇒
. . . | . | |     lnk(e) = l
. . . | . | |     ⇒
. . . | . | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . | |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . | | 17. ∀a:Action(i).
. . . | . | |     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . | | 18. ∀x:Id.
. . . | . | |     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . | |                       2of(ds)(x)
. . . | . | |                     else Top fi
. . . | . | | 19. IdDeq ∈ EqDecider(Id)
. . . | . | | 20. j : Id
. . . | . | | 21. t : ℕ
. . . | . | | 22. ¬isnull(a(j;t))
. . . | . | | 23. True
. . . | . | | 24. j = i
. . . | . | | 25. kind(a(j;t)) = k
. . . | . | | 26. ¬isnull(a(i;t))
. . . | . | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . | | 28. ∀l@0:IdLnk.
. . . | . | |     (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . | |      (<k,l>
. . . | . | |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . | |     ⇒
. . . | . | |     withlnk(l@0;m(i;t))
. . . | . | |     =
. . . | . | |     if source(l@0) = i→
. . . | . | |        concat(map(λtgf.map(λx.<1of(tgf)
. . . | . | |                                ,x>;2of(tgf)
```

```
. . . | . | |                                    ((λx.s(i;t).x)
. . . | . | |                                    ,val(a(i;t))));f))
. . . | . | |     else nil fi
. . . | . | | ⊢  kind(a(i;t)) = k
. . . | . | |   by Assert (kind(a(j;t)) = kind(a(i;t)))
. . . | . | |      THEN
. . . | . | |      Repeat Analyze
. . . | . | |
. . . | . | .\
. . . | . | . 1. i : Id
. . . | . | . 2. k : Knd
. . . | . | . 3. l : IdLnk
. . . | . | . 4. ds : x:Id fp-> Type
. . . | . | . 5. da : a:Knd fp-> Type
. . . | . | . 6. f :
. . . | . | .    (tg:Id
. . . | . | .     ×State(ds)→ma-valtype(da; k)→
. . . | . | .     (da(rcv(l; tg))?Void List)) List
. . . | . | . 7. source(l) = i
. . . | . | . 8. w : World
. . . | . | . 9. p : FairFifo
. . . | . | . 10. FairFifo
. . . | . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . | .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . | .                        <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . | .                   else  fi.da(rcv(l@0; tg))
. . . | . | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . | . 14. ∀e:E.
. . . | . | .     loc(e) = i
. . . | . | .     ⇒
. . . | . | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . | . 15. ∀e:E.
. . . | . | .     isrcv(e)
. . . | . | .     ⇒
. . . | . | .     lnk(e) = l
. . . | . | .     ⇒
. . . | . | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . | .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . | . 17. ∀a:Action(i).
. . . | . | .     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . | . 18. ∀x:Id.
. . . | . | .     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . | .                       2of(ds)(x)
. . . | . | .                  else Top fi
. . . | . | . 19. IdDeq ∈ EqDecider(Id)
. . . | . | . 20. j : Id
. . . | . | . 21. t : ℕ
. . . | . | . 22. ¬isnull(a(j;t))
. . . | . | . 23. True
. . . | . | . 24. j = i
. . . | . | . 25. kind(a(j;t)) = k
. . . | . | . 26. ¬isnull(a(i;t))
. . . | . | . 27. (λx.s(i;t).x) ∈ State(ds)
```

```
. . . | . | . 28. ∀l@0:IdLnk.
. . . | . | .     (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . | .      (<k,l>
. . . | . | .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . | .     ⇒
. . . | . | .     withlnk(l@0;m(i;t))
. . . | . | .     =

. . . | . | .     if source(l@0) = i→
. . . | . | .        concat(map(λtgf.map(λx.<1of(tgf)
. . . | . | .                                ,x>;2of(tgf)
. . . | . | .                                    ((λx.s(i;t).x)
. . . | . | .                                    ,val(a(i;t))));f))
. . . | . | .     else nil fi
. . . | . | . ⊢  eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . | .    (<k,l>
. . . | . | .    ,<k,l>) ∨₂ false₂
. . . | . | .   by RW0
. . . | . | .      Thm* ∀d:EqDecider(A), i:A. (eqof(d)(i,i)) ∼ true₂
. . . | . | .      {eqof_eq_btrue}
. . . | . | .      0
. . . | . | .      THEN
. . . | . | .      Reduce 0
. . . | . |
. . . | . .\
. . . | . . 1. i : Id
. . . | . . 2. k : Knd
. . . | . . 3. l : IdLnk
. . . | . . 4. ds : x:Id fp-> Type
. . . | . . 5. da : a:Knd fp-> Type
. . . | . . 6. f :
. . . | . . |  (tg:Id
. . . | . . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . |   (da(rcv(l; tg))?Void List)) List
. . . | . . 7. source(l) = i
. . . | . . 8. w : World
. . . | . . 9. p : FairFifo
. . . | . . 10. FairFifo
. . . | . . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . |                        <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . . | . . |                    else  fi.da(rcv(l@0; tg))
. . . | . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . 14. ∀e:E.
. . . | . . |   loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . 15. ∀e:E.
. . . | . . |   isrcv(e)
. . . | . . |   ⇒
. . . | . . |   lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . 17. ∀a:Action(i).
. . . | . . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . 18. ∀x:Id.
. . . | . . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
```

```
. . . | . . |                      2of(ds)(x)
. . . | . . |                    else Top fi
. . . | . . 19. IdDeq ∈ EqDecider(Id)
. . . | . . 20. j : Id
. . . | . . 21. t : ℕ
. . . | . . 22. ¬isnull(a(j;t))
. . . | . . 23. True
. . . | . . 24. j = i
. . . | . . 25. kind(<j,t>) = k
. . . | . . 26. ¬isnull(a(i;t))
. . . | . . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . 28. ∀l@0:IdLnk.
. . . | . . |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . |    (<k,l>
. . . | . . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . |   ⇒
. . . | . . |   withlnk(l@0;m(i;t))
. . . | . . |   =
. . . | . . |   if source(l@0) = i→
. . . | . . |      concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . |                              ,x>;2of(tgf)
. . . | . . |                                  ((λx.s(i;t).x)
. . . | . . |                                  ,val(a(i;t))));f))
. . . | . . |   else nil fi
. . . | . . 29. withlnk(l;m(i;t))
. . . | . . |   =
. . . | . . |   if source(l) = i→
. . . | . . |      concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . |                              ,x>;2of(tgf)
. . . | . . |                                  ((λx.s(i;t).x)
. . . | . . |                                  ,val(a(i;t))));f))
. . . | . . |   else nil fi
. . . | . . ⊢  sends(l;<j,t>)
. . . | . . |  =
. . . | . . |  tagged-messages(l;λz.s(j;t).z;val(a(j;t));f)
. . . | . . | by Unfolds [‘es-eq-Lnk‘;‘w-sends‘] 0
. . . | . . |    THEN
. . . | . . |    Unfolds [‘w-onlnk‘;‘w-loc‘;‘w-time‘] 0
. . . | . . |    THEN
. . . | . . |    Reduce 0
. . . | . . |
. . . | . . 1. i : Id
. . . | . . 2. k : Knd
. . . | . . 3. l : IdLnk
. . . | . . 4. ds : x:Id fp-> Type
. . . | . . 5. da : a:Knd fp-> Type
. . . | . . 6. f :
. . . | . . |  (tg:Id
. . . | . . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . |   (da(rcv(l; tg))?Void List)) List
. . . | . . 7. source(l) = i
. . . | . . 8. w : World
. . . | . . 9. p : FairFifo
. . . | . . 10. FairFifo
. . . | . . 11. ∀l@0:IdLnk, tg:Id.
```

```
. . . | . . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . |                       <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . |                    else  fi.da(rcv(l@0; tg))
. . . | . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . 14. ∀e:E.
. . . | . . |   loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . 15. ∀e:E.
. . . | . . |   isrcv(e)
. . . | . . |   ⇒
. . . | . . |   lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . 17. ∀a:Action(i).
. . . | . . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . 18. ∀x:Id.
. . . | . . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . |                      2of(ds)(x)
. . . | . . |                   else Top fi
. . . | . . 19. IdDeq ∈ EqDecider(Id)
. . . | . . 20. j : Id
. . . | . . 21. t : ℕ
. . . | . . 22. ¬isnull(a(j;t))
. . . | . . 23. True
. . . | . . 24. j = i
. . . | . . 25. kind(<j,t>) = k
. . . | . . 26. ¬isnull(a(i;t))
. . . | . . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . 28. ∀l@0:IdLnk.
. . . | . . |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . |    (<k,l>
. . . | . . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . |   ⇒
. . . | . . |   withlnk(l@0;m(i;t))
. . . | . . |   =
. . . | . . |   if source(l@0) = i→
. . . | . . |      concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . |                              ,x>;2of(tgf)
. . . | . . |                                  ((λx.s(i;t).x)
. . . | . . |                                  ,val(a(i;t))));f))
. . . | . . |   else nil fi
. . . | . . 29. withlnk(l;m(i;t))
. . . | . . |   =
. . . | . . |   if source(l) = i→
. . . | . . |      concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . |                              ,x>;2of(tgf)
. . . | . . |                                  ((λx.s(i;t).x)
. . . | . . |                                  ,val(a(i;t))));f))
. . . | . . |   else nil fi
. . . | . . ⊢  filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . |  =
. . . | . . |  tagged-messages(l;λz.s(j;t).z;val(a(j;t));f)
. . . | . . | by SplitOnHypITE -1
. . . | . . |
. . . | . . |\
```

```
. . . | . . | .....truecase .....
. . . | . . | 1. i : Id
. . . | . . | 2. k : Knd
. . . | . . | 3. l : IdLnk
. . . | . . | 4. ds : x:Id fp-> Type
. . . | . . | 5. da : a:Knd fp-> Type
. . . | . . | 6. f :
. . . | . . | | (tg:Id
. . . | . . | |  ×State(ds)→ma-valtype(da; k)→
. . . | . . | |  (da(rcv(l; tg))?Void List)) List
. . . | . . | 7. source(l) = i
. . . | . . | 8. w : World
. . . | . . | 9. p : FairFifo
. . . | . . | 10. FairFifo
. . . | . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | |                    <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . | |                 else  fi.da(rcv(l@0; tg))
. . . | . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | 14. ∀e:E.
. . . | . . | | loc(e) = i
. . . | . . | | ⇒
. . . | . . | | (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | 15. ∀e:E.
. . . | . . | | isrcv(e)
. . . | . . | | ⇒
. . . | . . | | lnk(e) = l
. . . | . . | | ⇒
. . . | . . | | (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | 17. ∀a:Action(i).
. . . | . . | | ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | 18. ∀x:Id.
. . . | . . | | vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . | |                  2of(ds)(x)
. . . | . . | |               else Top fi
. . . | . . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | 20. j : Id
. . . | . . | 21. t : ℕ
. . . | . . | 22. ¬isnull(a(j;t))
. . . | . . | 23. True
. . . | . . | 24. j = i
. . . | . . | 25. kind(<j,t>) = k
. . . | . . | 26. ¬isnull(a(i;t))
. . . | . . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | 28. ∀l@0:IdLnk.
. . . | . . | | (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | |  (<k,l>
. . . | . . | |  ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | ⇒
. . . | . . | | withlnk(l@0;m(i;t))
. . . | . . | | =
. . . | . . | | if source(l@0) = i→
```

```
. . . | . . | |      concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | |                              ,x>;2of(tgf)
. . . | . . | |                                  ((λx.s(i;t).x)
. . . | . . | |                                  ,val(a(i;t))));f))
. . . | . . | |   else nil fi
. . . | . . | 29. withlnk(l;m(i;t))
. . . | . . | |   =
. . . | . . | |   concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | |                           ,x>;2of(tgf)
. . . | . . | |                               ((λx.s(i;t).x)
. . . | . . | |                               ,val(a(i;t))));f))
. . . | . . | 30. source(l) = i
. . . | . . | ⊢  filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . | |  =
. . . | . . | |  tagged-messages(l;λz.s(j;t).z;val(a(j;t));f)
. . . | . . | | by Unfold ‘tagged-messages‘ 0
. . . | . . | |
. . . | . . | 1. i : Id
. . . | . . | 2. k : Knd
. . . | . . | 3. l : IdLnk
. . . | . . | 4. ds : x:Id fp-> Type
. . . | . . | 5. da : a:Knd fp-> Type
. . . | . . | 6. f :
. . . | . . | |  (tg:Id
. . . | . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | 7. source(l) = i
. . . | . . | 8. w : World
. . . | . . | 9. p : FairFifo
. . . | . . | 10. FairFifo
. . . | . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | |                      <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . | |                   else  fi.da(rcv(l@0; tg))
. . . | . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | 14. ∀e:E.
. . . | . . | |   loc(e) = i
. . . | . . | |   ⇒
. . . | . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | 15. ∀e:E.
. . . | . . | |   isrcv(e)
. . . | . . | |   ⇒
. . . | . . | |   lnk(e) = l
. . . | . . | |   ⇒
. . . | . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | 17. ∀a:Action(i).
. . . | . . | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | 18. ∀x:Id.
. . . | . . | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . | |                      2of(ds)(x)
. . . | . . | |                   else Top fi
. . . | . . | 19. IdDeq ∈ EqDecider(Id)
```

```
. . . | . . | 20. j : Id
. . . | . . | 21. t : ℕ
. . . | . . | 22. ¬isnull(a(j;t))
. . . | . . | 23. True
. . . | . . | 24. j = i
. . . | . . | 25. kind(<j,t>) = k
. . . | . . | 26. ¬isnull(a(i;t))
. . . | . . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | 28. ∀l@0:IdLnk.
. . . | . . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | |    (<k,l>
. . . | . . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | |   ⇒
. . . | . . | |   withlnk(l@0;m(i;t))
. . . | . . | |   =
. . . | . . | |   if source(l@0) = i→
. . . | . . | |      concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | |                              ,x>;2of(tgf)
. . . | . . | |                                  ((λx.s(i;t).x)
. . . | . . | |                                  ,val(a(i;t))));f))
. . . | . . | |   else nil fi
. . . | . . | 29. withlnk(l;m(i;t))
. . . | . . | |   =
. . . | . . | |   concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | |                         ,x>;2of(tgf)
. . . | . . | |                             ((λx.s(i;t).x)
. . . | . . | |                             ,val(a(i;t))));f))
. . . | . . | 30. source(l) = i
. . . | . . | ⊢  filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . | |  =
. . . | . . | |  map(λx.<l,x>;tagged-list-messages(λz.
. . . | . . | |  s(j;t).z;val(a(j;t));f))

. . . | . . | | by Unfold ‘w-withlnk‘ -3 THEN Unfold ‘mapfilter‘ -3
. . . | . . | |

. . . | . . | 1. i : Id
. . . | . . | 2. k : Knd
. . . | . . | 3. l : IdLnk
. . . | . . | 4. ds : x:Id fp-> Type
. . . | . . | 5. da : a:Knd fp-> Type
. . . | . . | 6. f :
. . . | . . | |  (tg:Id
. . . | . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | 7. source(l) = i
. . . | . . | 8. w : World
. . . | . . | 9. p : FairFifo
. . . | . . | 10. FairFifo
. . . | . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | |                        <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . | |                    else  fi.da(rcv(l@0; tg))
. . . | . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | 14. ∀e:E.
. . . | . . | |   loc(e) = i
```

```
. . . | . . | |   ⇒
. . . | . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | 15. ∀e:E.
. . . | . . | |   isrcv(e)
. . . | . . | |   ⇒
. . . | . . | |   lnk(e) = l
. . . | . . | |   ⇒
. . . | . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | 17. ∀a:Action(i).
. . . | . . | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | 18. ∀x:Id.
. . . | . . | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . | |                     2of(ds)(x)
. . . | . . | |                  else Top fi
. . . | . . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | 20. j : Id
. . . | . . | 21. t : ℕ
. . . | . . | 22. ¬isnull(a(j;t))
. . . | . . | 23. True
. . . | . . | 24. j = i
. . . | . . | 25. kind(<j,t>) = k
. . . | . . | 26. ¬isnull(a(i;t))
. . . | . . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | 28. ∀l@0:IdLnk.
. . . | . . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | |    (<k,l>
. . . | . . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | |   ⇒
. . . | . . | |   map(λms.2of(ms);filter(λms.mlnk(ms) = l@0;m(i;t)))
. . . | . . | |   =
. . . | . . | |   if source(l@0) = i→
. . . | . . | |      concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | |                               ,x>;2of(tgf)
. . . | . . | |                                   ((λx.s(i;t).x)
. . . | . . | |                                   ,val(a(i;t))));f))
. . . | . . | |   else nil fi
. . . | . . | 29. withlnk(l;m(i;t))
. . . | . . | |   =
. . . | . . | |   concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | |                           ,x>;2of(tgf)
. . . | . . | |                               ((λx.s(i;t).x)
. . . | . . | |                               ,val(a(i;t))));f))
. . . | . . | 30. source(l) = i
. . . | . . | ⊢  filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . | |  =
. . . | . . | |  map(λx.<l,x>;tagged-list-messages(λz.
. . . | . . | |  s(j;t).z;val(a(j;t));f))
. . . | . . | | by Fold ‘tagged-list-messages‘ -3
. . . | . . | |
. . . | . . | 1. i : Id
. . . | . . | 2. k : Knd
. . . | . . | 3. l : IdLnk
. . . | . . | 4. ds : x:Id fp-> Type
```

```
. . . | . . | 5. da : a:Knd fp-> Type
. . . | . . | 6. f :
. . . | . . | | (tg:Id
. . . | . . | |  ×State(ds)→ma-valtype(da; k)→
. . . | . . | |  (da(rcv(l; tg))?Void List)) List
. . . | . . | 7. source(l) = i
. . . | . . | 8. w : World
. . . | . . | 9. p : FairFifo
. . . | . . | 10. FairFifo
. . . | . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | |  (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | |                    <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . . | . . | |                  else  fi.da(rcv(l@0; tg))
. . . | . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | 14. ∀e:E.
. . . | . . | |  loc(e) = i
. . . | . . | |  ⇒
. . . | . . | |  (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | 15. ∀e:E.
. . . | . . | |  isrcv(e)
. . . | . . | |  ⇒

. . . | . . | |  lnk(e) = l
. . . | . . | |  ⇒
. . . | . . | |  (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | |  ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | 17. ∀a:Action(i).
. . . | . . | |  ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | 18. ∀x:Id.
. . . | . . | |  vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . | |                    2of(ds)(x)
. . . | . . | |                  else Top fi
. . . | . . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | 20. j : Id
. . . | . . | 21. t : ℕ
. . . | . . | 22. ¬isnull(a(j;t))
. . . | . . | 23. True
. . . | . . | 24. j = i
. . . | . . | 25. kind(<j,t>) = k
. . . | . . | 26. ¬isnull(a(i;t))
. . . | . . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | 28. ∀l@0:IdLnk.
. . . | . . | |  (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | |   (<k,l>
. . . | . . | |   ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | |  ⇒
. . . | . . | |  map(λms.2of(ms);filter(λms.mlnk(ms) = l@0;m(i;t)))
. . . | . . | |  =
. . . | . . | |  if source(l@0) = i→
. . . | . . | |     tagged-list-messages(λx.s(i;t).x;val(a(i;t));f)
. . . | . . | |  else nil fi
. . . | . . | 29. withlnk(l;m(i;t))
. . . | . . | |  =
. . . | . . | |  concat(map(λtgf.map(λx.<1of(tgf)
```

```
. . . | . . | |                          ,x>;2of(tgf)
. . . | . . | |                              ((λx.s(i;t).x)
. . . | . . | |                              ,val(a(i;t))));f))
. . . | . . | 30. source(l) = i
. . . | . . | ⊢  filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . | |  =
. . . | . . | |  map(λx.<l,x>;tagged-list-messages(λz.
. . . | . . | |  s(j;t).z;val(a(j;t));f))
. . . | . . | | by Subst
. . . | . . | |    (tagged-list-messages(λz.s(j;t).z;val(a(j;t));f)
. . . | . . | |     =
. . . | . . | |     tagged-list-messages(λx.s(i;t).x;val(a(i;t));f))
. . . | . . | |    0
. . . | . . | |
. . . | . . | |\
. . . | . . | | .....equality .....
. . . | . . | | 1. i : Id
. . . | . . | | 2. k : Knd
. . . | . . | | 3. l : IdLnk
. . . | . . | | 4. ds : x:Id fp-> Type
. . . | . . | | 5. da : a:Knd fp-> Type
. . . | . . | | 6. f :
. . . | . . | | |  (tg:Id
. . . | . . | | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | 7. source(l) = i
. . . | . . | | 8. w : World
. . . | . . | | 9. p : FairFifo
. . . | . . | | 10. FairFifo
. . . | . . | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | |                        <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . | | |                    else  fi.da(rcv(l@0; tg))
. . . | . . | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | 14. ∀e:E.
. . . | . . | | |   loc(e) = i
. . . | . . | | |   ⇒
. . . | . . | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | 15. ∀e:E.
. . . | . . | | |   isrcv(e)
. . . | . . | | |   ⇒
. . . | . . | | |   lnk(e) = l
. . . | . . | | |   ⇒
. . . | . . | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | 17. ∀a:Action(i).
. . . | . . | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | 18. ∀x:Id.
. . . | . . | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . | | |                       2of(ds)(x)
. . . | . . | | |                   else Top fi
. . . | . . | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | 20. j : Id
```

```
. . . | . . | | 21. t : ℕ
. . . | . . | | 22. ¬isnull(a(j;t))
. . . | . . | | 23. True
. . . | . . | | 24. j = i
. . . | . . | | 25. kind(<j,t>) = k
. . . | . . | | 26. ¬isnull(a(i;t))
. . . | . . | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | 28. ∀l@0:IdLnk.
. . . | . . | | |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | |    (<k,l>
. . . | . . | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | |   ⇒
. . . | . . | | |   map(λms.2of(ms);filter(λms.mlnk(ms) = l@0;m(i;t)))
. . . | . . | | |   =
. . . | . . | | |   if source(l@0) = i→
. . . | . . | | |      tagged-list-messages(λx.s(i;t).x;val(a(i;t));f)
. . . | . . | | |   else nil fi
. . . | . . | | 29. withlnk(l;m(i;t))
. . . | . . | | |   =
. . . | . . | | |   concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | | |                           ,x>;2of(tgf)
. . . | . . | | |                               ((λx.s(i;t).x)
. . . | . . | | |                               ,val(a(i;t))));f))
. . . | . . | | 30. source(l) = i
. . . | . . | | ⊢  tagged-list-messages(λz.s(j;t).z;val(a(j;t));f)
. . . | . . | | |  =
. . . | . . | | |  tagged-list-messages(λx.s(i;t).x;val(a(i;t));f)
. . . | . . | | | by Analyze THEN Try (Complete Auto)
. . . | . . | | |
. . . | . . | | |\
. . . | . . | | | .....subterm .....
. . . | . . | | | 1. i : Id
. . . | . . | | | 2. k : Knd
. . . | . . | | | 3. l : IdLnk
. . . | . . | | | 4. ds : x:Id fp-> Type
. . . | . . | | | 5. da : a:Knd fp-> Type
. . . | . . | | | 6. f :
. . . | . . | | | |  (tg:Id
. . . | . . | | | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | 7. source(l) = i
. . . | . . | | | 8. w : World
. . . | . . | | | 9. p : FairFifo
. . . | . . | | | 10. FairFifo
. . . | . . | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | | |                      <ds
. . . | . . | | | |                      ,da
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,<[<k,l>],λx.f>
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,.>
```

```
. . . | . . | | | |                    else  fi.da(rcv(l@0; tg))
. . . | . . | | | 12. (eqof(IdDeq)(i,i)) ~ true₂
. . . | . . | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | 14. ∀e:E.
. . . | . . | | | |   loc(e) = i
. . . | . . | | | |   ⇒
. . . | . . | | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | 15. ∀e:E.
. . . | . . | | | |   isrcv(e)
. . . | . . | | | |   ⇒
. . . | . . | | | |   lnk(e) = l
. . . | . . | | | |   ⇒
. . . | . . | | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | 17. ∀a:Action(i).
. . . | . . | | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | 18. ∀x:Id.
. . . | . . | | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . | | | |                      2of(ds)(x)
. . . | . . | | | |                   else Top fi
. . . | . . | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | 20. j : Id
. . . | . . | | | 21. t : ℕ
. . . | . . | | | 22. ¬isnull(a(j;t))
. . . | . . | | | 23. True
. . . | . . | | | 24. j = i
. . . | . . | | | 25. kind(<j,t>) = k
. . . | . . | | | 26. ¬isnull(a(i;t))
. . . | . . | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | 28. ∀l@0:IdLnk.
. . . | . . | | | |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | | |    (<k,l>
. . . | . . | | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | |   ⇒
. . . | . . | | | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | |   =
. . . | . . | | | |   if source(l@0) = i→
. . . | . . | | | |      tagged-list-messages(λx.
. . . | . . | | | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | |   else nil fi
. . . | . . | | | 29. withlnk(l;m(i;t))
. . . | . . | | | |   =
. . . | . . | | | |   concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | | | |                          ,x>;2of(tgf)
. . . | . . | | | |                              ((λx.s(i;t).x)
. . . | . . | | | |                              ,val(a(i;t))));f))
. . . | . . | | | 30. source(l) = i
. . . | . . | | | ⊢  (λz.s(j;t).z) = (λx.s(i;t).x)
. . . | . . | | | | by Unfold ‘ma-state‘ 0 THEN Analyze
. . . | . . | | | |
. . . | . . | | | |\
. . . | . . | | | | .....subterm .....
. . . | . . | | | | 1. i : Id
```

```
. . . | . . | | | | 2. k : Knd
. . . | . . | | | | 3. l : IdLnk

. . . | . . | | | | 4. ds : x:Id fp-> Type
. . . | . . | | | | 5. da : a:Knd fp-> Type
. . . | . . | | | | 6. f :
. . . | . . | | | | |  (tg:Id
. . . | . . | | | | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | | 7. source(l) = i
. . . | . . | | | | 8. w : World
. . . | . . | | | | 9. p : FairFifo
. . . | . . | | | | 10. FairFifo
. . . | . . | | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | | |   (w.M
. . . | . . | | | | |    (l@0
. . . | . . | | | | |    ,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | | | |                <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . | | | | |             else  fi.da(rcv(l@0; tg))
. . . | . . | | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | | 14. ∀e:E.
. . . | . . | | | | |   loc(e) = i
. . . | . . | | | | |   ⇒
. . . | . . | | | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | 15. ∀e:E.
. . . | . . | | | | |   isrcv(e)
. . . | . . | | | | |   ⇒
. . . | . . | | | | |   lnk(e) = l
. . . | . . | | | | |   ⇒
. . . | . . | | | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | | 17. ∀a:Action(i).
. . . | . . | | | | |   ¬isnull(a)
. . . | . . | | | | |   ⇒
. . . | . . | | | | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | | 18. ∀x:Id.
. . . | . . | | | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | | |                                          ds))→
. . . | . . | | | | |                     2of(ds)(x)
. . . | . . | | | | |                   else Top fi
. . . | . . | | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | | 20. j : Id
. . . | . . | | | | 21. t : ℕ
. . . | . . | | | | 22. ¬isnull(a(j;t))
. . . | . . | | | | 23. True
. . . | . . | | | | 24. j = i
. . . | . . | | | | 25. kind(<j,t>) = k
. . . | . . | | | | 26. ¬isnull(a(i;t))
. . . | . . | | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | | 28. ∀l@0:IdLnk.
. . . | . . | | | | |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | | | |    (<k,l>
. . . | . . | | | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | | |   ⇒
```

```
. . . | . . | | | | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | | |   =
. . . | . . | | | | |   if source(l@0) = i→
. . . | . . | | | | |      tagged-list-messages(λx.
. . . | . . | | | | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | | |   else nil fi
. . . | . . | | | | 29. withlnk(l;m(i;t))
. . . | . . | | | | |   =
. . . | . . | | | | |   concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | | | | |                          ,x>;2of(tgf)
. . . | . . | | | | |                              ((λx.s(i;t).x)
. . . | . . | | | | |                              ,val(a(i;t))));f))
. . . | . . | | | | 30. source(l) = i
. . . | . . | | | | 31. z : Id
. . . | . . | | | | ⊢  s(j;t).z = s(i;t).z  by StrongHypSubst -8 0
. . . | . . | | | | |
. . . | . . | | | | |\
. . . | . . | | | | 1. i : Id
. . . | . . | | | | 2. k : Knd
. . . | . . | | | | 3. l : IdLnk
. . . | . . | | | | 4. ds : x:Id fp-> Type
. . . | . . | | | | 5. da : a:Knd fp-> Type
. . . | . . | | | | 6. f :
. . . | . . | | | |    (tg:Id
. . . | . . | | | |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | |     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | | 7. source(l) = i
. . . | . . | | | | 8. w : World
. . . | . . | | | | 9. p : FairFifo
. . . | . . | | | | 10. FairFifo
. . . | . . | | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | |                          (source(l@0)
. . . | . . | | | |                          ,i)→
. . . | . . | | | |                          <ds
. . . | . . | | | |                          ,da
. . . | . . | | | |                          ,
. . . | . . | | | |                          ,
. . . | . . | | | |                          ,
. . . | . . | | | |                          ,<[<k,l>],λx.f>
. . . | . . | | | |                          ,
. . . | . . | | | |                          ,
. . . | . . | | | |                          ,·>
. . . | . . | | | |                      else  fi.da(rcv(l@0; tg))
. . . | . . | | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | | 14. ∀e:E.
. . . | . . | | | |     loc(e) = i
. . . | . . | | | |     ⇒
. . . | . . | | | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | 15. ∀e:E.
. . . | . . | | | |     isrcv(e)
. . . | . . | | | |     ⇒
. . . | . . | | | |     lnk(e) = l
```

```
. . . | . . | | | | |     ⇒
. . . | . . | | | | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | | |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | | | 17. ∀a:Action(i).
. . . | . . | | | | |     ¬isnull(a)
. . . | . . | | | | |     ⇒
. . . | . . | | | | |     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | | | 18. ∀x:Id.
. . . | . . | | | | |     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | | |                                          ds))→
. . . | . . | | | | |                        2of(ds)(x)
. . . | . . | | | | |                      else Top fi
. . . | . . | | | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | | | 20. j : Id
. . . | . . | | | | | 21. t : ℕ
. . . | . . | | | | | 22. ¬isnull(a(j;t))
. . . | . . | | | | | 23. True
. . . | . . | | | | | 24. j = i
. . . | . . | | | | | 25. kind(<j,t>) = k
. . . | . . | | | | | 26. ¬isnull(a(i;t))
. . . | . . | | | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | | | 28. ∀l@0:IdLnk.
. . . | . . | | | | |     (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | | |      (<k,l>
. . . | . . | | | | |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | | |     ⇒
. . . | . . | | | | |     map(λms.2of(ms);filter(λms.
. . . | . . | | | | |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | | |     =
. . . | . . | | | | |     if source(l@0) = i→
. . . | . . | | | | |        tagged-list-messages(λx.
. . . | . . | | | | |        s(i;t).x;val(a(i;t));f)
. . . | . . | | | | |     else nil fi
. . . | . . | | | | | 29. withlnk(l;m(i;t))
. . . | . . | | | | |     =
. . . | . . | | | | |     concat(map(λtgf.map(λx.
. . . | . . | | | | |     <1of(tgf),x>;2of(tgf)
. . . | . . | | | | |                  ((λx.s(i;t).x)
. . . | . . | | | | |                  ,val(a(i;t))));f))
. . . | . . | | | | | 30. source(l) = i
. . . | . . | | | | | 31. z : Id
. . . | . . | | | | | ⊢ s(i;t).z = s(i;t).z
. . . | . . | | | | |   by Fold ‘member‘ 0 THEN DoSubsume
. . . | . . | | | | |      THEN
. . . | . . | | | | |      Try BackThruSomeHyp
. . . | . . | | | | |      THEN
. . . | . . | | | | |      D_ES_Subtype
. . . | . . | | | | |
. . . | . . | | | | .\
. . . | . . | | | | . 1. i : Id
. . . | . . | | | | . 2. k : Knd
. . . | . . | | | | . 3. l : IdLnk
. . . | . . | | | | . 4. ds : x:Id fp-> Type
. . . | . . | | | | . 5. da : a:Knd fp-> Type
```

```
. . . | . . | | | | . 6. f :
. . . | . . | | | | . |  (tg:Id
. . . | . . | | | | . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | | . |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | | . 7. source(l) = i
. . . | . . | | | | . 8. w : World
. . . | . . | | | | . 9. p : FairFifo
. . . | . . | | | | . 10. FairFifo
. . . | . . | | | | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | | . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | | . |                       (source(l@0)
. . . | . . | | | | . |                       ,i)→
. . . | . . | | | | . |                       <ds
. . . | . . | | | | . |                       ,da
. . . | . . | | | | . |                       ,
. . . | . . | | | | . |                       ,
. . . | . . | | | | . |                       ,
. . . | . . | | | | . |                       ,<[<k,l>],λx.f>
. . . | . . | | | | . |                       ,
. . . | . . | | | | . |                       ,
. . . | . . | | | | . |                       ,·>
. . . | . . | | | | . |                     else  fi.da(rcv(l@0; tg))
. . . | . . | | | | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | | . 14. ∀e:E.
. . . | . . | | | | . |   loc(e) = i
. . . | . . | | | | . |   ⇒
. . . | . . | | | | . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | . 15. ∀e:E.
. . . | . . | | | | . |   isrcv(e)
. . . | . . | | | | . |   ⇒
. . . | . . | | | | . |   lnk(e) = l
. . . | . . | | | | . |   ⇒
. . . | . . | | | | . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | | . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | | . 17. ∀a:Action(i).
. . . | . . | | | | . |   ¬isnull(a)
. . . | . . | | | | . |   ⇒
. . . | . . | | | | . |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | | . 18. ∀x:Id.
. . . | . . | | | | . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | | . |                                          ds))→
. . . | . . | | | | . |                     2of(ds)(x)
. . . | . . | | | | . |                   else Top fi
. . . | . . | | | | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | | . 20. j : Id
. . . | . . | | | | . 21. t : ℕ
. . . | . . | | | | . 22. ¬isnull(a(j;t))
. . . | . . | | | | . 23. True
. . . | . . | | | | . 24. j = i
. . . | . . | | | | . 25. kind(<j,t>) = k
. . . | . . | | | | . 26. ¬isnull(a(i;t))
. . . | . . | | | | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | | . 28. ∀l@0:IdLnk.
```

```
. . . | . . | | | | . |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | | . |    (<k,l>
. . . | . . | | | | . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | | . |   ⇒
. . . | . . | | | | . |   map(λms.2of(ms);filter(λms.
. . . | . . | | | | . |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | | . |   =
. . . | . . | | | | . |   if source(l@0) = i→
. . . | . . | | | | . |      tagged-list-messages(λx.
. . . | . . | | | | . |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | | . |   else nil fi
. . . | . . | | | | . 29. withlnk(l;m(i;t))
. . . | . . | | | | . |   =
. . . | . . | | | | . |   concat(map(λtgf.map(λx.
. . . | . . | | | | . |   <1of(tgf),x>;2of(tgf)
. . . | . . | | | | . |                ((λx.s(i;t).x)
. . . | . . | | | | . |                ,val(a(i;t))));f))
. . . | . . | | | | . 30. source(l) = i
. . . | . . | | | | . 31. z : Id
. . . | . . | | | | . 32. z1 : Id
. . . | . . | | | | . 33. z1 = i
. . . | . . | | | | . ⊢  (s(z1;t).z = s(i;t).z) ∈ Prop  by Auto
. . . | . . | | | | . |
. . . | . . | | | | . |\
. . . | . . | | | | . | .....subterm .....
. . . | . . | | | | . | 1. i : Id
. . . | . . | | | | . | 2. k : Knd
. . . | . . | | | | . | 3. l : IdLnk
. . . | . . | | | | . | 4. ds : x:Id fp-> Type
. . . | . . | | | | . | 5. da : a:Knd fp-> Type
. . . | . . | | | | . | 6. f :
. . . | . . | | | | . | |  (tg:Id
. . . | . . | | | | . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | | . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | | . | 7. source(l) = i
. . . | . . | | | | . | 8. w : World
. . . | . . | | | | . | 9. p : FairFifo
. . . | . . | | | | . | 10. FairFifo
. . . | . . | | | | . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | | . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | | . | |                       (source(l@0)
. . . | . . | | | | . | |                       ,i)→
. . . | . . | | | | . | |                       <ds
. . . | . . | | | | . | |                       ,da
. . . | . . | | | | . | |                       ,
. . . | . . | | | | . | |                       ,
. . . | . . | | | | . | |                       ,
. . . | . . | | | | . | |                       ,<[<k,l>],λx.f>
. . . | . . | | | | . | |                       ,
. . . | . . | | | | . | |                       ,
. . . | . . | | | | . | |                       ,•>
. . . | . . | | | | . | |                   else  fi.da(rcv(l@0; tg))
. . . | . . | | | | . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | | . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | | . | 14. ∀e:E.
```

```
. . . | . . | | | | . | |   loc(e) = i
. . . | . . | | | | . | |   ⇒
. . . | . . | | | | . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | . | 15. ∀e:E.
. . . | . . | | | | . | |   isrcv(e)
. . . | . . | | | | . | |   ⇒
. . . | . . | | | | . | |   lnk(e) = l
. . . | . . | | | | . | |   ⇒
. . . | . . | | | | . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | | . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | | . | 17. ∀a:Action(i).
. . . | . . | | | | . | |   ¬isnull(a)
. . . | . . | | | | . | |   ⇒
. . . | . . | | | | . | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | | . | 18. ∀x:Id.
. . . | . . | | | | . | |   vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | | . | |                                     ds))→
. . . | . . | | | | . | |                    2of(ds)(x)
. . . | . . | | | | . | |                  else Top fi
. . . | . . | | | | . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | | . | 20. j : Id
. . . | . . | | | | . | 21. t : ℕ
. . . | . . | | | | . | 22. ¬isnull(a(j;t))
. . . | . . | | | | . | 23. True
. . . | . . | | | | . | 24. j = i
. . . | . . | | | | . | 25. kind(<j,t>) = k
. . . | . . | | | | . | 26. ¬isnull(a(i;t))
. . . | . . | | | | . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | | . | 28. ∀l@0:IdLnk.
. . . | . . | | | | . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | | . | |    (<k,l>
. . . | . . | | | | . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | | . | |   ⇒
. . . | . . | | | | . | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | | . | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | | . | |   =
. . . | . . | | | | . | |   if source(l@0) = i→
. . . | . . | | | | . | |      tagged-list-messages(λx.
. . . | . . | | | | . | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | | . | |   else nil fi
. . . | . . | | | | . | 29. withlnk(l;m(i;t))
. . . | . . | | | | . | |   =
. . . | . . | | | | . | |   concat(map(λtgf.map(λx.
. . . | . . | | | | . | |   <1of(tgf),x>;2of(tgf)
. . . | . . | | | | . | |                ((λx.s(i;t).x)
. . . | . . | | | | . | |                ,val(a(i;t))));f))
. . . | . . | | | | . | 30. source(l) = i
. . . | . . | | | | . | 31. z : Id
. . . | . . | | | | . | 32. z1 : Id
. . . | . . | | | | . | 33. z1 = i
. . . | . . | | | | . | ⊢  s(z1;t).z ∈ ds(z)?Top  by DoSubsume
. . . | . . | | | | . | |
. . . | . . | | | | . | .....subtype_rel .....
. . . | . . | | | | . | 1. i : Id
```

```
. . . | . . | | | | . | 2. k : Knd
. . . | . . | | | | . | 3. l : IdLnk
. . . | . . | | | | . | 4. ds : x:Id fp-> Type
. . . | . . | | | | . | 5. da : a:Knd fp-> Type
. . . | . . | | | | . | 6. f :
. . . | . . | | | | . |    (tg:Id
. . . | . . | | | | . |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | | . |     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | | . | 7. source(l) = i
. . . | . . | | | | . | 8. w : World
. . . | . . | | | | . | 9. p : FairFifo
. . . | . . | | | | . | 10. FairFifo
. . . | . . | | | | . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | | . |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | | . |                          (source(l@0)
. . . | . . | | | | . |                          ,i)→
. . . | . . | | | | . |                          <ds
. . . | . . | | | | . |                          ,da
. . . | . . | | | | . |                          ,
. . . | . . | | | | . |                          ,
. . . | . . | | | | . |                          ,
. . . | . . | | | | . |                          ,<[<k,l>],λx.f>
. . . | . . | | | | . |                          ,
. . . | . . | | | | . |                          ,
. . . | . . | | | | . |                          ,•>
. . . | . . | | | | . |                       else  fi.da(rcv(l@0; tg))
. . . | . . | | | | . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | | . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | | . | 14. ∀e:E.
. . . | . . | | | | . |     loc(e) = i
. . . | . . | | | | . |     ⇒
. . . | . . | | | | . |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | . | 15. ∀e:E.
. . . | . . | | | | . |     isrcv(e)
. . . | . . | | | | . |     ⇒
. . . | . . | | | | . |     lnk(e) = l
. . . | . . | | | | . |     ⇒
. . . | . . | | | | . |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | | . |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | | . | 17. ∀a:Action(i).
. . . | . . | | | | . |     ¬isnull(a)
. . . | . . | | | | . |     ⇒
. . . | . . | | | | . |     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | | . | 18. ∀x:Id.
. . . | . . | | | | . |     vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | | . |                                          ds))→
. . . | . . | | | | . |                          2of(ds)(x)
. . . | . . | | | | . |                       else Top fi
. . . | . . | | | | . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | | . | 20. j : Id
. . . | . . | | | | . | 21. t : ℕ
. . . | . . | | | | . | 22. ¬isnull(a(j;t))
. . . | . . | | | | . | 23. True
. . . | . . | | | | . | 24. j = i
```

```
. . . | . . | | | | . | 25. kind(<j,t>) = k
. . . | . . | | | | . | 26. ¬isnull(a(i;t))
. . . | . . | | | | . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | | . | 28. ∀l@0:IdLnk.
. . . | . . | | | | . |     (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | | . |      (<k,l>
. . . | . . | | | | . |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | | . |     ⇒
. . . | . . | | | | . |     map(λms.2of(ms);filter(λms.
. . . | . . | | | | . |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | | . |     =
. . . | . . | | | | . |     if source(l@0) = i→
. . . | . . | | | | . |        tagged-list-messages(λx.
. . . | . . | | | | . |        s(i;t).x;val(a(i;t));f)
. . . | . . | | | | . |     else nil fi
. . . | . . | | | | . | 29. withlnk(l;m(i;t))
. . . | . . | | | | . |     =
. . . | . . | | | | . |     concat(map(λtgf.map(λx.
. . . | . . | | | | . |     <1of(tgf),x>;2of(tgf)
. . . | . . | | | | . |                  ((λx.s(i;t).x)
. . . | . . | | | | . |                  ,val(a(i;t))));f))
. . . | . . | | | | . | 30. source(l) = i
. . . | . . | | | | . | 31. z : Id
. . . | . . | | | | . | 32. z1 : Id
. . . | . . | | | | . | 33. z1 = i
. . . | . . | | | | . | ⊢  vartype(z1;z) ⊆r ds(z)?Top
. . . | . . | | | | . |   by HypSubst -1 0 THEN BackThruSomeHyp
. . . | . . | | | | . |
. . . | . . | | | | . .\
. . . | . . | | | | . . .....subterm .....
. . . | . . | | | | . . 1. i : Id
. . . | . . | | | | . . 2. k : Knd
. . . | . . | | | | . . 3. l : IdLnk
. . . | . . | | | | . . 4. ds : x:Id fp-> Type
. . . | . . | | | | . . 5. da : a:Knd fp-> Type
. . . | . . | | | | . . 6. f :
. . . | . . | | | | . .    (tg:Id
. . . | . . | | | | . .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | | . .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | | . . 7. source(l) = i
. . . | . . | | | | . . 8. w : World
. . . | . . | | | | . . 9. p : FairFifo
. . . | . . | | | | . . 10. FairFifo
. . . | . . | | | | . . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | | . .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | | . .                      (source(l@0)
. . . | . . | | | | . .                      ,i)→
. . . | . . | | | | . .                      <ds
. . . | . . | | | | . .                      ,da
. . . | . . | | | | . .                      ,
. . . | . . | | | | . .                      ,
. . . | . . | | | | . .                      ,
. . . | . . | | | | . .                      ,<[<k,l>],λx.f>
. . . | . . | | | | . .                      ,
. . . | . . | | | | . .                      ,
```

```
. . . | . . | | | | . .                          ,·>
. . . | . . | | | | . .                        else  fi.da(rcv(l@0; tg))
. . . | . . | | | | . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | | . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | | . . 14. ∀e:E.
. . . | . . | | | | . .     loc(e) = i
. . . | . . | | | | . .     ⇒
. . . | . . | | | | . .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | . . 15. ∀e:E.
. . . | . . | | | | . .     isrcv(e)
. . . | . . | | | | . .     ⇒
. . . | . . | | | | . .     lnk(e) = l
. . . | . . | | | | . .     ⇒
. . . | . . | | | | . .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | | . .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | | . . 17. ∀a:Action(i).
. . . | . . | | | | . .     ¬isnull(a)
. . . | . . | | | | . .     ⇒
. . . | . . | | | | . .     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | | . . 18. ∀x:Id.
. . . | . . | | | | . .     vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | | . .                                        ds))→
. . . | . . | | | | . .                          2of(ds)(x)
. . . | . . | | | | . .                        else Top fi
. . . | . . | | | | . . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | | . . 20. j : Id
. . . | . . | | | | . . 21. t : ℕ
. . . | . . | | | | . . 22. ¬isnull(a(j;t))
. . . | . . | | | | . . 23. True
. . . | . . | | | | . . 24. j = i
. . . | . . | | | | . . 25. kind(<j,t>) = k
. . . | . . | | | | . . 26. ¬isnull(a(i;t))
. . . | . . | | | | . . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | | . . 28. ∀l@0:IdLnk.
. . . | . . | | | | . .     (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | | . .      (<k,l>
. . . | . . | | | | . .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | | . .     ⇒
. . . | . . | | | | . .     map(λms.2of(ms);filter(λms.
. . . | . . | | | | . .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | | . .     =
. . . | . . | | | | . .     if source(l@0) = i→
. . . | . . | | | | . .        tagged-list-messages(λx.
. . . | . . | | | | . .        s(i;t).x;val(a(i;t));f)
. . . | . . | | | | . .     else nil fi
. . . | . . | | | | . . 29. withlnk(l;m(i;t))
. . . | . . | | | | . .     =
. . . | . . | | | | . .     concat(map(λtgf.map(λx.
. . . | . . | | | | . .     <1of(tgf),x>;2of(tgf)
. . . | . . | | | | . .                  ((λx.s(i;t).x)
. . . | . . | | | | . .                  ,val(a(i;t))));f))
. . . | . . | | | | . . 30. source(l) = i
. . . | . . | | | | . . 31. z : Id
. . . | . . | | | | . . 32. z1 : Id
```

```
. . . | . . | | | | . . 33. z1 = i
. . . | . . | | | | . . ⊢  s(i;t).z ∈ ds(z)?Top
. . . | . . | | | | . .   by DoSubsume THEN Try BackThruSomeHyp
. . . | . . | | | | . .      THEN
. . . | . . | | | | . .      D_ES_Subtype
. . . | . . | | | |
. . . | . . | | | .\

. . . | . . | | | . .....eq aux .....
. . . | . . | | | . 1. i : Id
. . . | . . | | | . 2. k : Knd
. . . | . . | | | . 3. l : IdLnk
. . . | . . | | | . 4. ds : x:Id fp-> Type
. . . | . . | | | . 5. da : a:Knd fp-> Type
. . . | . . | | | . 6. f :
. . . | . . | | | .    (tg:Id
. . . | . . | | | .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . 7. source(l) = i
. . . | . . | | | . 8. w : World
. . . | . . | | | . 9. p : FairFifo
. . . | . . | | | . 10. FairFifo
. . . | . . | | | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | .     (w.M
. . . | . . | | | .      (l@0
. . . | . . | | | .      ,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | | .                  <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . . | . . | | | .                else  fi.da(rcv(l@0; tg))
. . . | . . | | | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . 14. ∀e:E.
. . . | . . | | | .     loc(e) = i
. . . | . . | | | .     ⇒
. . . | . . | | | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . 15. ∀e:E.
. . . | . . | | | .     isrcv(e)
. . . | . . | | | .     ⇒
. . . | . . | | | .     lnk(e) = l
. . . | . . | | | .     ⇒
. . . | . . | | | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . 17. ∀a:Action(i).
. . . | . . | | | .     ¬isnull(a)
. . . | . . | | | .     ⇒
. . . | . . | | | .     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . 18. ∀x:Id.
. . . | . . | | | .     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | .                                          ds))→
. . . | . . | | | .                          2of(ds)(x)
. . . | . . | | | .                       else Top fi
. . . | . . | | | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . 20. j : Id
. . . | . . | | | . 21. t : ℕ
. . . | . . | | | . 22. ¬isnull(a(j;t))
. . . | . . | | | . 23. True
```

```
. . . | . . | | | . 24. j = i
. . . | . . | | | . 25. kind(<j,t>) = k
. . . | . . | | | . 26. ¬isnull(a(i;t))
. . . | . . | | | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . 28. ∀l@0:IdLnk.
. . . | . . | | | .     (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | | .      (<k,l>
. . . | . . | | | .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | .     ⇒
. . . | . . | | | .     map(λms.2of(ms);filter(λms.
. . . | . . | | | .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | .     =
. . . | . . | | | .     if source(l@0) = i→
. . . | . . | | | .        tagged-list-messages(λx.
. . . | . . | | | .        s(i;t).x;val(a(i;t));f)
. . . | . . | | | .     else nil fi
. . . | . . | | | . 29. withlnk(l;m(i;t))
. . . | . . | | | .     =
. . . | . . | | | .     concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | | | .                              ,x>;2of(tgf)
. . . | . . | | | .                                  ((λx.s(i;t).x)
. . . | . . | | | .                                  ,val(a(i;t))));f))
. . . | . . | | | . 30. source(l) = i
. . . | . . | | | . ⊢  Id = Id  by Auto
. . . | . . | | |
. . . | . . | | |\
. . . | . . | | | .....subterm .....
. . . | . . | | | 1. i : Id
. . . | . . | | | 2. k : Knd
. . . | . . | | | 3. l : IdLnk
. . . | . . | | | 4. ds : x:Id fp-> Type
. . . | . . | | | 5. da : a:Knd fp-> Type
. . . | . . | | | 6. f :
. . . | . . | | | |  (tg:Id
. . . | . . | | | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | 7. source(l) = i
. . . | . . | | | 8. w : World
. . . | . . | | | 9. p : FairFifo
. . . | . . | | | 10. FairFifo
. . . | . . | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | | |                        <ds
. . . | . . | | | |                        ,da
. . . | . . | | | |                        ,
. . . | . . | | | |                        ,
. . . | . . | | | |                        ,
. . . | . . | | | |                        ,<[<k,l>],λx.f>
. . . | . . | | | |                        ,
. . . | . . | | | |                        ,
. . . | . . | | | |                        ,•>
. . . | . . | | | |                    else  fi.da(rcv(l@0; tg))
. . . | . . | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | 14. ∀e:E.
```

```
. . . | . . | | | |   loc(e) = i
. . . | . . | | | |   ⇒
. . . | . . | | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | 15. ∀e:E.
. . . | . . | | | |   isrcv(e)
. . . | . . | | | |   ⇒
. . . | . . | | | |   lnk(e) = l
. . . | . . | | | |   ⇒
. . . | . . | | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | 17. ∀a:Action(i).
. . . | . . | | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | 18. ∀x:Id.
. . . | . . | | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . | | | |                      2of(ds)(x)
. . . | . . | | | |                  else Top fi
. . . | . . | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | 20. j : Id
. . . | . . | | | 21. t : ℕ
. . . | . . | | | 22. ¬isnull(a(j;t))
. . . | . . | | | 23. True
. . . | . . | | | 24. j = i
. . . | . . | | | 25. kind(<j,t>) = k
. . . | . . | | | 26. ¬isnull(a(i;t))
. . . | . . | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | 28. ∀l@0:IdLnk.
. . . | . . | | | |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | | |    (<k,l>
. . . | . . | | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | |   ⇒
. . . | . . | | | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | |   =
. . . | . . | | | |   if source(l@0) = i→
. . . | . . | | | |      tagged-list-messages(λx.
. . . | . . | | | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | |   else nil fi
. . . | . . | | | 29. withlnk(l;m(i;t))
. . . | . . | | | |   =
. . . | . . | | | |   concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | | | |                          ,x>;2of(tgf)
. . . | . . | | | |                              ((λx.s(i;t).x)
. . . | . . | | | |                              ,val(a(i;t))));f))
. . . | . . | | | 30. source(l) = i
. . . | . . | | | ⊢  val(a(j;t)) = val(a(i;t))
. . . | . . | | | | by Assert (ma-valtype(da; k) ∈ Type)
. . . | . . | | | |
. . . | . . | | | |\
. . . | . . | | | | .....assertion .....
. . . | . . | | | | 1. i : Id
. . . | . . | | | | 2. k : Knd
. . . | . . | | | | 3. l : IdLnk
. . . | . . | | | | 4. ds : x:Id fp-> Type
. . . | . . | | | | 5. da : a:Knd fp-> Type
```

```
. . . | . . | | | | 6. f :
. . . | . . | | | |    (tg:Id
. . . | . . | | | |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | |     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | | 7. source(l) = i
. . . | . . | | | | 8. w : World
. . . | . . | | | | 9. p : FairFifo
. . . | . . | | | | 10. FairFifo
. . . | . . | | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | |     (w.M
. . . | . . | | | |      (l@0
. . . | . . | | | |      ,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | | |                  <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . . | . . | | | |               else  fi.da(rcv(l@0; tg))
. . . | . . | | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | | 14. ∀e:E.
. . . | . . | | | |     loc(e) = i
. . . | . . | | | |     ⇒
. . . | . . | | | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | 15. ∀e:E.
. . . | . . | | | |     isrcv(e)
. . . | . . | | | |     ⇒
. . . | . . | | | |     lnk(e) = l
. . . | . . | | | |     ⇒
. . . | . . | | | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | | 17. ∀a:Action(i).
. . . | . . | | | |     ¬isnull(a)
. . . | . . | | | |     ⇒
. . . | . . | | | |     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | | 18. ∀x:Id.
. . . | . . | | | |     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | |                                          ds))→
. . . | . . | | | |                        2of(ds)(x)
. . . | . . | | | |                     else Top fi
. . . | . . | | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | | 20. j : Id
. . . | . . | | | | 21. t : ℕ
. . . | . . | | | | 22. ¬isnull(a(j;t))
. . . | . . | | | | 23. True
. . . | . . | | | | 24. j = i
. . . | . . | | | | 25. kind(<j,t>) = k
. . . | . . | | | | 26. ¬isnull(a(i;t))
. . . | . . | | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | | 28. ∀l@0:IdLnk.
. . . | . . | | | |     (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | | |      (<k,l>
. . . | . . | | | |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | |     ⇒
. . . | . . | | | |     map(λms.2of(ms);filter(λms.
. . . | . . | | | |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | |     =
. . . | . . | | | |     if source(l@0) = i→
```

```
. . . | . . | | | |        tagged-list-messages(λx.
. . . | . . | | | |        s(i;t).x;val(a(i;t));f)
. . . | . . | | | |     else nil fi
. . . | . . | | | | 29. withlnk(l;m(i;t))
. . . | . . | | | |     =
. . . | . . | | | |     concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | | | |                             ,x>;2of(tgf)
. . . | . . | | | |                                 ((λx.s(i;t).x)
. . . | . . | | | |                                 ,val(a(i;t))));f))
. . . | . . | | | | 30. source(l) = i
. . . | . . | | | | ⊢  ma-valtype(da; k) ∈ Type
. . . | . . | | | |   by Unfold ‘ma-valtype‘ 0
. . . | . . | | | |
. . . | . . | | | .\
. . . | . . | | | . 1. i : Id
. . . | . . | | | . 2. k : Knd
. . . | . . | | | . 3. l : IdLnk
. . . | . . | | | . 4. ds : x:Id fp-> Type
. . . | . . | | | . 5. da : a:Knd fp-> Type
. . . | . . | | | . 6. f :
. . . | . . | | | . |  (tg:Id
. . . | . . | | | . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . 7. source(l) = i
. . . | . . | | | . 8. w : World
. . . | . . | | | . 9. p : FairFifo
. . . | . . | | | . 10. FairFifo
. . . | . . | | | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . |   (w.M
. . . | . . | | | . |    (l@0
. . . | . . | | | . |    ,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | | . |                <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . . | . . | | | . |             else  fi.da(rcv(l@0; tg))
. . . | . . | | | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . 14. ∀e:E.
. . . | . . | | | . |   loc(e) = i
. . . | . . | | | . |   ⇒
. . . | . . | | | . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . 15. ∀e:E.
. . . | . . | | | . |   isrcv(e)
. . . | . . | | | . |   ⇒
. . . | . . | | | . |   lnk(e) = l
. . . | . . | | | . |   ⇒
. . . | . . | | | . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . 17. ∀a:Action(i).
. . . | . . | | | . |   ¬isnull(a)
. . . | . . | | | . |   ⇒
. . . | . . | | | . |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . 18. ∀x:Id.
. . . | . . | | | . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | . |                                        ds))→
. . . | . . | | | . |                     2of(ds)(x)
```

```
. . . | . . | | | . |                    else Top fi
. . . | . . | | | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . 20. j : Id
. . . | . . | | | . 21. t : ℕ
. . . | . . | | | . 22. ¬isnull(a(j;t))
. . . | . . | | | . 23. True
. . . | . . | | | . 24. j = i
. . . | . . | | | . 25. kind(<j,t>) = k
. . . | . . | | | . 26. ¬isnull(a(i;t))
. . . | . . | | | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . 28. ∀l@0:IdLnk.
. . . | . . | | | . |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | | . |    (<k,l>
. . . | . . | | | . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . |   ⇒
. . . | . . | | | . |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . |   =
. . . | . . | | | . |   if source(l@0) = i→
. . . | . . | | | . |      tagged-list-messages(λx.
. . . | . . | | | . |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . |   else nil fi
. . . | . . | | | . 29. withlnk(l;m(i;t))
. . . | . . | | | . |   =
. . . | . . | | | . |   concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | | | . |                          ,x>;2of(tgf)
. . . | . . | | | . |                              ((λx.s(i;t).x)
. . . | . . | | | . |                              ,val(a(i;t))));f))
. . . | . . | | | . 30. source(l) = i
. . . | . . | | | . 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . ⊢  val(a(j;t)) = val(a(i;t))
. . . | . . | | | . | by Assert

. . . | . . | | | . |    (valtype(i;a(i;t)) ⊆r ma-valtype(da; k))
. . . | . . | | | . |
. . . | . . | | | . |\
. . . | . . | | | . | .....assertion .....
. . . | . . | | | . | 1. i : Id
. . . | . . | | | . | 2. k : Knd
. . . | . . | | | . | 3. l : IdLnk
. . . | . . | | | . | 4. ds : x:Id fp-> Type
. . . | . . | | | . | 5. da : a:Knd fp-> Type
. . . | . . | | | . | 6. f :
. . . | . . | | | . | |  (tg:Id
. . . | . . | | | . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | 7. source(l) = i
. . . | . . | | | . | 8. w : World
. . . | . . | | | . | 9. p : FairFifo
. . . | . . | | | . | 10. FairFifo
. . . | . . | | | . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | |                       (source(l@0)
. . . | . . | | | . | |                       ,i)→
. . . | . . | | | . | |                       <ds
. . . | . . | | | . | |                       ,da
```

```
. . . | . . | | | . | |                        ,
. . . | . . | | | . | |                        ,
. . . | . . | | | . | |                        ,
. . . | . . | | | . | |                        ,<[<k,l>],λx.f>
. . . | . . | | | . | |                        ,
. . . | . . | | | . | |                        ,
. . . | . . | | | . | |                        ,·>
. . . | . . | | | . | |                    else  fi.da(rcv(l@0; tg))
. . . | . . | | | . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | 14. ∀e:E.
. . . | . . | | | . | |   loc(e) = i
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | 15. ∀e:E.
. . . | . . | | | . | |   isrcv(e)
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   lnk(e) = l
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | 17. ∀a:Action(i).
. . . | . . | | | . | |   ¬isnull(a)
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | 18. ∀x:Id.
. . . | . . | | | . | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | . | |                                        ds))→
. . . | . . | | | . | |                      2of(ds)(x)
. . . | . . | | | . | |                    else Top fi
. . . | . . | | | . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | 20. j : Id
. . . | . . | | | . | 21. t : ℕ
. . . | . . | | | . | 22. ¬isnull(a(j;t))
. . . | . . | | | . | 23. True
. . . | . . | | | . | 24. j = i
. . . | . . | | | . | 25. kind(<j,t>) = k
. . . | . . | | | . | 26. ¬isnull(a(i;t))
. . . | . . | | | . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | 28. ∀l@0:IdLnk.
. . . | . . | | | . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . | |    (<k,l>
. . . | . . | | | . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | |   =
. . . | . . | | | . | |   if source(l@0) = i→
. . . | . . | | | . | |      tagged-list-messages(λx.
. . . | . . | | | . | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | |   else nil fi
. . . | . . | | | . | 29. withlnk(l;m(i;t))
. . . | . . | | | . | |   =
. . . | . . | | | . | |   concat(map(λtgf.map(λx.
```

```
. . . | . . | | | . | |   <1of(tgf),x>;2of(tgf)
. . . | . . | | | . | |               ((λx.s(i;t).x)
. . . | . . | | | . | |               ,val(a(i;t))));f))
. . . | . . | | | . | 30. source(l) = i
. . . | . . | | | . | 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . | ⊢  valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . | | by AllHyps
. . . | . . | | | . | |    (λh.
. . . | . . | | | . | |     InstHyp [<i,t>] h
. . . | . . | | | . | |     THENA
. . . | . . | | | . | |     Complete
. . . | . . | | | . | |     (Unfolds [‘w-E‘;‘w-loc‘] 0 THEN Reduce 0
. . . | . . | | | . | |      THEN
. . . | . . | | | . | |      Analyze
. . . | . . | | | . | |      THEN
. . . | . . | | | . | |      Reduce 0))
. . . | . . | | | . | |
. . . | . . | | | . | 1. i : Id
. . . | . . | | | . | 2. k : Knd
. . . | . . | | | . | 3. l : IdLnk
. . . | . . | | | . | 4. ds : x:Id fp-> Type
. . . | . . | | | . | 5. da : a:Knd fp-> Type
. . . | . . | | | . | 6. f :
. . . | . . | | | . | |  (tg:Id
. . . | . . | | | . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | 7. source(l) = i
. . . | . . | | | . | 8. w : World
. . . | . . | | | . | 9. p : FairFifo
. . . | . . | | | . | 10. FairFifo
. . . | . . | | | . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | |                       (source(l@0)
. . . | . . | | | . | |                       ,i)→
. . . | . . | | | . | |                       <ds
. . . | . . | | | . | |                       ,da
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,<[<k,l>],λx.f>
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,·>
. . . | . . | | | . | |                   else  fi.da(rcv(l@0; tg))
. . . | . . | | | . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | 14. ∀e:E.
. . . | . . | | | . | |   loc(e) = i
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | 15. ∀e:E.
. . . | . . | | | . | |   isrcv(e)
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   lnk(e) = l
. . . | . . | | | . | |   ⇒
```

```
. . . | . . | | | . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | 17. ∀a:Action(i).
. . . | . . | | | . | |   ¬isnull(a)
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | 18. ∀x:Id.
. . . | . . | | | . | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | . | |                                    ds))→
. . . | . . | | | . | |                      2of(ds)(x)
. . . | . . | | | . | |                   else Top fi
. . . | . . | | | . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | 20. j : Id
. . . | . . | | | . | 21. t : ℕ
. . . | . . | | | . | 22. ¬isnull(a(j;t))
. . . | . . | | | . | 23. True
. . . | . . | | | . | 24. j = i
. . . | . . | | | . | 25. kind(<j,t>) = k
. . . | . . | | | . | 26. ¬isnull(a(i;t))
. . . | . . | | | . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | 28. ∀l@0:IdLnk.
. . . | . . | | | . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . | |    (<k,l>
. . . | . . | | | . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | |   =
. . . | . . | | | . | |   if source(l@0) = i→
. . . | . . | | | . | |      tagged-list-messages(λx.
. . . | . . | | | . | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | |   else nil fi
. . . | . . | | | . | 29. withlnk(l;m(i;t))
. . . | . . | | | . | |   =
. . . | . . | | | . | |   concat(map(λtgf.map(λx.
. . . | . . | | | . | |   <1of(tgf),x>;2of(tgf)
. . . | . . | | | . | |                ((λx.s(i;t).x)
. . . | . . | | | . | |                ,val(a(i;t))));f))
. . . | . . | | | . | 30. source(l) = i
. . . | . . | | | . | 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . | 32. valtype(<i,t>) ⊆r ma-valtype
. . . | . . | | | . | |   (da; kind(<i,t>))
. . . | . . | | | . | ⊢  valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . |  | by NthHypEq -1 THEN Analyze
. . . | . . | | | . |  |    THEN
. . . | . . | | | . |  |    Try (Complete Auto)
. . . | . . | | | . |  |
. . . | . . | | | . |  |\
. . . | . . | | | . |  | .....subterm .....
. . . | . . | | | . |  | 1. i : Id
. . . | . . | | | . |  | 2. k : Knd
. . . | . . | | | . |  | 3. l : IdLnk
. . . | . . | | | . |  | 4. ds : x:Id fp-> Type
. . . | . . | | | . |  | 5. da : a:Knd fp-> Type
```

. . . | . . | | | . | | 6. f :
. . . | . . | | | . | | | (tg:Id
. . . | . . | | | . | | | ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | | (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | 7. source(l) = i
. . . | . . | | | . | | 8. w : World
. . . | . . | | | . | | 9. p : FairFifo
. . . | . . | | | . | | 10. FairFifo
. . . | . . | | | . | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | | (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | | (source(l@0)
. . . | . . | | | . | | | ,i)→
. . . | . . | | | . | | | <ds
. . . | . . | | | . | | | ,da
. . . | . . | | | . | | | ,
. . . | . . | | | . | | | ,
. . . | . . | | | . | | | ,
. . . | . . | | | . | | | ,<[<k,l>],λx.f>
. . . | . . | | | . | | | ,
. . . | . . | | | . | | | ,
. . . | . . | | | . | | | ,•>
. . . | . . | | | . | | | else fi.da(rcv(l@0; tg))
. . . | . . | | | . | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | 14. ∀e:E.
. . . | . . | | | . | | | loc(e) = i
. . . | . . | | | . | | | ⇒
. . . | . . | | | . | | | (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | 15. ∀e:E.
. . . | . . | | | . | | | isrcv(e)
. . . | . . | | | . | | | ⇒
. . . | . . | | | . | | | lnk(e) = l
. . . | . . | | | . | | | ⇒
. . . | . . | | | . | | | (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | | ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | 17. ∀a:Action(i).
. . . | . . | | | . | | | ¬isnull(a)
. . . | . . | | | . | | | ⇒
. . . | . . | | | . | | | (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | 18. ∀x:Id.
. . . | . . | | | . | | | vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | . | | | ds))→
. . . | . . | | | . | | | 2of(ds)(x)
. . . | . . | | | . | | | else Top fi
. . . | . . | | | . | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | 20. j : Id
. . . | . . | | | . | | 21. t : ℕ
. . . | . . | | | . | | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | 23. True
. . . | . . | | | . | | 24. j = i
. . . | . . | | | . | | 25. kind(<j,t>) = k
. . . | . . | | | . | | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | 28. ∀l@0:IdLnk.

```
. . . | . . | | | . | | |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . | | |    (<k,l>
. . . | . . | | | . | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | |   ⇒
. . . | . . | | | . | | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | |   =
. . . | . . | | | . | | |   if source(l@0) = i→
. . . | . . | | | . | | |      tagged-list-messages(λx.
. . . | . . | | | . | | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | |   else nil fi
. . . | . . | | | . | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | |   =
. . . | . . | | | . | | |   concat(map(λtgf.map(λx.
. . . | . . | | | . | | |   <1of(tgf),x>;2of(tgf)
. . . | . . | | | . | | |               ((λx.s(i;t).x)
. . . | . . | | | . | | |               ,val(a(i;t))));f))
. . . | . . | | | . | | 30. source(l) = i
. . . | . . | | | . | | 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . | | 32. valtype(<i,t>) ⊆r ma-valtype(da;
. . . | . . | | | . | | |                                kind(<i,t>))
. . . | . . | | | . | | ⊢  valtype(i;a(i;t)) = valtype(<i,t>)
. . . | . . | | | . | | | by Subst
. . . | . . | | | . | | |    (valtype(<i,t>) ∼ valtype(i;a(i;t)))
. . . | . . | | | . | | |    0
. . . | . . | | | . | | |
. . . | . . | | | . | | |\
. . . | . . | | | . | | | .....equality .....
. . . | . . | | | . | | | 1. i : Id
. . . | . . | | | . | | | 2. k : Knd
. . . | . . | | | . | | | 3. l : IdLnk
. . . | . . | | | . | | | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | | 5. da : a:Knd fp-> Type
. . . | . . | | | . | | | 6. f :
. . . | . . | | | . | | |    (tg:Id
. . . | . . | | | . | | |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | |     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | | 7. source(l) = i
. . . | . . | | | . | | | 8. w : World
. . . | . . | | | . | | | 9. p : FairFifo
. . . | . . | | | . | | | 10. FairFifo
. . . | . . | | | . | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | |                          (source(l@0)
. . . | . . | | | . | | |                          ,i)→
. . . | . . | | | . | | |                          <ds
. . . | . . | | | . | | |                          ,da
. . . | . . | | | . | | |                          ,
. . . | . . | | | . | | |                          ,
. . . | . . | | | . | | |                          ,
. . . | . . | | | . | | |                          ,<[<k,l>],λx.f>
. . . | . . | | | . | | |                          ,
. . . | . . | | | . | | |                          ,
. . . | . . | | | . | | |                          ,·>
. . . | . . | | | . | | |                     else  fi.da(rcv
```

```
. . . | . . | | | . | | |     (l@0; tg))
. . . | . . | | | . | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂

. . . | . . | | | . | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | | 14. ∀e:E.
. . . | . . | | | . | | |     loc(e) = i
. . . | . . | | | . | | |     ⇒
. . . | . . | | | . | | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | | 15. ∀e:E.
. . . | . . | | | . | | |     isrcv(e)
. . . | . . | | | . | | |     ⇒
. . . | . . | | | . | | |     lnk(e) = l
. . . | . . | | | . | | |     ⇒
. . . | . . | | | . | | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | | 17. ∀a:Action(i).
. . . | . . | | | . | | |     ¬isnull(a)
. . . | . . | | | . | | |     ⇒
. . . | . . | | | . | | |     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | | 18. ∀x:Id.
. . . | . . | | | . | | |     vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | |                         2of(ds)(x)
. . . | . . | | | . | | |                       else Top fi
. . . | . . | | | . | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | | 20. j : Id
. . . | . . | | | . | | | 21. t : ℕ
. . . | . . | | | . | | | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | | 23. True
. . . | . . | | | . | | | 24. j = i
. . . | . . | | | . | | | 25. kind(<j,t>) = k
. . . | . . | | | . | | | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | |     (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . | | |      (<k,l>
. . . | . . | | | . | | |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | |     ⇒
. . . | . . | | | . | | |     map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | |     =
. . . | . . | | | . | | |     if source(l@0) = i→
. . . | . . | | | . | | |        tagged-list-messages(λx.
. . . | . . | | | . | | |        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | |     else nil fi
. . . | . . | | | . | | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | |     =
. . . | . . | | | . | | |     concat(map(λtgf.map(λx.
. . . | . . | | | . | | |     <1of(tgf),x>;2of(tgf)
. . . | . . | | | . | | |                  ((λx.s(i;t).x)
. . . | . . | | | . | | |                  ,val(a(i;t))));f))
. . . | . . | | | . | | | 30. source(l) = i
. . . | . . | | | . | | | 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . | | | 32. valtype(<i,t>) ⊆r ma-valtype(da;
. . . | . . | | | . | | |                                  kind(<i,t>)
. . . | . . | | | . | | |                                  )
```

```
. . . | . . | | | . | | | ⊢  valtype(<i,t>) ∼ valtype(i;a(i;t))
. . . | . . | | | . | | |   by Fold ‘w-es‘ 0
. . . | . . | | | . | | |      THEN
. . . | . . | | | . | | |      BackThru
. . . | . . | | | . | | |      Thm* ∀i:Id, w:World, p:FairFifo, t:ℕ.
. . . | . . | | | . | | |           ¬isnull(a(i;t))
. . . | . . | | | . | | |           ⇒
. . . | . . | | | . | | |           (valtype(<i,t>) ∼ valtype(i;...))
. . . | . . | | | . | | |      {es_DASH_valtype_DASH_w_DASH_valtype}
. . . | . . | | | . | | |
. . . | . . | | | . | | .\
. . . | . . | | | . | | . 1. i : Id
. . . | . . | | | . | | . 2. k : Knd
. . . | . . | | | . | | . 3. l : IdLnk
. . . | . . | | | . | | . 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . 6. f :
. . . | . . | | | . | | .    (tg:Id
. . . | . . | | | . | | .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . 7. source(l) = i
. . . | . . | | | . | | . 8. w : World
. . . | . . | | | . | | . 9. p : FairFifo
. . . | . . | | | . | | . 10. FairFifo
. . . | . . | | | . | | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | .                         (source(l@0)
. . . | . . | | | . | | .                         ,i)→
. . . | . . | | | . | | .                         <ds
. . . | . . | | | . | | .                         ,da
. . . | . . | | | . | | .                         ,
. . . | . . | | | . | | .                         ,
. . . | . . | | | . | | .                         ,
. . . | . . | | | . | | .                         ,<[<k,l>],λx.f>
. . . | . . | | | . | | .                         ,
. . . | . . | | | . | | .                         ,
. . . | . . | | | . | | .                         ,•>
. . . | . . | | | . | | .                      else  fi.da(rcv
. . . | . . | | | . | | .     (l@0; tg))
. . . | . . | | | . | | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . 14. ∀e:E.
. . . | . . | | | . | | .     loc(e) = i
. . . | . . | | | . | | .     ⇒
. . . | . . | | | . | | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | . 15. ∀e:E.
. . . | . . | | | . | | .     isrcv(e)
. . . | . . | | | . | | .     ⇒
. . . | . . | | | . | | .     lnk(e) = l
. . . | . . | | | . | | .     ⇒
. . . | . . | | | . | | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . 17. ∀a:Action(i).
. . . | . . | | | . | | .     ¬isnull(a)
```

```
. . . | . . | | | . | | .     ⇒
. . . | . . | | | . | | .     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . 18. ∀x:Id.
. . . | . . | | | . | | .     vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | .                        2of(ds)(x)
. . . | . . | | | . | | .                      else Top fi
. . . | . . | | | . | | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . 20. j : Id
. . . | . . | | | . | | . 21. t : ℕ
. . . | . . | | | . | | . 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . 23. True
. . . | . . | | | . | | . 24. j = i
. . . | . . | | | . | | . 25. kind(<j,t>) = k
. . . | . . | | | . | | . 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . 28. ∀l@0:IdLnk.
. . . | . . | | | . | | .     (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . | | .      (<k,l>
. . . | . . | | | . | | .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | .     ⇒
. . . | . . | | | . | | .     map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | .     =
. . . | . . | | | . | | .     if source(l@0) = i→
. . . | . . | | | . | | .        tagged-list-messages(λx.
. . . | . . | | | . | | .        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | .     else nil fi
. . . | . . | | | . | | . 29. withlnk(l;m(i;t))
. . . | . . | | | . | | .     =
. . . | . . | | | . | | .     concat(map(λtgf.map(λx.
. . . | . . | | | . | | .     <1of(tgf),x>;2of(tgf)
. . . | . . | | | . | | .                  ((λx.s(i;t).x)
. . . | . . | | | . | | .                  ,val(a(i;t))));f))
. . . | . . | | | . | | . 30. source(l) = i
. . . | . . | | | . | | . 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . | | . 32. valtype(<i,t>) ⊆r ma-valtype(da;
. . . | . . | | | . | | .                                  kind(<i,t>)
. . . | . . | | | . | | .                                  )
. . . | . . | | | . | | . ⊢  valtype(i;a(i;t)) = valtype(i;a(i;t))
. . . | . . | | | . | | .   by Auto
. . . | . . | | | . | |
. . . | . . | | | . | .\
. . . | . . | | | . | . .....subterm .....
. . . | . . | | | . | . 1. i : Id
. . . | . . | | | . | . 2. k : Knd
. . . | . . | | | . | . 3. l : IdLnk
. . . | . . | | | . | . 4. ds : x:Id fp-> Type
. . . | . . | | | . | . 5. da : a:Knd fp-> Type
. . . | . . | | | . | . 6. f :
. . . | . . | | | . | . |  (tg:Id
. . . | . . | | | . | . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | . |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | . 7. source(l) = i
. . . | . . | | | . | . 8. w : World
. . . | . . | | | . | . 9. p : FairFifo
```

```
. . . | . . | | | . | . 10. FairFifo
. . . | . . | | | . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | . |                       (source(l@0)
. . . | . . | | | . | . |                       ,i)→
. . . | . . | | | . | . |                       <ds
. . . | . . | | | . | . |                       ,da
. . . | . . | | | . | . |                       ,
. . . | . . | | | . | . |                       ,
. . . | . . | | | . | . |                       ,
. . . | . . | | | . | . |                       ,<[<k,l>],λx.f>
. . . | . . | | | . | . |                       ,
. . . | . . | | | . | . |                       ,
. . . | . . | | | . | . |                       ,•>
. . . | . . | | | . | . |                   else  fi.da(rcv(l@0; tg))
. . . | . . | | | . | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | . 14. ∀e:E.
. . . | . . | | | . | . |   loc(e) = i
. . . | . . | | | . | . |   ⇒
. . . | . . | | | . | . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | . 15. ∀e:E.
. . . | . . | | | . | . |   isrcv(e)
. . . | . . | | | . | . |   ⇒
. . . | . . | | | . | . |   lnk(e) = l
. . . | . . | | | . | . |   ⇒
. . . | . . | | | . | . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | . 17. ∀a:Action(i).
. . . | . . | | | . | . |   ¬isnull(a)
. . . | . . | | | . | . |   ⇒
. . . | . . | | | . | . |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | . 18. ∀x:Id.
. . . | . . | | | . | . |   vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | . | . |                                        ds))→
. . . | . . | | | . | . |                     2of(ds)(x)

. . . | . . | | | . | . |                   else Top fi
. . . | . . | | | . | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | . 20. j : Id
. . . | . . | | | . | . 21. t : ℕ
. . . | . . | | | . | . 22. ¬isnull(a(j;t))
. . . | . . | | | . | . 23. True
. . . | . . | | | . | . 24. j = i
. . . | . . | | | . | . 25. kind(<j,t>) = k
. . . | . . | | | . | . 26. ¬isnull(a(i;t))
. . . | . . | | | . | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | . 28. ∀l@0:IdLnk.
. . . | . . | | | . | . |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . | . |    (<k,l>
. . . | . . | | | . | . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | . |   ⇒
. . . | . . | | | . | . |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | . |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | . |   =
```

```
. . . | . . | | | . | . |   if source(l@0) = i→
. . . | . . | | | . | . |      tagged-list-messages(λx.
. . . | . . | | | . | . |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | . |   else nil fi
. . . | . . | | | . | . 29. withlnk(l;m(i;t))
. . . | . . | | | . | . |   =
. . . | . . | | | . | . |   concat(map(λtgf.map(λx.
. . . | . . | | | . | . |   <1of(tgf),x>;2of(tgf)
. . . | . . | | | . | . |                ((λx.s(i;t).x)
. . . | . . | | | . | . |                ,val(a(i;t))));f))
. . . | . . | | | . | . 30. source(l) = i
. . . | . . | | | . | . 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . | . 32. valtype(<i,t>) ⊆r ma-valtype(da;
. . . | . . | | | . | . |                                kind(<i,t>))
. . . | . . | | | . | . ⊢  ma-valtype(da; k)
. . . | . . | | | . | . |  =
. . . | . . | | | . | . |  ma-valtype(da; kind(<i,t>))
. . . | . . | | | . | . | by Analyze
. . . | . . | | | . | . |    THEN
. . . | . . | | | . | . |    Assert (kind(<i,t>) = kind(<j,t>))
. . . | . . | | | . | . |    THEN
. . . | . . | | | . | . |    Analyze
. . . | . . | | | . | . |
. . . | . . | | | . | . .....subterm .....
. . . | . . | | | . | . 1. i : Id
. . . | . . | | | . | . 2. k : Knd
. . . | . . | | | . | . 3. l : IdLnk
. . . | . . | | | . | . 4. ds : x:Id fp-> Type
. . . | . . | | | . | . 5. da : a:Knd fp-> Type
. . . | . . | | | . | . 6. f :
. . . | . . | | | . | .    (tg:Id
. . . | . . | | | . | .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | . 7. source(l) = i
. . . | . . | | | . | . 8. w : World
. . . | . . | | | . | . 9. p : FairFifo
. . . | . . | | | . | . 10. FairFifo
. . . | . . | | | . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | .                        (source(l@0)
. . . | . . | | | . | .                        ,i)→
. . . | . . | | | . | .                        <ds
. . . | . . | | | . | .                        ,da
. . . | . . | | | . | .                        ,
. . . | . . | | | . | .                        ,
. . . | . . | | | . | .                        ,
. . . | . . | | | . | .                        ,<[<k,l>],λx.f>
. . . | . . | | | . | .                        ,
. . . | . . | | | . | .                        ,
. . . | . . | | | . | .                        ,·>
. . . | . . | | | . | .                    else  fi.da(rcv(l@0; tg))
. . . | . . | | | . | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | . 14. ∀e:E.
. . . | . . | | | . | .     loc(e) = i
```

```
. . . | . . | | | . | .     ⇒
. . . | . . | | | . | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | . 15. ∀e:E.
. . . | . . | | | . | .     isrcv(e)
. . . | . . | | | . | .     ⇒
. . . | . . | | | . | .     lnk(e) = l
. . . | . . | | | . | .     ⇒
. . . | . . | | | . | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | . 17. ∀a:Action(i).
. . . | . . | | | . | .     ¬isnull(a)
. . . | . . | | | . | .     ⇒
. . . | . . | | | . | .     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | . 18. ∀x:Id.
. . . | . . | | | . | .     vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | . | .                                          ds))→
. . . | . . | | | . | .                        2of(ds)(x)
. . . | . . | | | . | .                    else Top fi
. . . | . . | | | . | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | . 20. j : Id
. . . | . . | | | . | . 21. t : ℕ
. . . | . . | | | . | . 22. ¬isnull(a(j;t))
. . . | . . | | | . | . 23. True
. . . | . . | | | . | . 24. j = i
. . . | . . | | | . | . 25. kind(<j,t>) = k
. . . | . . | | | . | . 26. ¬isnull(a(i;t))
. . . | . . | | | . | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | . 28. ∀l@0:IdLnk.
. . . | . . | | | . | .     (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . | .      (<k,l>
. . . | . . | | | . | .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | .     ⇒
. . . | . . | | | . | .     map(λms.2of(ms);filter(λms.
. . . | . . | | | . | .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | .     =
. . . | . . | | | . | .     if source(l@0) = i→
. . . | . . | | | . | .        tagged-list-messages(λx.
. . . | . . | | | . | .        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | .     else nil fi
. . . | . . | | | . | . 29. withlnk(l;m(i;t))
. . . | . . | | | . | .     =
. . . | . . | | | . | .     concat(map(λtgf.map(λx.
. . . | . . | | | . | .     <1of(tgf),x>;2of(tgf)
. . . | . . | | | . | .                  ((λx.s(i;t).x)
. . . | . . | | | . | .                  ,val(a(i;t))));f))
. . . | . . | | | . | . 30. source(l) = i
. . . | . . | | | . | . 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . | . 32. valtype(<i,t>) ⊆r ma-valtype(da;
. . . | . . | | | . | .                                  kind(<i,t>))
. . . | . . | | | . | . ⊢  <i,t> = <j,t>
. . . | . . | | | . | .   by Unfold `w-E` 0 THEN Analyze THEN Reduce 0
. . . | . . | | | . | .      THEN
. . . | . . | | | . | .      Analyze
. . . | . . | | | . |
```

```
. . . | . . | | | . .\
. . . | . . | | | . . 1. i : Id
. . . | . . | | | . . 2. k : Knd
. . . | . . | | | . . 3. l : IdLnk
. . . | . . | | | . . 4. ds : x:Id fp-> Type
. . . | . . | | | . . 5. da : a:Knd fp-> Type
. . . | . . | | | . . 6. f :
. . . | . . | | | . . |  (tg:Id
. . . | . . | | | . . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . 7. source(l) = i
. . . | . . | | | . . 8. w : World
. . . | . . | | | . . 9. p : FairFifo
. . . | . . | | | . . 10. FairFifo
. . . | . . | | | . . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . |                       (source(l@0)
. . . | . . | | | . . |                       ,i)→
. . . | . . | | | . . |                       <ds
. . . | . . | | | . . |                       ,da
. . . | . . | | | . . |                       ,
. . . | . . | | | . . |                       ,
. . . | . . | | | . . |                       ,
. . . | . . | | | . . |                       ,<[<k,l>],λx.f>
. . . | . . | | | . . |                       ,
. . . | . . | | | . . |                       ,
. . . | . . | | | . . |                       ,.>
. . . | . . | | | . . |                     else  fi.da(rcv(l@0; tg))
. . . | . . | | | . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . 14. ∀e:E.
. . . | . . | | | . . |   loc(e) = i
. . . | . . | | | . . |   ⇒
. . . | . . | | | . . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . 15. ∀e:E.
. . . | . . | | | . . |   isrcv(e)
. . . | . . | | | . . |   ⇒
. . . | . . | | | . . |   lnk(e) = l
. . . | . . | | | . . |   ⇒
. . . | . . | | | . . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . 17. ∀a:Action(i).
. . . | . . | | | . . |   ¬isnull(a)
. . . | . . | | | . . |   ⇒
. . . | . . | | | . . |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . 18. ∀x:Id.
. . . | . . | | | . . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | . . |                                          ds))→
. . . | . . | | | . . |                       2of(ds)(x)
. . . | . . | | | . . |                     else Top fi
. . . | . . | | | . . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . 20. j : Id
. . . | . . | | | . . 21. t : ℕ
. . . | . . | | | . . 22. ¬isnull(a(j;t))
```

```
. . . | . . | | | . . 23. True
. . . | . . | | | . . 24. j = i
. . . | . . | | | . . 25. kind(<j,t>) = k
. . . | . . | | | . . 26. ¬isnull(a(i;t))
. . . | . . | | | . . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . 28. ∀l@0:IdLnk.
. . . | . . | | | . . |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . . |    (<k,l>
. . . | . . | | | . . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . |   ⇒
. . . | . . | | | . . |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . . |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . |   =
. . . | . . | | | . . |   if source(l@0) = i→
. . . | . . | | | . . |      tagged-list-messages(λx.
. . . | . . | | | . . |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . |   else nil fi

. . . | . . | | | . . 29. withlnk(l;m(i;t))
. . . | . . | | | . . |   =
. . . | . . | | | . . |   concat(map(λtgf.map(λx.
. . . | . . | | | . . |   <1of(tgf),x>;2of(tgf)
. . . | . . | | | . . |                ((λx.s(i;t).x)
. . . | . . | | | . . |                ,val(a(i;t))));f))
. . . | . . | | | . . 30. source(l) = i
. . . | . . | | | . . 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . . 32. valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . . ⊢  val(a(j;t)) = val(a(i;t))
. . . | . . | | | . . | by Assert (a(j;t) = a(i;t))
. . . | . . | | | . . |
. . . | . . | | | . . |\
. . . | . . | | | . . | .....assertion .....
. . . | . . | | | . . | 1. i : Id
. . . | . . | | | . . | 2. k : Knd
. . . | . . | | | . . | 3. l : IdLnk
. . . | . . | | | . . | 4. ds : x:Id fp-> Type
. . . | . . | | | . . | 5. da : a:Knd fp-> Type
. . . | . . | | | . . | 6. f :
. . . | . . | | | . . | |  (tg:Id
. . . | . . | | | . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . | 7. source(l) = i
. . . | . . | | | . . | 8. w : World
. . . | . . | | | . . | 9. p : FairFifo
. . . | . . | | | . . | 10. FairFifo
. . . | . . | | | . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . | |                      (source(l@0)
. . . | . . | | | . . | |                      ,i)→
. . . | . . | | | . . | |                      <ds
. . . | . . | | | . . | |                      ,da
. . . | . . | | | . . | |                      ,
. . . | . . | | | . . | |                      ,
. . . | . . | | | . . | |                      ,
. . . | . . | | | . . | |                      ,<[<k,l>],λx.f>
. . . | . . | | | . . | |                      ,
```

```
. . . | . . | | | . . | |                          ,
. . . | . . | | | . . | |                          ,·>
. . . | . . | | | . . | |                       else  fi.da(rcv(l@0; tg))
. . . | . . | | | . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . | 14. ∀e:E.
. . . | . . | | | . . | |   loc(e) = i
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | 15. ∀e:E.
. . . | . . | | | . . | |   isrcv(e)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   lnk(e) = l
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . | 17. ∀a:Action(i).
. . . | . . | | | . . | |   ¬isnull(a)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . | 18. ∀x:Id.
. . . | . . | | | . . | |   vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | . . | |                                        ds))→
. . . | . . | | | . . | |                        2of(ds)(x)
. . . | . . | | | . . | |                     else Top fi
. . . | . . | | | . . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . | 20. j : Id
. . . | . . | | | . . | 21. t : ℕ
. . . | . . | | | . . | 22. ¬isnull(a(j;t))
. . . | . . | | | . . | 23. True
. . . | . . | | | . . | 24. j = i
. . . | . . | | | . . | 25. kind(<j,t>) = k
. . . | . . | | | . . | 26. ¬isnull(a(i;t))
. . . | . . | | | . . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . | 28. ∀l@0:IdLnk.
. . . | . . | | | . . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . . | |    (<k,l>
. . . | . . | | | . . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . . | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . | |   =
. . . | . . | | | . . | |   if source(l@0) = i→
. . . | . . | | | . . | |      tagged-list-messages(λx.
. . . | . . | | | . . | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . | |   else nil fi
. . . | . . | | | . . | 29. withlnk(l;m(i;t))
. . . | . . | | | . . | |   =
. . . | . . | | | . . | |   concat(map(λtgf.map(λx.
. . . | . . | | | . . | |   <1of(tgf),x>;2of(tgf)
. . . | . . | | | . . | |                ((λx.s(i;t).x)
. . . | . . | | | . . | |                ,val(a(i;t))));f))
. . . | . . | | | . . | 30. source(l) = i
. . . | . . | | | . . | 31. ma-valtype(da; k) ∈ Type
```

```
. . . | . . | | | . . | 32. valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . . | ⊢  a(j;t) = a(i;t)  by Analyze
. . . | . . | | | . . | |
. . . | . . | | | . . | 1. i : Id
. . . | . . | | | . . | 2. k : Knd
. . . | . . | | | . . | 3. l : IdLnk
. . . | . . | | | . . | 4. ds : x:Id fp-> Type
. . . | . . | | | . . | 5. da : a:Knd fp-> Type
. . . | . . | | | . . | 6. f :
. . . | . . | | | . . | |  (tg:Id
. . . | . . | | | . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . | 7. source(l) = i
. . . | . . | | | . . | 8. w : World
. . . | . . | | | . . | 9. p : FairFifo
. . . | . . | | | . . | 10. FairFifo
. . . | . . | | | . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . | |                       (source(l@0)
. . . | . . | | | . . | |                       ,i)→
. . . | . . | | | . . | |                       <ds
. . . | . . | | | . . | |                       ,da
. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,<[<k,l>],λx.f>
. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,•>
. . . | . . | | | . . | |                    else  fi.da(rcv(l@0; tg))
. . . | . . | | | . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . | 14. ∀e:E.
. . . | . . | | | . . | |   loc(e) = i
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | 15. ∀e:E.
. . . | . . | | | . . | |   isrcv(e)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   lnk(e) = l
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . | 17. ∀a:Action(i).
. . . | . . | | | . . | |   ¬isnull(a)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . | 18. ∀x:Id.
. . . | . . | | | . . | |   vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | . . | |                                        ds))→
. . . | . . | | | . . | |                      2of(ds)(x)
. . . | . . | | | . . | |                   else Top fi
. . . | . . | | | . . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . | 20. j : Id
```

```
. . . | . . | | | . . | 21. t : ℕ
. . . | . . | | | . . | 22. ¬isnull(a(j;t))
. . . | . . | | | . . | 23. True
. . . | . . | | | . . | 24. j = i
. . . | . . | | | . . | 25. kind(<j,t>) = k
. . . | . . | | | . . | 26. ¬isnull(a(i;t))
. . . | . . | | | . . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . | 28. ∀l@0:IdLnk.
. . . | . . | | | . . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . . | |    (<k,l>
. . . | . . | | | . . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . . | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . | |   =
. . . | . . | | | . . | |   if source(l@0) = i→
. . . | . . | | | . . | |      tagged-list-messages(λx.
. . . | . . | | | . . | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . | |   else nil fi
. . . | . . | | | . . | 29. withlnk(l;m(i;t))
. . . | . . | | | . . | |   =
. . . | . . | | | . . | |   concat(map(λtgf.map(λx.
. . . | . . | | | . . | |   <1of(tgf),x>;2of(tgf)
. . . | . . | | | . . | |               ((λx.s(i;t).x)
. . . | . . | | | . . | |               ,val(a(i;t))));f))
. . . | . . | | | . . | 30. source(l) = i
. . . | . . | | | . . | 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . . | 32. valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . . | ⊢  a(j;t) = a(i;t)  by StrongHypSubst -9 0
. . . | . . | | | . . | |
. . . | . . | | | . . | |\
. . . | . . | | | . . | | 1. i : Id
. . . | . . | | | . . | | 2. k : Knd
. . . | . . | | | . . | | 3. l : IdLnk
. . . | . . | | | . . | | 4. ds : x:Id fp-> Type
. . . | . . | | | . . | | 5. da : a:Knd fp-> Type
. . . | . . | | | . . | | 6. f :
. . . | . . | | | . . | |    (tg:Id
. . . | . . | | | . . | |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . | |     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . | | 7. source(l) = i
. . . | . . | | | . . | | 8. w : World
. . . | . . | | | . . | | 9. p : FairFifo
. . . | . . | | | . . | | 10. FairFifo
. . . | . . | | | . . | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . | |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . | |                          (source(l@0)
. . . | . . | | | . . | |                          ,i)→
. . . | . . | | | . . | |                          <ds
. . . | . . | | | . . | |                          ,da
. . . | . . | | | . . | |                          ,
. . . | . . | | | . . | |                          ,
. . . | . . | | | . . | |                          ,
. . . | . . | | | . . | |                          ,<[<k,l>],λx.f>
. . . | . . | | | . . | |                          ,
```

. . . | . . | | | . . | | ,
. . . | . . | | | . . | | ,·>
. . . | . . | | | . . | | else fi.da(rcv
. . . | . . | | | . . | | (l@0; tg))
. . . | . . | | | . . | | 12. (eqof(IdDeq)(i,i)) ∼ true$_2$
. . . | . . | | | . . | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . | | 14. ∀e:E.
. . . | . . | | | . . | | loc(e) = i
. . . | . . | | | . . | | ⇒
. . . | . . | | | . . | | (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | | 15. ∀e:E.
. . . | . . | | | . . | | isrcv(e)
. . . | . . | | | . . | | ⇒
. . . | . . | | | . . | | lnk(e) = l
. . . | . . | | | . . | | ⇒
. . . | . . | | | . . | | (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . | | ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . | | 17. ∀a:Action(i).
. . . | . . | | | . . | | ¬isnull(a)
. . . | . . | | | . . | | ⇒
. . . | . . | | | . . | | (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . | | 18. ∀x:Id.
. . . | . . | | | . . | | vartype(i;x) ⊆r if ...→
. . . | . . | | | . . | | 2of(ds)(x)
. . . | . . | | | . . | | else Top fi
. . . | . . | | | . . | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . | | 20. j : Id
. . . | . . | | | . . | | 21. t : ℕ
. . . | . . | | | . . | | 22. ¬isnull(a(j;t))
. . . | . . | | | . . | | 23. True
. . . | . . | | | . . | | 24. j = i
. . . | . . | | | . . | | 25. kind(<j,t>) = k
. . . | . . | | | . . | | 26. ¬isnull(a(i;t))
. . . | . . | | | . . | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . | | 28. ∀l@0:IdLnk.
. . . | . . | | | . . | | (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . . | | (<k,l>
. . . | . . | | | . . | | ,<kind(a(i;t)),l@0>) ∨$_2$ false$_2$)
. . . | . . | | | . . | | ⇒
. . . | . . | | | . . | | map(λms.2of(ms);filter(λms.
. . . | . . | | | . . | | mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . | | =
. . . | . . | | | . . | | if source(l@0) = i→
. . . | . . | | | . . | | tagged-list-messages(λx.
. . . | . . | | | . . | | s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . | | else nil fi
. . . | . . | | | . . | | 29. withlnk(l;m(i;t))
. . . | . . | | | . . | | =
. . . | . . | | | . . | | concat(map(λtgf.map(λx.
. . . | . . | | | . . | | <1of(tgf),x>;2of(tgf)
. . . | . . | | | . . | | ((λx.s(i;t).x)
. . . | . . | | | . . | | ,val(a(i;t))));f))
. . . | . . | | | . . | | 30. source(l) = i
. . . | . . | | | . . | | 31. ma-valtype(da; k) ∈ Type

```
. . . | . . | | | . . | | 32. valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . . | | ⊢  a(i;t) = a(i;t)  by Auto
. . . | . . | | | . . | |
. . . | . . | | | . . | .\
. . . | . . | | | . . | . 1. i : Id
. . . | . . | | | . . | . 2. k : Knd
. . . | . . | | | . . | . 3. l : IdLnk
. . . | . . | | | . . | . 4. ds : x:Id fp-> Type
. . . | . . | | | . . | . 5. da : a:Knd fp-> Type
. . . | . . | | | . . | . 6. f :
. . . | . . | | | . . | .    (tg:Id
. . . | . . | | | . . | .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . | .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . | . 7. source(l) = i
. . . | . . | | | . . | . 8. w : World
. . . | . . | | | . . | . 9. p : FairFifo
. . . | . . | | | . . | . 10. FairFifo
. . . | . . | | | . . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . | .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . | .                         (source(l@0)
. . . | . . | | | . . | .                         ,i)→
. . . | . . | | | . . | .                         <ds
. . . | . . | | | . . | .                         ,da
. . . | . . | | | . . | .                         ,
. . . | . . | | | . . | .                         ,
. . . | . . | | | . . | .                         ,
. . . | . . | | | . . | .                         ,<[<k,l>],λx.f>
. . . | . . | | | . . | .                         ,
. . . | . . | | | . . | .                         ,
. . . | . . | | | . . | .                         ,•>
. . . | . . | | | . . | .                    else  fi.da(rcv
. . . | . . | | | . . | .     (l@0; tg))
. . . | . . | | | . . | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . | . 14. ∀e:E.
. . . | . . | | | . . | .     loc(e) = i
. . . | . . | | | . . | .     ⇒
. . . | . . | | | . . | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | . 15. ∀e:E.
. . . | . . | | | . . | .     isrcv(e)
. . . | . . | | | . . | .     ⇒
. . . | . . | | | . . | .     lnk(e) = l

. . . | . . | | | . . | .     ⇒
. . . | . . | | | . . | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . | .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . | . 17. ∀a:Action(i).
. . . | . . | | | . . | .     ¬isnull(a)
. . . | . . | | | . . | .     ⇒
. . . | . . | | | . . | .     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . | . 18. ∀x:Id.
. . . | . . | | | . . | .     vartype(i;x) ⊆r if ...→
. . . | . . | | | . . | .                         2of(ds)(x)
. . . | . . | | | . . | .                    else Top fi
. . . | . . | | | . . | . 19. IdDeq ∈ EqDecider(Id)
```

```
. . . | . . | | | . . | . 20. j : Id
. . . | . . | | | . . | . 21. t : ℕ
. . . | . . | | | . . | . 22. ¬isnull(a(j;t))
. . . | . . | | | . . | . 23. True
. . . | . . | | | . . | . 24. j = i
. . . | . . | | | . . | . 25. kind(<j,t>) = k
. . . | . . | | | . . | . 26. ¬isnull(a(i;t))
. . . | . . | | | . . | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . | . 28. ∀l@0:IdLnk.
. . . | . . | | | . . | .     (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . . | .      (<k,l>
. . . | . . | | | . . | .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . | .     ⇒
. . . | . . | | | . . | .     map(λms.2of(ms);filter(λms.
. . . | . . | | | . . | .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . | .     =
. . . | . . | | | . . | .     if source(l@0) = i→
. . . | . . | | | . . | .        tagged-list-messages(λx.
. . . | . . | | | . . | .        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . | .     else nil fi
. . . | . . | | | . . | . 29. withlnk(l;m(i;t))
. . . | . . | | | . . | .     =
. . . | . . | | | . . | .     concat(map(λtgf.map(λx.
. . . | . . | | | . . | .     <1of(tgf),x>;2of(tgf)
. . . | . . | | | . . | .                 ((λx.s(i;t).x)
. . . | . . | | | . . | .                 ,val(a(i;t))));f))
. . . | . . | | | . . | . 30. source(l) = i
. . . | . . | | | . . | . 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . . | . 32. valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . . | . 33. z : Id
. . . | . . | | | . . | . 34. z = i
. . . | . . | | | . . | . ⊢  (a(z;t) = a(i;t)) ∈ Prop
. . . | . . | | | . . | .   by Auto THEN DoSubsume THEN HypSubst -1 0
. . . | . . | | | . . |
. . . | . . | | | . . .\
. . . | . . | | | . . . 1. i : Id
. . . | . . | | | . . . 2. k : Knd
. . . | . . | | | . . . 3. l : IdLnk
. . . | . . | | | . . . 4. ds : x:Id fp-> Type
. . . | . . | | | . . . 5. da : a:Knd fp-> Type
. . . | . . | | | . . . 6. f :
. . . | . . | | | . . . |  (tg:Id
. . . | . . | | | . . . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . . |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . . 7. source(l) = i
. . . | . . | | | . . . 8. w : World
. . . | . . | | | . . . 9. p : FairFifo
. . . | . . | | | . . . 10. FairFifo
. . . | . . | | | . . . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . . |                     (source(l@0)
. . . | . . | | | . . . |                     ,i)→
. . . | . . | | | . . . |                     <ds
. . . | . . | | | . . . |                     ,da
. . . | . . | | | . . . |                     ,
```

```
. . . | . . | | | . . . |                        ,
. . . | . . | | | . . . |                        ,
. . . | . . | | | . . . |                        ,<[<k,l>],λx.f>
. . . | . . | | | . . . |                        ,
. . . | . . | | | . . . |                        ,
. . . | . . | | | . . . |                        ,.>
. . . | . . | | | . . . |                    else  fi.da(rcv(l@0; tg))
. . . | . . | | | . . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . . 14. ∀e:E.
. . . | . . | | | . . . |   loc(e) = i
. . . | . . | | | . . . |   ⇒
. . . | . . | | | . . . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . . 15. ∀e:E.
. . . | . . | | | . . . |   isrcv(e)
. . . | . . | | | . . . |   ⇒
. . . | . . | | | . . . |   lnk(e) = l
. . . | . . | | | . . . |   ⇒
. . . | . . | | | . . . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . . 17. ∀a:Action(i).
. . . | . . | | | . . . |   ¬isnull(a)
. . . | . . | | | . . . |   ⇒
. . . | . . | | | . . . |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . . 18. ∀x:Id.
. . . | . . | | | . . . |   vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | . . . |                                        ds))→
. . . | . . | | | . . . |                      2of(ds)(x)
. . . | . . | | | . . . |                    else Top fi
. . . | . . | | | . . . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . . 20. j : Id
. . . | . . | | | . . . 21. t : ℕ
. . . | . . | | | . . . 22. ¬isnull(a(j;t))
. . . | . . | | | . . . 23. True
. . . | . . | | | . . . 24. j = i
. . . | . . | | | . . . 25. kind(<j,t>) = k
. . . | . . | | | . . . 26. ¬isnull(a(i;t))
. . . | . . | | | . . . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . . 28. ∀l@0:IdLnk.
. . . | . . | | | . . . |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . . . |    (<k,l>
. . . | . . | | | . . . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . . |   ⇒
. . . | . . | | | . . . |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . . . |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . . |   =
. . . | . . | | | . . . |   if source(l@0) = i→
. . . | . . | | | . . . |      tagged-list-messages(λx.
. . . | . . | | | . . . |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . . |   else nil fi
. . . | . . | | | . . . 29. withlnk(l;m(i;t))
. . . | . . | | | . . . |   =
. . . | . . | | | . . . |   concat(map(λtgf.map(λx.
. . . | . . | | | . . . |   <1of(tgf),x>;2of(tgf)
```

```
. . . | . . | | | . . . |                ((λx.s(i;t).x)
. . . | . . | | | . . . |                ,val(a(i;t))));f))
. . . | . . | | | . . . 30. source(l) = i
. . . | . . | | | . . . 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . . . 32. valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . . . 33. a(j;t) = a(i;t)
. . . | . . | | | . . . ⊢  val(a(j;t)) = val(a(i;t))
. . . | . . | | | . . . | by Assert (val(a(j;t)) = val(a(i;t)))
. . . | . . | | | . . . |
. . . | . . | | | . . . |\
. . . | . . | | | . . . | .....assertion .....
. . . | . . | | | . . . | 1. i : Id
. . . | . . | | | . . . | 2. k : Knd
. . . | . . | | | . . . | 3. l : IdLnk
. . . | . . | | | . . . | 4. ds : x:Id fp-> Type
. . . | . . | | | . . . | 5. da : a:Knd fp-> Type
. . . | . . | | | . . . | 6. f :
. . . | . . | | | . . . | |  (tg:Id
. . . | . . | | | . . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . . | 7. source(l) = i
. . . | . . | | | . . . | 8. w : World
. . . | . . | | | . . . | 9. p : FairFifo
. . . | . . | | | . . . | 10. FairFifo
. . . | . . | | | . . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . . | |                       (source(l@0)
. . . | . . | | | . . . | |                       ,i)→
. . . | . . | | | . . . | |                       <ds
. . . | . . | | | . . . | |                       ,da
. . . | . . | | | . . . | |                       ,
. . . | . . | | | . . . | |                       ,
. . . | . . | | | . . . | |                       ,
. . . | . . | | | . . . | |                       ,<[<k,l>],λx.f>
. . . | . . | | | . . . | |                       ,
. . . | . . | | | . . . | |                       ,
. . . | . . | | | . . . | |                       ,·>
. . . | . . | | | . . . | |                   else  fi.da(rcv
. . . | . . | | | . . . | |   (l@0; tg))
. . . | . . | | | . . . | 12. (eqof(IdDeq)(i,i)) ∼ true2
. . . | . . | | | . . . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . . | 14. ∀e:E.
. . . | . . | | | . . . | |   loc(e) = i
. . . | . . | | | . . . | |   ⇒
. . . | . . | | | . . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . . | 15. ∀e:E.
. . . | . . | | | . . . | |   isrcv(e)
. . . | . . | | | . . . | |   ⇒
. . . | . . | | | . . . | |   lnk(e) = l
. . . | . . | | | . . . | |   ⇒
. . . | . . | | | . . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . . | 17. ∀a:Action(i).
. . . | . . | | | . . . | |   ¬isnull(a)
```

```
. . . | . . | | | . . . | |   ⇒
. . . | . . | | | . . . | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . . | 18. ∀x:Id.
. . . | . . | | | . . . | |   vartype(i;x) ⊆r if ...→
. . . | . . | | | . . . | |                    2of(ds)(x)
. . . | . . | | | . . . | |                  else Top fi
. . . | . . | | | . . . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . . | 20. j : Id
. . . | . . | | | . . . | 21. t : ℕ
. . . | . . | | | . . . | 22. ¬isnull(a(j;t))
. . . | . . | | | . . . | 23. True
. . . | . . | | | . . . | 24. j = i
. . . | . . | | | . . . | 25. kind(<j,t>) = k
. . . | . . | | | . . . | 26. ¬isnull(a(i;t))
. . . | . . | | | . . . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . . | 28. ∀l@0:IdLnk.
. . . | . . | | | . . . | |   (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . . . | |    (<k,l>
. . . | . . | | | . . . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . . | |   ⇒
. . . | . . | | | . . . | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . . . | |   mlnk(ms) = l@0;m(i;t)))

. . . | . . | | | . . . | |   =
. . . | . . | | | . . . | |   if source(l@0) = i→
. . . | . . | | | . . . | |      tagged-list-messages(λx.
. . . | . . | | | . . . | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . . | |   else nil fi
. . . | . . | | | . . . | 29. withlnk(l;m(i;t))
. . . | . . | | | . . . | |   =
. . . | . . | | | . . . | |   concat(map(λtgf.map(λx.
. . . | . . | | | . . . | |   <1of(tgf),x>;2of(tgf)
. . . | . . | | | . . . | |                ((λx.s(i;t).x)
. . . | . . | | | . . . | |                ,val(a(i;t))));f))
. . . | . . | | | . . . | 30. source(l) = i
. . . | . . | | | . . . | 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . . . | 32. valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . . . | 33. a(j;t) = a(i;t)
. . . | . . | | | . . . | ⊢ val(a(j;t)) = val(a(i;t))
. . . | . . | | | . . . | | by StrongHypSubst -1 0
. . . | . . | | | . . . | |
. . . | . . | | | . . . | |\
. . . | . . | | | . . . | | 1. i : Id
. . . | . . | | | . . . | | 2. k : Knd
. . . | . . | | | . . . | | 3. l : IdLnk
. . . | . . | | | . . . | | 4. ds : x:Id fp-> Type
. . . | . . | | | . . . | | 5. da : a:Knd fp-> Type
. . . | . . | | | . . . | | 6. f :
. . . | . . | | | . . . | |    (tg:Id
. . . | . . | | | . . . | |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . . | |     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . . | | 7. source(l) = i
. . . | . . | | | . . . | | 8. w : World
. . . | . . | | | . . . | | 9. p : FairFifo
. . . | . . | | | . . . | | 10. FairFifo
. . . | . . | | | . . . | | 11. ∀l@0:IdLnk, tg:Id.
```

```
. . . | . . | | | . . . | |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . . | |                          (source(l@0)
. . . | . . | | | . . . | |                          ,i)→
. . . | . . | | | . . . | |                          <ds
. . . | . . | | | . . . | |                          ,da
. . . | . . | | | . . . | |                          ,
. . . | . . | | | . . . | |                          ,
. . . | . . | | | . . . | |                          ,
. . . | . . | | | . . . | |                          ,<[<k,l>],λx.f>
. . . | . . | | | . . . | |                          ,
. . . | . . | | | . . . | |                          ,
. . . | . . | | | . . . | |                          ,·>
. . . | . . | | | . . . | |                       else  fi.da(rcv(l@0;
. . . | . . | | | . . . | |                                              tg))
. . . | . . | | | . . . | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . . | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . . | | 14. ∀e:E.
. . . | . . | | | . . . | |     loc(e) = i
. . . | . . | | | . . . | |     ⇒
. . . | . . | | | . . . | |     (valtype(e) ⊆r ma-valtype
. . . | . . | | | . . . | |      (da; kind(e)))
. . . | . . | | | . . . | | 15. ∀e:E.
. . . | . . | | | . . . | |     isrcv(e)
. . . | . . | | | . . . | |     ⇒
. . . | . . | | | . . . | |     lnk(e) = l
. . . | . . | | | . . . | |     ⇒
. . . | . . | | | . . . | |     (valtype(e) ⊆r ma-valtype
. . . | . . | | | . . . | |      (da; kind(e)))
. . . | . . | | | . . . | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . . | |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . . | | 17. ∀a:Action(i).
. . . | . . | | | . . . | |     ¬isnull(a)
. . . | . . | | | . . . | |     ⇒
. . . | . . | | | . . . | |     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . . | | 18. ∀x:Id.
. . . | . . | | | . . . | |     vartype(i;x) ⊆r if ...→
. . . | . . | | | . . . | |                          2of(ds)(x)
. . . | . . | | | . . . | |                       else Top fi
. . . | . . | | | . . . | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . . | | 20. j : Id
. . . | . . | | | . . . | | 21. t : ℕ
. . . | . . | | | . . . | | 22. ¬isnull(a(j;t))
. . . | . . | | | . . . | | 23. True
. . . | . . | | | . . . | | 24. j = i
. . . | . . | | | . . . | | 25. kind(<j,t>) = k
. . . | . . | | | . . . | | 26. ¬isnull(a(i;t))
. . . | . . | | | . . . | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . . | | 28. ∀l@0:IdLnk.
. . . | . . | | | . . . | |     (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . . . | |      (<k,l>
. . . | . . | | | . . . | |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . . | |     ⇒
. . . | . . | | | . . . | |     map(λms.2of(ms);filter(λms.
. . . | . . | | | . . . | |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . . | |     =
```

```
. . . | . . | | | . . . | |     if source(l@0) = i→
. . . | . . | | | . . . | |        tagged-list-messages(λx.
. . . | . . | | | . . . | |        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . . | |     else nil fi
. . . | . . | | | . . . | | 29. withlnk(l;m(i;t))
. . . | . . | | | . . . | |     =
. . . | . . | | | . . . | |     concat(map(λtgf.map(λx.
. . . | . . | | | . . . | |     <1of(tgf),x>;2of(tgf)
. . . | . . | | | . . . | |                  ((λx.s(i;t).x)
. . . | . . | | | . . . | |                  ,val(a(i;t))));f))
. . . | . . | | | . . . | | 30. source(l) = i
. . . | . . | | | . . . | | 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . . . | | 32. valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . . . | | 33. a(j;t) = a(i;t)
. . . | . . | | | . . . | | ⊢  val(a(i;t)) = val(a(i;t))  by Auto
. . . | . . | | | . . . | |
. . . | . . | | | . . . | .\
. . . | . . | | | . . . | . 1. i : Id
. . . | . . | | | . . . | . 2. k : Knd
. . . | . . | | | . . . | . 3. l : IdLnk
. . . | . . | | | . . . | . 4. ds : x:Id fp-> Type
. . . | . . | | | . . . | . 5. da : a:Knd fp-> Type
. . . | . . | | | . . . | . 6. f :
. . . | . . | | | . . . | . |  (tg:Id
. . . | . . | | | . . . | . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . . | . |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . . | . 7. source(l) = i
. . . | . . | | | . . . | . 8. w : World
. . . | . . | | | . . . | . 9. p : FairFifo
. . . | . . | | | . . . | . 10. FairFifo
. . . | . . | | | . . . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . . | . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . . | . |                       (source(l@0)
. . . | . . | | | . . . | . |                       ,i)→
. . . | . . | | | . . . | . |                       <ds
. . . | . . | | | . . . | . |                       ,da
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,<[<k,l>],λx.f>
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,·>
. . . | . . | | | . . . | . |                    else  fi.da(rcv(l@0;
. . . | . . | | | . . . | . |                                    tg))
. . . | . . | | | . . . | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . . | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . . | . 14. ∀e:E.
. . . | . . | | | . . . | . |   loc(e) = i
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   (valtype(e) ⊆r ma-valtype
. . . | . . | | | . . . | . |    (da; kind(e)))
. . . | . . | | | . . . | . 15. ∀e:E.
. . . | . . | | | . . . | . |   isrcv(e)
. . . | . . | | | . . . | . |   ⇒
```

```
. . . | . . | | | . . . | . |   lnk(e) = l
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   (valtype(e) ⊆r ma-valtype
. . . | . . | | | . . . | . |    (da; kind(e)))
. . . | . . | | | . . . | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . . | . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . . | . 17. ∀a:Action(i).
. . . | . . | | | . . . | . |   ¬isnull(a)
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . . | . 18. ∀x:Id.
. . . | . . | | | . . . | . |   vartype(i;x) ⊆r if ...→
. . . | . . | | | . . . | . |                      2of(ds)(x)
. . . | . . | | | . . . | . |                   else Top fi
. . . | . . | | | . . . | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . . | . 20. j : Id
. . . | . . | | | . . . | . 21. t : ℕ
. . . | . . | | | . . . | . 22. ¬isnull(a(j;t))
. . . | . . | | | . . . | . 23. True
. . . | . . | | | . . . | . 24. j = i
. . . | . . | | | . . . | . 25. kind(<j,t>) = k
. . . | . . | | | . . . | . 26. ¬isnull(a(i;t))
. . . | . . | | | . . . | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . . | . 28. ∀l@0:IdLnk.
. . . | . . | | | . . . | . |   (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . . . | . |    (<k,l>
. . . | . . | | | . . . | . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . . . | . |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . . | . |   =
. . . | . . | | | . . . | . |   if source(l@0) = i→
. . . | . . | | | . . . | . |      tagged-list-messages(λx.
. . . | . . | | | . . . | . |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . . | . |   else nil fi
. . . | . . | | | . . . | . 29. withlnk(l;m(i;t))
. . . | . . | | | . . . | . |   =
. . . | . . | | | . . . | . |   concat(map(λtgf.map(λx.
. . . | . . | | | . . . | . |   <1of(tgf),x>;2of(tgf)
. . . | . . | | | . . . | . |                ((λx.s(i;t).x)
. . . | . . | | | . . . | . |                ,val(a(i;t))));f))
. . . | . . | | | . . . | . 30. source(l) = i
. . . | . . | | | . . . | . 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . . . | . 32. valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . . . | . 33. a(j;t) = a(i;t)
. . . | . . | | | . . . | . 34. z : {aaa:Action(i)| ¬isnull(aaa) }
. . . | . . | | | . . . | . 35. z = a(i;t)
. . . | . . | | | . . . | . ⊢  (val(z) = val(a(i;t))) ∈ Prop
. . . | . . | | | . . . | . | by Analyze -2
. . . | . . | | | . . . | . |
. . . | . . | | | . . . | . .....subterm .....
. . . | . . | | | . . . | . 1. i : Id
. . . | . . | | | . . . | . 2. k : Knd
. . . | . . | | | . . . | . 3. l : IdLnk
. . . | . . | | | . . . | . 4. ds : x:Id fp-> Type
```

```
. . . | . . | | | . . . | . 5. da : a:Knd fp-> Type
. . . | . . | | | . . . | . 6. f :

. . . | . . | | | . . . | . |  (tg:Id
. . . | . . | | | . . . | . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . . | . |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . . | . 7. source(l) = i
. . . | . . | | | . . . | . 8. w : World
. . . | . . | | | . . . | . 9. p : FairFifo
. . . | . . | | | . . . | . 10. FairFifo
. . . | . . | | | . . . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . . | . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . . | . |                       (source(l@0)
. . . | . . | | | . . . | . |                       ,i)→
. . . | . . | | | . . . | . |                       <ds
. . . | . . | | | . . . | . |                       ,da
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,<[<k,l>],λx.f>
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,·>
. . . | . . | | | . . . | . |                    else  fi.da(rcv(l@0;
. . . | . . | | | . . . | . |                                    tg))
. . . | . . | | | . . . | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . . | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . . | . 14. ∀e:E.
. . . | . . | | | . . . | . |   loc(e) = i
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   (valtype(e) ⊆r ma-valtype
. . . | . . | | | . . . | . |    (da; kind(e)))
. . . | . . | | | . . . | . 15. ∀e:E.
. . . | . . | | | . . . | . |   isrcv(e)
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   lnk(e) = l
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   (valtype(e) ⊆r ma-valtype
. . . | . . | | | . . . | . |    (da; kind(e)))
. . . | . . | | | . . . | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . . | . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . . | . 17. ∀a:Action(i).
. . . | . . | | | . . . | . |   ¬isnull(a)
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . . | . 18. ∀x:Id.
. . . | . . | | | . . . | . |   vartype(i;x) ⊆r if ...→
. . . | . . | | | . . . | . |                        2of(ds)(x)
. . . | . . | | | . . . | . |                     else Top fi
. . . | . . | | | . . . | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . . | . 20. j : Id
. . . | . . | | | . . . | . 21. t : ℕ
. . . | . . | | | . . . | . 22. ¬isnull(a(j;t))
. . . | . . | | | . . . | . 23. True
. . . | . . | | | . . . | . 24. j = i
. . . | . . | | | . . . | . 25. kind(<j,t>) = k
```

```
. . . | . . | | | . . . | . 26. ¬isnull(a(i;t))
. . . | . . | | | . . . | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . . | . 28. ∀l@0:IdLnk.
. . . | . . | | | . . . | . |   (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . . . | . |    (<k,l>
. . . | . . | | | . . . | . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . . . | . |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . . | . |   =
. . . | . . | | | . . . | . |   if source(l@0) = i→
. . . | . . | | | . . . | . |      tagged-list-messages(λx.
. . . | . . | | | . . . | . |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . . | . |   else nil fi
. . . | . . | | | . . . | . 29. withlnk(l;m(i;t))
. . . | . . | | | . . . | . |   =
. . . | . . | | | . . . | . |   concat(map(λtgf.map(λx.
. . . | . . | | | . . . | . |   <1of(tgf),x>;2of(tgf)
. . . | . . | | | . . . | . |                ((λx.s(i;t).x)
. . . | . . | | | . . . | . |                ,val(a(i;t))));f))
. . . | . . | | | . . . | . 30. source(l) = i
. . . | . . | | | . . . | . 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . . . | . 32. valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . . . | . 33. a(j;t) = a(i;t)
. . . | . . | | | . . . | . 34. z : Action(i)
. . . | . . | | | . . . | . 35. ¬isnull(z)
. . . | . . | | | . . . | . 36. z = a(i;t)
. . . | . . | | | . . . | . ⊢  val(z) ∈ valtype(i;a(i;t))
. . . | . . | | | . . . | . | by DoSubsume
. . . | . . | | | . . . | . |
. . . | . . | | | . . . | . .....subtype_rel .....
. . . | . . | | | . . . | . 1. i : Id
. . . | . . | | | . . . | . 2. k : Knd
. . . | . . | | | . . . | . 3. l : IdLnk
. . . | . . | | | . . . | . 4. ds : x:Id fp-> Type
. . . | . . | | | . . . | . 5. da : a:Knd fp-> Type
. . . | . . | | | . . . | . 6. f :
. . . | . . | | | . . . | . |  (tg:Id
. . . | . . | | | . . . | . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . . | . |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . . | . 7. source(l) = i
. . . | . . | | | . . . | . 8. w : World
. . . | . . | | | . . . | . 9. p : FairFifo
. . . | . . | | | . . . | . 10. FairFifo
. . . | . . | | | . . . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . . | . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . . | . |                      (source(l@0)
. . . | . . | | | . . . | . |                      ,i)→
. . . | . . | | | . . . | . |                      <ds
. . . | . . | | | . . . | . |                      ,da
. . . | . . | | | . . . | . |                      ,
. . . | . . | | | . . . | . |                      ,
. . . | . . | | | . . . | . |                      ,
. . . | . . | | | . . . | . |                      ,<[<k,l>],λx.f>
. . . | . . | | | . . . | . |                      ,
```

```
. . . | . . | | | . . . | . |                        ,
. . . | . . | | | . . . | . |                        ,·>
. . . | . . | | | . . . | . |                    else  fi.da(rcv(l@0;
. . . | . . | | | . . . | . |                                  tg))
. . . | . . | | | . . . | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . . | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . . | . 14. ∀e:E.
. . . | . . | | | . . . | . |   loc(e) = i
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   (valtype(e) ⊆r ma-valtype
. . . | . . | | | . . . | . |    (da; kind(e)))
. . . | . . | | | . . . | . 15. ∀e:E.
. . . | . . | | | . . . | . |   isrcv(e)
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   lnk(e) = l
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   (valtype(e) ⊆r ma-valtype
. . . | . . | | | . . . | . |    (da; kind(e)))
. . . | . . | | | . . . | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . . | . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . . | . 17. ∀a:Action(i).
. . . | . . | | | . . . | . |   ¬isnull(a)
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . . | . 18. ∀x:Id.
. . . | . . | | | . . . | . |   vartype(i;x) ⊆r if ...→
. . . | . . | | | . . . | . |                      2of(ds)(x)
. . . | . . | | | . . . | . |                   else Top fi
. . . | . . | | | . . . | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . . | . 20. j : Id
. . . | . . | | | . . . | . 21. t : ℕ
. . . | . . | | | . . . | . 22. ¬isnull(a(j;t))
. . . | . . | | | . . . | . 23. True
. . . | . . | | | . . . | . 24. j = i
. . . | . . | | | . . . | . 25. kind(<j,t>) = k
. . . | . . | | | . . . | . 26. ¬isnull(a(i;t))
. . . | . . | | | . . . | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . . | . 28. ∀l@0:IdLnk.
. . . | . . | | | . . . | . |   (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . . . | . |    (<k,l>
. . . | . . | | | . . . | . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . . . | . |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . . | . |   =
. . . | . . | | | . . . | . |   if source(l@0) = i→
. . . | . . | | | . . . | . |      tagged-list-messages(λx.
. . . | . . | | | . . . | . |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . . | . |   else nil fi
. . . | . . | | | . . . | . 29. withlnk(l;m(i;t))
. . . | . . | | | . . . | . |   =
. . . | . . | | | . . . | . |   concat(map(λtgf.map(λx.
. . . | . . | | | . . . | . |   <1of(tgf),x>;2of(tgf)
. . . | . . | | | . . . | . |                ((λx.s(i;t).x)
. . . | . . | | | . . . | . |                ,val(a(i;t))));f))
```

```
. . . | . . | | | . . . | . 30. source(l) = i
. . . | . . | | | . . . | . 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . . . | . 32. valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . . . | . 33. a(j;t) = a(i;t)
. . . | . . | | | . . . | . 34. z : Action(i)
. . . | . . | | | . . . | . 35. ¬isnull(z)
. . . | . . | | | . . . | . 36. z = a(i;t)
. . . | . . | | | . . . | . ⊢  valtype(i;z) ⊆r valtype(i;a(i;t))
. . . | . . | | | . . . | . | by HypSubst -1 0
. . . | . . | | | . . . | . |
. . . | . . | | | . . . | . 1. i : Id
. . . | . . | | | . . . | . 2. k : Knd
. . . | . . | | | . . . | . 3. l : IdLnk
. . . | . . | | | . . . | . 4. ds : x:Id fp-> Type
. . . | . . | | | . . . | . 5. da : a:Knd fp-> Type
. . . | . . | | | . . . | . 6. f :
. . . | . . | | | . . . | . |  (tg:Id
. . . | . . | | | . . . | . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . . | . |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . . | . 7. source(l) = i
. . . | . . | | | . . . | . 8. w : World
. . . | . . | | | . . . | . 9. p : FairFifo
. . . | . . | | | . . . | . 10. FairFifo
. . . | . . | | | . . . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . . | . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . . | . |                       (source(l@0)
. . . | . . | | | . . . | . |                       ,i)→
. . . | . . | | | . . . | . |                       <ds
. . . | . . | | | . . . | . |                       ,da
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,<[<k,l>],λx.f>
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,
. . . | . . | | | . . . | . |                       ,·>
. . . | . . | | | . . . | . |                   else  fi.da(rcv(l@0;
                                                          tg))
. . . | . . | | | . . . | . 12. (eqof(IdDeq)(i,i)) ∼ true2
. . . | . . | | | . . . | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . . | . 14. ∀e:E.
. . . | . . | | | . . . | . |   loc(e) = i
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   (valtype(e) ⊆r ma-valtype
. . . | . . | | | . . . | . |    (da; kind(e)))
. . . | . . | | | . . . | . 15. ∀e:E.
. . . | . . | | | . . . | . |   isrcv(e)
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   lnk(e) = l
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   (valtype(e) ⊆r ma-valtype
. . . | . . | | | . . . | . |    (da; kind(e)))
. . . | . . | | | . . . | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . . | . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . . | . 17. ∀a:Action(i).
```

```
. . . | . . | | | . . . | . |   ¬isnull(a)
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . . | . 18. ∀x:Id.
. . . | . . | | | . . . | . |   vartype(i;x) ⊆r if ...→
. . . | . . | | | . . . | . |                     2of(ds)(x)
. . . | . . | | | . . . | . |                   else Top fi
. . . | . . | | | . . . | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . . | . 20. j : Id
. . . | . . | | | . . . | . 21. t : ℕ
. . . | . . | | | . . . | . 22. ¬isnull(a(j;t))
. . . | . . | | | . . . | . 23. True
. . . | . . | | | . . . | . 24. j = i
. . . | . . | | | . . . | . 25. kind(<j,t>) = k
. . . | . . | | | . . . | . 26. ¬isnull(a(i;t))
. . . | . . | | | . . . | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . . | . 28. ∀l@0:IdLnk.
. . . | . . | | | . . . | . |   (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . . . | . |    (<k,l>
. . . | . . | | | . . . | . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . . | . |   ⇒
. . . | . . | | | . . . | . |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . . . | . |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . . | . |   =
. . . | . . | | | . . . | . |   if source(l@0) = i→
. . . | . . | | | . . . | . |      tagged-list-messages(λx.
. . . | . . | | | . . . | . |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . . | . |   else nil fi
. . . | . . | | | . . . | . 29. withlnk(l;m(i;t))
. . . | . . | | | . . . | . |   =
. . . | . . | | | . . . | . |   concat(map(λtgf.map(λx.
. . . | . . | | | . . . | . |   <1of(tgf),x>;2of(tgf)
. . . | . . | | | . . . | . |                ((λx.s(i;t).x)
. . . | . . | | | . . . | . |                ,val(a(i;t))));f))
. . . | . . | | | . . . | . 30. source(l) = i
. . . | . . | | | . . . | . 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . . . | . 32. valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . . . | . 33. a(j;t) = a(i;t)
. . . | . . | | | . . . | . 34. z : Action(i)
. . . | . . | | | . . . | . 35. ¬isnull(z)
. . . | . . | | | . . . | . 36. z = a(i;t)
. . . | . . | | | . . . | . 37. z1 : {aaa:Action(i)| ¬isnull(aaa) }
. . . | . . | | | . . . | . ⊢  ¬isnull(z1)  by Analyze 0
. . . | . . | | | . . . | . |
. . . | . . | | | . . . | . 1. i : Id
. . . | . . | | | . . . | . 2. k : Knd
. . . | . . | | | . . . | . 3. l : IdLnk
. . . | . . | | | . . . | . 4. ds : x:Id fp-> Type
. . . | . . | | | . . . | . 5. da : a:Knd fp-> Type
. . . | . . | | | . . . | . 6. f :
. . . | . . | | | . . . | .    (tg:Id
. . . | . . | | | . . . | .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . . | .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . . | . 7. source(l) = i
. . . | . . | | | . . . | . 8. w : World
```

. . . | . . | | | . . . | . 9. p : FairFifo
. . . | . . | | | . . . | . 10. FairFifo
. . . | . . | | | . . . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . . | . (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . . | . (source(l@0)
. . . | . . | | | . . . | . ,i)→
. . . | . . | | | . . . | . <ds
. . . | . . | | | . . . | . ,da
. . . | . . | | | . . . | . ,
. . . | . . | | | . . . | . ,
. . . | . . | | | . . . | . ,
. . . | . . | | | . . . | . ,<[<k,l>],λx.f>
. . . | . . | | | . . . | . ,
. . . | . . | | | . . . | . ,
. . . | . . | | | . . . | . ,·>
. . . | . . | | | . . . | . else fi.da(rcv(l@0;
. . . | . . | | | . . . | . tg))
. . . | . . | | | . . . | . 12. (eqof(IdDeq)(i,i)) ∼ true$_2$
. . . | . . | | | . . . | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . . | . 14. ∀e:E.
. . . | . . | | | . . . | . loc(e) = i
. . . | . . | | | . . . | . ⇒
. . . | . . | | | . . . | . (valtype(e) ⊆r ma-valtype
. . . | . . | | | . . . | . (da; kind(e)))
. . . | . . | | | . . . | . 15. ∀e:E.
. . . | . . | | | . . . | . isrcv(e)
. . . | . . | | | . . . | . ⇒
. . . | . . | | | . . . | . lnk(e) = l
. . . | . . | | | . . . | . ⇒
. . . | . . | | | . . . | . (valtype(e) ⊆r ma-valtype
. . . | . . | | | . . . | . (da; kind(e)))
. . . | . . | | | . . . | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . . | . ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . . | . 17. ∀a:Action(i).
. . . | . . | | | . . . | . ¬isnull(a)
. . . | . . | | | . . . | . ⇒
. . . | . . | | | . . . | . (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . . | . 18. ∀x:Id.
. . . | . . | | | . . . | . vartype(i;x) ⊆r if ...→
. . . | . . | | | . . . | . 2of(ds)(x)
. . . | . . | | | . . . | . else Top fi
. . . | . . | | | . . . | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . . | . 20. j : Id
. . . | . . | | | . . . | . 21. t : ℕ
. . . | . . | | | . . . | . 22. ¬isnull(a(j;t))
. . . | . . | | | . . . | . 23. True
. . . | . . | | | . . . | . 24. j = i
. . . | . . | | | . . . | . 25. kind(<j,t>) = k
. . . | . . | | | . . . | . 26. ¬isnull(a(i;t))
. . . | . . | | | . . . | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . . | . 28. ∀l@0:IdLnk.
. . . | . . | | | . . . | . (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . . . | . (<k,l>
. . . | . . | | | . . . | . ,<kind(a(i;t)),l@0>) ∨$_2$ false$_2$)
. . . | . . | | | . . . | . ⇒

```
. . . | . . | | | . . . | .     map(λms.2of(ms);filter(λms.
. . . | . . | | | . . . | .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . . | .     =
. . . | . . | | | . . . | .     if source(l@0) = i→
. . . | . . | | | . . . | .        tagged-list-messages(λx.
. . . | . . | | | . . . | .        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . . | .     else nil fi
. . . | . . | | | . . . | . 29. withlnk(l;m(i;t))
. . . | . . | | | . . . | .     =
. . . | . . | | | . . . | .     concat(map(λtgf.map(λx.
. . . | . . | | | . . . | .     <1of(tgf),x>;2of(tgf)
. . . | . . | | | . . . | .                  ((λx.s(i;t).x)
. . . | . . | | | . . . | .                  ,val(a(i;t))));f))
. . . | . . | | | . . . | . 30. source(l) = i
. . . | . . | | | . . . | . 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . . . | . 32. valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . . . | . 33. a(j;t) = a(i;t)
. . . | . . | | | . . . | . 34. z : Action(i)
. . . | . . | | | . . . | . 35. ¬isnull(z)
. . . | . . | | | . . . | . 36. z = a(i;t)
. . . | . . | | | . . . | . 37. z1 : {aaa:Action(i)| ¬isnull(aaa) }
. . . | . . | | | . . . | . 38. isnull(z1)
. . . | . . | | | . . . | . ⊢  False  by Analyze -2
. . . | . . | | | . . . |
. . . | . . | | | . . . .\
. . . | . . | | | . . . . 1. i : Id
. . . | . . | | | . . . . 2. k : Knd
. . . | . . | | | . . . . 3. l : IdLnk
. . . | . . | | | . . . . 4. ds : x:Id fp-> Type
. . . | . . | | | . . . . 5. da : a:Knd fp-> Type
. . . | . . | | | . . . . 6. f :
. . . | . . | | | . . . .    (tg:Id
. . . | . . | | | . . . .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . . .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . . . 7. source(l) = i
. . . | . . | | | . . . . 8. w : World
. . . | . . | | | . . . . 9. p : FairFifo
. . . | . . | | | . . . . 10. FairFifo
. . . | . . | | | . . . . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . . .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . . .                      (source(l@0)
. . . | . . | | | . . . .                      ,i)→
. . . | . . | | | . . . .                      <ds
. . . | . . | | | . . . .                      ,da
. . . | . . | | | . . . .                      ,
. . . | . . | | | . . . .                      ,
. . . | . . | | | . . . .                      ,
. . . | . . | | | . . . .                      ,<[<k,l>],λx.f>
. . . | . . | | | . . . .                      ,
. . . | . . | | | . . . .                      ,
. . . | . . | | | . . . .                      ,·>
. . . | . . | | | . . . .                   else  fi.da(rcv
. . . | . . | | | . . . .     (l@0; tg))
. . . | . . | | | . . . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
```

```
. . . | . . | | | . . . . 14. ∀e:E.
. . . | . . | | | . . . .     loc(e) = i
. . . | . . | | | . . . .     ⇒
. . . | . . | | | . . . .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . . . 15. ∀e:E.
. . . | . . | | | . . . .     isrcv(e)
. . . | . . | | | . . . .     ⇒
. . . | . . | | | . . . .     lnk(e) = l
. . . | . . | | | . . . .     ⇒
. . . | . . | | | . . . .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . . .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . . . 17. ∀a:Action(i).
. . . | . . | | | . . . .     ¬isnull(a)
. . . | . . | | | . . . .     ⇒
. . . | . . | | | . . . .     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . . . 18. ∀x:Id.
. . . | . . | | | . . . .     vartype(i;x) ⊆r if ...→
. . . | . . | | | . . . .                        2of(ds)(x)
. . . | . . | | | . . . .                     else Top fi
. . . | . . | | | . . . . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . . . 20. j : Id
. . . | . . | | | . . . . 21. t : ℕ
. . . | . . | | | . . . . 22. ¬isnull(a(j;t))
. . . | . . | | | . . . . 23. True
. . . | . . | | | . . . . 24. j = i
. . . | . . | | | . . . . 25. kind(<j,t>) = k
. . . | . . | | | . . . . 26. ¬isnull(a(i;t))
. . . | . . | | | . . . . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . . . 28. ∀l@0:IdLnk.
. . . | . . | | | . . . .     (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . . . .      (<k,l>
. . . | . . | | | . . . .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . . .     ⇒
. . . | . . | | | . . . .     map(λms.2of(ms);filter(λms.
. . . | . . | | | . . . .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . . .     =
. . . | . . | | | . . . .     if source(l@0) = i→
. . . | . . | | | . . . .        tagged-list-messages(λx.
. . . | . . | | | . . . .        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . . .     else nil fi
. . . | . . | | | . . . . 29. withlnk(l;m(i;t))
. . . | . . | | | . . . .     =
. . . | . . | | | . . . .     concat(map(λtgf.map(λx.
. . . | . . | | | . . . .     <1of(tgf),x>;2of(tgf)
. . . | . . | | | . . . .                  ((λx.s(i;t).x)
. . . | . . | | | . . . .                  ,val(a(i;t))));f))
. . . | . . | | | . . . . 30. source(l) = i
. . . | . . | | | . . . . 31. ma-valtype(da; k) ∈ Type
. . . | . . | | | . . . . 32. valtype(i;a(i;t)) ⊆r ma-valtype(da; k)
. . . | . . | | | . . . . 33. a(j;t) = a(i;t)
. . . | . . | | | . . . . 34. val(a(j;t)) = val(a(i;t))
. . . | . . | | | . . . . ⊢  val(a(j;t)) = val(a(i;t))
. . . | . . | | | . . . .   by HypSubst -1 0
. . . | . . | | | . . . .      THEN
```

```
. . . | . . | | | . . . .      Try (Fold `member` 0)
. . . | . . | | | . . . .      THEN
. . . | . . | | | . . . .      DoSubsume
. . . | . . | | |
. . . | . . | | .\
. . . | . . | | . .....subterm .....
. . . | . . | | . 1. i : Id
. . . | . . | | . 2. k : Knd
. . . | . . | | . 3. l : IdLnk
. . . | . . | | . 4. ds : x:Id fp-> Type
. . . | . . | | . 5. da : a:Knd fp-> Type
. . . | . . | | . 6. f :
. . . | . . | | .    (tg:Id
. . . | . . | | .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | . 7. source(l) = i
. . . | . . | | . 8. w : World
. . . | . . | | . 9. p : FairFifo
. . . | . . | | . 10. FairFifo
. . . | . . | | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | .                         <ds
. . . | . . | | .                         ,da
. . . | . . | | .                         ,
. . . | . . | | .                         ,
. . . | . . | | .                         ,
. . . | . . | | .                         ,<[<k,l>],λx.f>
. . . | . . | | .                         ,
. . . | . . | | .                         ,
. . . | . . | | .                         ,•>
. . . | . . | | .                       else  fi.da(rcv(l@0; tg))
. . . | . . | | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | . 14. ∀e:E.
. . . | . . | | .     loc(e) = i
. . . | . . | | .     ⇒
. . . | . . | | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | . 15. ∀e:E.
. . . | . . | | .     isrcv(e)
. . . | . . | | .     ⇒
. . . | . . | | .     lnk(e) = l
. . . | . . | | .     ⇒
. . . | . . | | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | . 17. ∀a:Action(i).
. . . | . . | | .     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | . 18. ∀x:Id.
. . . | . . | | .     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . | | .                         2of(ds)(x)
. . . | . . | | .                       else Top fi
. . . | . . | | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | . 20. j : Id
. . . | . . | | . 21. t : ℕ
. . . | . . | | . 22. ¬isnull(a(j;t))
```

```
. . . | . . | | . 23. True
. . . | . . | | . 24. j = i
. . . | . . | | . 25. kind(<j,t>) = k
. . . | . . | | . 26. ¬isnull(a(i;t))
. . . | . . | | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | . 28. ∀l@0:IdLnk.
. . . | . . | | .     (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | .      (<k,l>
. . . | . . | | .      ,<kind(a(i;t)),l@0>) ∨2 false2)
. . . | . . | | .     ⇒
. . . | . . | | .     map(λms.2of(ms);filter(λms.
. . . | . . | | .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | .     =
. . . | . . | | .     if source(l@0) = i→
. . . | . . | | .        tagged-list-messages(λx.
. . . | . . | | .        s(i;t).x;val(a(i;t));f)
. . . | . . | | .     else nil fi
. . . | . . | | . 29. withlnk(l;m(i;t))
. . . | . . | | .     =
. . . | . . | | .     concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | | .                             ,x>;2of(tgf)
. . . | . . | | .                                 ((λx.s(i;t).x)
. . . | . . | | .                                 ,val(a(i;t))));f))
. . . | . . | | . 30. source(l) = i
. . . | . . | | . ⊢ f = f
. . . | . . | | .   by Fold ‘member‘ 0 THEN DoSubsume
. . . | . . | | .      THEN
. . . | . . | | .      RepeatFor 5 UseSubtypeRelLemmas
. . . | . . | |
. . . | . . | |\
. . . | . . | | 1. i : Id
. . . | . . | | 2. k : Knd
. . . | . . | | 3. l : IdLnk
. . . | . . | | 4. ds : x:Id fp-> Type
. . . | . . | | 5. da : a:Knd fp-> Type
. . . | . . | | 6. f :
. . . | . . | | |  (tg:Id
. . . | . . | | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | 7. source(l) = i
. . . | . . | | 8. w : World
. . . | . . | | 9. p : FairFifo
. . . | . . | | 10. FairFifo
. . . | . . | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | |                       <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . | | |                   else  fi.da(rcv(l@0; tg))
. . . | . . | | 12. (eqof(IdDeq)(i,i)) ∼ true2
. . . | . . | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | 14. ∀e:E.
. . . | . . | | |   loc(e) = i
. . . | . . | | |   ⇒
. . . | . . | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | 15. ∀e:E.
. . . | . . | | |   isrcv(e)
```

```
. . . | . . | | |   ⇒
. . . | . . | | |   lnk(e) = l
. . . | . . | | |   ⇒
. . . | . . | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | 17. ∀a:Action(i).
. . . | . . | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | 18. ∀x:Id.
. . . | . . | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . | | |                     2of(ds)(x)
. . . | . . | | |                   else Top fi
. . . | . . | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | 20. j : Id
. . . | . . | | 21. t : ℕ
. . . | . . | | 22. ¬isnull(a(j;t))
. . . | . . | | 23. True
. . . | . . | | 24. j = i
. . . | . . | | 25. kind(<j,t>) = k
. . . | . . | | 26. ¬isnull(a(i;t))
. . . | . . | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | 28. ∀l@0:IdLnk.
. . . | . . | | |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | |    (<k,l>
. . . | . . | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | |   ⇒
. . . | . . | | |   map(λms.2of(ms);filter(λms.mlnk(ms) = l@0;m(i;t)))
. . . | . . | | |   =
. . . | . . | | |   if source(l@0) = i→
. . . | . . | | |      tagged-list-messages(λx.s(i;t).x;val(a(i;t));f)
. . . | . . | | |   else nil fi
. . . | . . | | 29. withlnk(l;m(i;t))
. . . | . . | | |   =
. . . | . . | | |   concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | | |                           ,x>;2of(tgf)
. . . | . . | | |                               ((λx.s(i;t).x)
. . . | . . | | |                               ,val(a(i;t))));f))
. . . | . . | | 30. source(l) = i
. . . | . . | | ⊢  filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . | | |  =
. . . | . . | | |  map(λx.<l,x>;tagged-list-messages(λx.
. . . | . . | | |  s(i;t).x;val(a(i;t));f))
. . . | . . | | | by Fold ‘tagged-list-messages‘ -2
. . . | . . | | |    THEN
. . . | . . | | |    RevHypSubst -2 0
. . . | . . | | |    THEN
. . . | . . | | |    Try (Complete Auto)
. . . | . . | | |
. . . | . . | | |\
. . . | . . | | | 1. i : Id
. . . | . . | | | 2. k : Knd
. . . | . . | | | 3. l : IdLnk
. . . | . . | | | 4. ds : x:Id fp-> Type
. . . | . . | | | 5. da : a:Knd fp-> Type
. . . | . . | | | 6. f :
```

```
. . . | . . | | | |  (tg:Id
. . . | . . | | | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | 7. source(l) = i
. . . | . . | | | 8. w : World
. . . | . . | | | 9. p : FairFifo
. . . | . . | | | 10. FairFifo
. . . | . . | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | | |                      <ds
. . . | . . | | | |                      ,da
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,<[<k,l>],λx.f>
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,·>
. . . | . . | | | |                  else  fi.da(rcv(l@0; tg))
. . . | . . | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | 14. ∀e:E.
. . . | . . | | | |   loc(e) = i
. . . | . . | | | |   ⇒
. . . | . . | | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | 15. ∀e:E.
. . . | . . | | | |   isrcv(e)
. . . | . . | | | |   ⇒
. . . | . . | | | |   lnk(e) = l
. . . | . . | | | |   ⇒
. . . | . . | | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | 17. ∀a:Action(i).
. . . | . . | | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | 18. ∀x:Id.
. . . | . . | | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . | | | |                      2of(ds)(x)
. . . | . . | | | |                  else Top fi
. . . | . . | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | 20. j : Id
. . . | . . | | | 21. t : ℕ
. . . | . . | | | 22. ¬isnull(a(j;t))
. . . | . . | | | 23. True
. . . | . . | | | 24. j = i
. . . | . . | | | 25. kind(<j,t>) = k
. . . | . . | | | 26. ¬isnull(a(i;t))
. . . | . . | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | 28. ∀l@0:IdLnk.
. . . | . . | | | |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | | |    (<k,l>
. . . | . . | | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | |   ⇒
. . . | . . | | | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | |   mlnk(ms) = l@0;m(i;t)))
```

```
. . . | . . | | | |   =
. . . | . . | | | |   if source(l@0) = i→
. . . | . . | | | |      tagged-list-messages(λx.
. . . | . . | | | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | |   else nil fi
. . . | . . | | | 29. withlnk(l;m(i;t))
. . . | . . | | | |   =
. . . | . . | | | |   tagged-list-messages(λx.s(i;t).x;val(a(i;t));f)
. . . | . . | | | 30. source(l) = i
. . . | . . | | | ⊢  filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . | | | |  =
. . . | . . | | | |  map(λx.<l,x>;withlnk(l;m(i;t)))
. . . | . . | | | | by Unfold ‘w-withlnk‘ 0 THEN Unfold ‘mapfilter‘ 0
. . . | . . | | | |
. . . | . . | | | 1. i : Id
. . . | . . | | | 2. k : Knd
. . . | . . | | | 3. l : IdLnk
. . . | . . | | | 4. ds : x:Id fp-> Type
. . . | . . | | | 5. da : a:Knd fp-> Type
. . . | . . | | | 6. f :
. . . | . . | | | |  (tg:Id
. . . | . . | | | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | 7. source(l) = i
. . . | . . | | | 8. w : World
. . . | . . | | | 9. p : FairFifo
. . . | . . | | | 10. FairFifo
. . . | . . | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | | |                      <ds
. . . | . . | | | |                      ,da
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,<[<k,l>],λx.f>
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,
. . . | . . | | | |                      ,.>
. . . | . . | | | |                  else  fi.da(rcv(l@0; tg))
. . . | . . | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | 14. ∀e:E.
. . . | . . | | | |   loc(e) = i
. . . | . . | | | |   ⇒
. . . | . . | | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | 15. ∀e:E.
. . . | . . | | | |   isrcv(e)
. . . | . . | | | |   ⇒
. . . | . . | | | |   lnk(e) = l
. . . | . . | | | |   ⇒
. . . | . . | | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | 17. ∀a:Action(i).
. . . | . . | | | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
```

```
. . . | . . | | | 18. ∀x:Id.
. . . | . . | | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . | | | |                     2of(ds)(x)
. . . | . . | | | |                  else Top fi
. . . | . . | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | 20. j : Id
. . . | . . | | | 21. t : ℕ
. . . | . . | | | 22. ¬isnull(a(j;t))
. . . | . . | | | 23. True
. . . | . . | | | 24. j = i
. . . | . . | | | 25. kind(<j,t>) = k
. . . | . . | | | 26. ¬isnull(a(i;t))
. . . | . . | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | 28. ∀l@0:IdLnk.
. . . | . . | | | |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | | |    (<k,l>
. . . | . . | | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | |   ⇒
. . . | . . | | | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | |   =
. . . | . . | | | |   if source(l@0) = i→
. . . | . . | | | |      tagged-list-messages(λx.
. . . | . . | | | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | |   else nil fi
. . . | . . | | | 29. withlnk(l;m(i;t))
. . . | . . | | | |   =
. . . | . . | | | |   tagged-list-messages(λx.s(i;t).x;val(a(i;t));f)
. . . | . . | | | 30. source(l) = i
. . . | . . | | | ⊢  filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . | | | |  =
. . . | . . | | | |  map(λx.<l,x>;map(λms.2of(ms);filter(λms.
. . . | . . | | | |  mlnk(ms) = l;m(i;t))))
. . . | . . | | | by RW0
. . . | . . | | | |    Thm* ∀as:Top List, f,g:Top.
. . . | . . | | | |         map(g;map(f;as)) ∼ map(g o f;as)
. . . | . . | | | |    {map_DASH_map}
. . . | . . | | | |    0
. . . | . . | | | |    THENA
. . . | . . | | | |    Try (Complete Auto)
. . . | . . | | | |
. . . | . . | | | |\
. . . | . . | | | | .....wf .....
. . . | . . | | | | 1. i : Id
. . . | . . | | | | 2. k : Knd
. . . | . . | | | | 3. l : IdLnk
. . . | . . | | | | 4. ds : x:Id fp-> Type
. . . | . . | | | | 5. da : a:Knd fp-> Type
. . . | . . | | | | 6. f :
. . . | . . | | | | |  (tg:Id
. . . | . . | | | | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | | 7. source(l) = i
. . . | . . | | | | 8. w : World
. . . | . . | | | | 9. p : FairFifo
```

```
. . . | . . | | | | 10. FairFifo
. . . | . . | | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | | |   (w.M
. . . | . . | | | | |    (l@0
. . . | . . | | | | |    ,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | | | |                 <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . . | . . | | | | |              else  fi.da(rcv(l@0; tg))
. . . | . . | | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | | 14. ∀e:E.
. . . | . . | | | | |   loc(e) = i
. . . | . . | | | | |   ⇒
. . . | . . | | | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | 15. ∀e:E.
. . . | . . | | | | |   isrcv(e)
. . . | . . | | | | |   ⇒
. . . | . . | | | | |   lnk(e) = l
. . . | . . | | | | |   ⇒
. . . | . . | | | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | | 17. ∀a:Action(i).
. . . | . . | | | | |   ¬isnull(a)
. . . | . . | | | | |   ⇒
. . . | . . | | | | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | | 18. ∀x:Id.
. . . | . . | | | | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | | |                                        ds))→
. . . | . . | | | | |                     2of(ds)(x)
. . . | . . | | | | |                   else Top fi
. . . | . . | | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | | 20. j : Id
. . . | . . | | | | 21. t : ℕ
. . . | . . | | | | 22. ¬isnull(a(j;t))
. . . | . . | | | | 23. True
. . . | . . | | | | 24. j = i
. . . | . . | | | | 25. kind(<j,t>) = k
. . . | . . | | | | 26. ¬isnull(a(i;t))
. . . | . . | | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | | 28. ∀l@0:IdLnk.
. . . | . . | | | | |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | | | |    (<k,l>
. . . | . . | | | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | | |   ⇒
. . . | . . | | | | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | | |   =
. . . | . . | | | | |   if source(l@0) = i→
. . . | . . | | | | |      tagged-list-messages(λx.
. . . | . . | | | | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | | |   else nil fi
. . . | . . | | | | 29. withlnk(l;m(i;t))
. . . | . . | | | | |   =
. . . | . . | | | | |   tagged-list-messages(λx.
. . . | . . | | | | |   s(i;t).x;val(a(i;t));f)
```

```
. . . | . . | | | | 30. source(l) = i
. . . | . . | | | | ⊢  filter(λms.mlnk(ms) = l;m(i;t)) ∈ Top List
. . . | . . | | | | | by DoSubsume
. . . | . . | | | | |
. . . | . . | | | | .....subtype_rel .....
. . . | . . | | | | 1. i : Id
. . . | . . | | | | 2. k : Knd
. . . | . . | | | | 3. l : IdLnk
. . . | . . | | | | 4. ds : x:Id fp-> Type
. . . | . . | | | | 5. da : a:Knd fp-> Type
. . . | . . | | | | 6. f :
. . . | . . | | | |    (tg:Id
. . . | . . | | | |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | |     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | | 7. source(l) = i
. . . | . . | | | | 8. w : World
. . . | . . | | | | 9. p : FairFifo
. . . | . . | | | | 10. FairFifo
. . . | . . | | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | |     (w.M
. . . | . . | | | |      (l@0
. . . | . . | | | |      ,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | | |                  <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . . | . . | | | |                else  fi.da(rcv(l@0; tg))
. . . | . . | | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | | 14. ∀e:E.
. . . | . . | | | |     loc(e) = i
. . . | . . | | | |     ⇒
. . . | . . | | | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | 15. ∀e:E.
. . . | . . | | | |     isrcv(e)
. . . | . . | | | |     ⇒
. . . | . . | | | |     lnk(e) = l
. . . | . . | | | |     ⇒
. . . | . . | | | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | | 17. ∀a:Action(i).
. . . | . . | | | |     ¬isnull(a)
. . . | . . | | | |     ⇒
. . . | . . | | | |     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | | 18. ∀x:Id.
. . . | . . | | | |     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | |                                          ds))→
. . . | . . | | | |                          2of(ds)(x)
. . . | . . | | | |                        else Top fi
. . . | . . | | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | | 20. j : Id
. . . | . . | | | | 21. t : ℕ
. . . | . . | | | | 22. ¬isnull(a(j;t))
. . . | . . | | | | 23. True
. . . | . . | | | | 24. j = i
. . . | . . | | | | 25. kind(<j,t>) = k
. . . | . . | | | | 26. ¬isnull(a(i;t))
```

```
. . . | . . | | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | | 28. ∀l@0:IdLnk.
. . . | . . | | | |     (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | | |      (<k,l>
. . . | . . | | | |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | |     ⇒
. . . | . . | | | |     map(λms.2of(ms);filter(λms.
. . . | . . | | | |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | |     =
. . . | . . | | | |     if source(l@0) = i→
. . . | . . | | | |        tagged-list-messages(λx.
. . . | . . | | | |        s(i;t).x;val(a(i;t));f)
. . . | . . | | | |     else nil fi
. . . | . . | | | | 29. withlnk(l;m(i;t))
. . . | . . | | | |     =
. . . | . . | | | |     tagged-list-messages(λx.
. . . | . . | | | |     s(i;t).x;val(a(i;t));f)
. . . | . . | | | | 30. source(l) = i
. . . | . . | | | | ⊢  ({m:Msg(w.M)
. . . | . . | | | |      | source(mlnk(m)) = i } List) ⊆r (Top List)
. . . | . . | | | |   by UseSubtypeRelLemmas THEN Analyze 0
. . . | . . | | | |

. . . | . . | | | .\
. . . | . . | | | . 1. i : Id
. . . | . . | | | . 2. k : Knd
. . . | . . | | | . 3. l : IdLnk
. . . | . . | | | . 4. ds : x:Id fp-> Type
. . . | . . | | | . 5. da : a:Knd fp-> Type
. . . | . . | | | . 6. f :
. . . | . . | | | . |  (tg:Id
. . . | . . | | | . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . 7. source(l) = i
. . . | . . | | | . 8. w : World
. . . | . . | | | . 9. p : FairFifo
. . . | . . | | | . 10. FairFifo
. . . | . . | | | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . |   (w.M
. . . | . . | | | . |    (l@0
. . . | . . | | | . |    ,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | | . |                <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . . | . . | | | . |             else  fi.da(rcv(l@0; tg))
. . . | . . | | | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . 14. ∀e:E.
. . . | . . | | | . |   loc(e) = i
. . . | . . | | | . |   ⇒
. . . | . . | | | . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . 15. ∀e:E.
. . . | . . | | | . |   isrcv(e)
. . . | . . | | | . |   ⇒
. . . | . . | | | . |   lnk(e) = l
. . . | . . | | | . |   ⇒
. . . | . . | | | . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
```

```
. . . | . . | | | . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . 17. ∀a:Action(i).
. . . | . . | | | . |   ¬isnull(a)
. . . | . . | | | . |   ⇒
. . . | . . | | | . |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . 18. ∀x:Id.
. . . | . . | | | . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | . |                                        ds))→
. . . | . . | | | . |                      2of(ds)(x)
. . . | . . | | | . |                  else Top fi
. . . | . . | | | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . 20. j : Id
. . . | . . | | | . 21. t : ℕ
. . . | . . | | | . 22. ¬isnull(a(j;t))
. . . | . . | | | . 23. True
. . . | . . | | | . 24. j = i
. . . | . . | | | . 25. kind(<j,t>) = k
. . . | . . | | | . 26. ¬isnull(a(i;t))
. . . | . . | | | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . 28. ∀l@0:IdLnk.
. . . | . . | | | . |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | | . |    (<k,l>
. . . | . . | | | . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . |   ⇒
. . . | . . | | | . |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . |   =
. . . | . . | | | . |   if source(l@0) = i→
. . . | . . | | | . |      tagged-list-messages(λx.
. . . | . . | | | . |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . |   else nil fi
. . . | . . | | | . 29. withlnk(l;m(i;t))
. . . | . . | | | . |   =
. . . | . . | | | . |   tagged-list-messages(λx.
. . . | . . | | | . |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . 30. source(l) = i
. . . | . . | | | . ⊢  filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . | | | . |  =
. . . | . . | | | . |  map((λx.<l,x>) o (λms.2of(ms));filter(λms.
. . . | . . | | | . |  mlnk(ms) = l;m(i;t)))
. . . | . . | | | . | by Assert
. . . | . . | | | . |    (map((λx.<l,x>) o (λms.2of(ms));filter(λms.
. . . | . . | | | . |     mlnk(ms) = l;m(i;t)))
. . . | . . | | | . |     =
. . . | . . | | | . |     filter(λms.mlnk(ms) = l;m(i;t)))
. . . | . . | | | . |
. . . | . . | | | . |\
. . . | . . | | | . | .....assertion .....
. . . | . . | | | . | 1. i : Id
. . . | . . | | | . | 2. k : Knd
. . . | . . | | | . | 3. l : IdLnk
. . . | . . | | | . | 4. ds : x:Id fp-> Type
. . . | . . | | | . | 5. da : a:Knd fp-> Type
. . . | . . | | | . | 6. f :
. . . | . . | | | . | |  (tg:Id
```

```
. . . | . . | | | . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | 7. source(l) = i
. . . | . . | | | . | 8. w : World
. . . | . . | | | . | 9. p : FairFifo
. . . | . . | | | . | 10. FairFifo
. . . | . . | | | . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | |                       (source(l@0)
. . . | . . | | | . | |                       ,i)→
. . . | . . | | | . | |                       <ds
. . . | . . | | | . | |                       ,da
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,<[<k,l>],λx.f>
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,·>
. . . | . . | | | . | |                    else  fi.da(rcv(l@0; tg))
. . . | . . | | | . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | 14. ∀e:E.
. . . | . . | | | . | |   loc(e) = i
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | 15. ∀e:E.
. . . | . . | | | . | |   isrcv(e)
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   lnk(e) = l
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | 17. ∀a:Action(i).
. . . | . . | | | . | |   ¬isnull(a)
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | 18. ∀x:Id.
. . . | . . | | | . | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | . | |                                        ds))→
. . . | . . | | | . | |                         2of(ds)(x)
. . . | . . | | | . | |                      else Top fi
. . . | . . | | | . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | 20. j : Id
. . . | . . | | | . | 21. t : ℕ
. . . | . . | | | . | 22. ¬isnull(a(j;t))
. . . | . . | | | . | 23. True
. . . | . . | | | . | 24. j = i
. . . | . . | | | . | 25. kind(<j,t>) = k
. . . | . . | | | . | 26. ¬isnull(a(i;t))
. . . | . . | | | . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | 28. ∀l@0:IdLnk.
. . . | . . | | | . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . | |    (<k,l>
```

```
. . . | . . | | | . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | |   =
. . . | . . | | | . | |   if source(l@0) = i→
. . . | . . | | | . | |      tagged-list-messages(λx.
. . . | . . | | | . | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | |   else nil fi
. . . | . . | | | . | 29. withlnk(l;m(i;t))
. . . | . . | | | . | |   =
. . . | . . | | | . | |   tagged-list-messages(λx.
. . . | . . | | | . | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | 30. source(l) = i
. . . | . . | | | . | ⊢  map((λx.<l,x>) o (λms.2of(ms));filter(λms.
. . . | . . | | | . | |  mlnk(ms) = l;m(i;t)))
. . . | . . | | | . | |  =
. . . | . . | | | . | |  filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . | | by Fold ‘mapfilter‘ 0
. . . | . . | | | . | |
. . . | . . | | | . | 1. i : Id
. . . | . . | | | . | 2. k : Knd
. . . | . . | | | . | 3. l : IdLnk
. . . | . . | | | . | 4. ds : x:Id fp-> Type
. . . | . . | | | . | 5. da : a:Knd fp-> Type
. . . | . . | | | . | 6. f :
. . . | . . | | | . | |  (tg:Id
. . . | . . | | | . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | 7. source(l) = i
. . . | . . | | | . | 8. w : World
. . . | . . | | | . | 9. p : FairFifo
. . . | . . | | | . | 10. FairFifo
. . . | . . | | | . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | |                       (source(l@0)
. . . | . . | | | . | |                       ,i)→
. . . | . . | | | . | |                       <ds
. . . | . . | | | . | |                       ,da
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,<[<k,l>],λx.f>
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,
. . . | . . | | | . | |                       ,·>
. . . | . . | | | . | |                    else  fi.da(rcv(l@0; tg))
. . . | . . | | | . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | 14. ∀e:E.
. . . | . . | | | . | |   loc(e) = i
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | 15. ∀e:E.
. . . | . . | | | . | |   isrcv(e)
```

```
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   lnk(e) = l

. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | 17. ∀a:Action(i).
. . . | . . | | | . | |   ¬isnull(a)
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | 18. ∀x:Id.
. . . | . . | | | . | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | . | |                                    ds))→
. . . | . . | | | . | |                      2of(ds)(x)
. . . | . . | | | . | |                  else Top fi
. . . | . . | | | . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | 20. j : Id
. . . | . . | | | . | 21. t : ℕ
. . . | . . | | | . | 22. ¬isnull(a(j;t))
. . . | . . | | | . | 23. True
. . . | . . | | | . | 24. j = i
. . . | . . | | | . | 25. kind(<j,t>) = k
. . . | . . | | | . | 26. ¬isnull(a(i;t))
. . . | . . | | | . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | 28. ∀l@0:IdLnk.
. . . | . . | | | . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . | |    (<k,l>
. . . | . . | | | . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | |   ⇒
. . . | . . | | | . | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | |   =
. . . | . . | | | . | |   if source(l@0) = i→
. . . | . . | | | . | |      tagged-list-messages(λx.
. . . | . . | | | . | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | |   else nil fi
. . . | . . | | | . | 29. withlnk(l;m(i;t))
. . . | . . | | | . | |   =
. . . | . . | | | . | |   tagged-list-messages(λx.
. . . | . . | | | . | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | 30. source(l) = i
. . . | . . | | | . | ⊢  mapfilter((λx.<l,x>) o (λms.2of(ms));λms.
. . . | . . | | | . | |  mlnk(ms) = l;m(i;t))
. . . | . . | | | . | |  =
. . . | . . | | | . | |  filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . | | by SubsumeEqC
. . . | . . | | | . | |    ({m:Msg(w.M)| source(mlnk(m)) = i } List)
. . . | . . | | | .
. . . | . . | | | . |\
. . . | . . | | | . | ....assertion .....
. . . | . . | | | . | 1. i : Id
. . . | . . | | | . | 2. k : Knd
. . . | . . | | | . | 3. l : IdLnk
. . . | . . | | | . | 4. ds : x:Id fp-> Type
. . . | . . | | | . | 5. da : a:Knd fp-> Type
```

```
. . . | . . | | | . | | 6. f :
. . . | . . | | | . | |    (tg:Id
. . . | . . | | | . | |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | |     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | 7. source(l) = i
. . . | . . | | | . | | 8. w : World
. . . | . . | | | . | | 9. p : FairFifo
. . . | . . | | | . | | 10. FairFifo
. . . | . . | | | . | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | |                          (source(l@0)
. . . | . . | | | . | |                          ,i)→
. . . | . . | | | . | |                          <ds
. . . | . . | | | . | |                          ,da
. . . | . . | | | . | |                          ,
. . . | . . | | | . | |                          ,
. . . | . . | | | . | |                          ,
. . . | . . | | | . | |                          ,<[<k,l>],λx.f>
. . . | . . | | | . | |                          ,
. . . | . . | | | . | |                          ,
. . . | . . | | | . | |                          ,•>
. . . | . . | | | . | |                       else  fi.da(rcv(l@0; tg))
. . . | . . | | | . | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | 14. ∀e:E.
. . . | . . | | | . | |     loc(e) = i
. . . | . . | | | . | |     ⇒
. . . | . . | | | . | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | 15. ∀e:E.
. . . | . . | | | . | |     isrcv(e)
. . . | . . | | | . | |     ⇒
. . . | . . | | | . | |     lnk(e) = l
. . . | . . | | | . | |     ⇒
. . . | . . | | | . | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | 17. ∀a:Action(i).
. . . | . . | | | . | |     ¬isnull(a)
. . . | . . | | | . | |     ⇒
. . . | . . | | | . | |     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | 18. ∀x:Id.
. . . | . . | | | . | |     vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | . | |                                          ds))→
. . . | . . | | | . | |                        2of(ds)(x)
. . . | . . | | | . | |                     else Top fi
. . . | . . | | | . | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | 20. j : Id
. . . | . . | | | . | | 21. t : ℕ
. . . | . . | | | . | | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | 23. True
. . . | . . | | | . | | 24. j = i
. . . | . . | | | . | | 25. kind(<j,t>) = k
. . . | . . | | | . | | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | 28. ∀l@0:IdLnk.
```

```
. . . | . . | | | . | |     (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . | |      (<k,l>
. . . | . . | | | . | |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | |     ⇒
. . . | . . | | | . | |     map(λms.2of(ms);filter(λms.
. . . | . . | | | . | |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | |     =
. . . | . . | | | . | |     if source(l@0) = i→
. . . | . . | | | . | |        tagged-list-messages(λx.
. . . | . . | | | . | |        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | |     else nil fi
. . . | . . | | | . | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | |     =
. . . | . . | | | . | |     tagged-list-messages(λx.
. . . | . . | | | . | |     s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | 30. source(l) = i
. . . | . . | | | . | | ⊢  ({m:Msg(w.M)
. . . | . . | | | . | |      | source(mlnk(m)) = i } List) ⊆r (Msg
. . . | . . | | | . | |    ((λl,tg. da(rcv(l; tg))?Top)) List)
. . . | . . | | | . | |   by UseSubtypeRelLemmas
. . . | . . | | | . | |      THEN
. . . | . . | | | . | |      AllHyps
. . . | . . | | | . | |      (λh.Unfold ‘es-Msg‘ h THEN Reduce h)
. . . | . . | | | . | |
. . . | . . | | | . | |\
. . . | . . | | | . | | .....assertion .....
. . . | . . | | | . | | 1. i : Id
. . . | . . | | | . | | 2. k : Knd
. . . | . . | | | . | | 3. l : IdLnk
. . . | . . | | | . | | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | 5. da : a:Knd fp-> Type
. . . | . . | | | . | | 6. f :
. . . | . . | | | . | | |  (tg:Id
. . . | . . | | | . | | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | 7. source(l) = i
. . . | . . | | | . | | 8. w : World
. . . | . . | | | . | | 9. p : FairFifo
. . . | . . | | | . | | 10. FairFifo
. . . | . . | | | . | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | |                       (source(l@0)
. . . | . . | | | . | | |                       ,i)→
. . . | . . | | | . | | |                       <ds
. . . | . . | | | . | | |                       ,da
. . . | . . | | | . | | |                       ,
. . . | . . | | | . | | |                       ,
. . . | . . | | | . | | |                       ,
. . . | . . | | | . | | |                       ,<[<k,l>],λx.f>
. . . | . . | | | . | | |                       ,
. . . | . . | | | . | | |                       ,
. . . | . . | | | . | | |                       ,·>
. . . | . . | | | . | | |                    else  fi.da(rcv(l@0; tg))
. . . | . . | | | . | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
```

```
. . . | . . | | | . | | 14. ∀e:E.
. . . | . . | | | . | | |   loc(e) = i
. . . | . . | | | . | | |   ⇒
. . . | . . | | | . | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | 15. ∀e:E.
. . . | . . | | | . | | |   isrcv(e)
. . . | . . | | | . | | |   ⇒
. . . | . . | | | . | | |   lnk(e) = l
. . . | . . | | | . | | |   ⇒
. . . | . . | | | . | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | 17. ∀a:Action(i).
. . . | . . | | | . | | |   ¬isnull(a)
. . . | . . | | | . | | |   ⇒
. . . | . . | | | . | | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | 18. ∀x:Id.
. . . | . . | | | . | | |   vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | . | | |                                    ds))→
. . . | . . | | | . | | |                     2of(ds)(x)
. . . | . . | | | . | | |                   else Top fi
. . . | . . | | | . | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | 20. j : Id
. . . | . . | | | . | | 21. t : ℕ
. . . | . . | | | . | | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | 23. True
. . . | . . | | | . | | 24. j = i
. . . | . . | | | . | | 25. kind(<j,t>) = k
. . . | . . | | | . | | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . | | |    (<k,l>
. . . | . . | | | . | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | |   ⇒
. . . | . . | | | . | | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | |   =
. . . | . . | | | . | | |   if source(l@0) = i→
. . . | . . | | | . | | |      tagged-list-messages(λx.
. . . | . . | | | . | | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | |   else nil fi
. . . | . . | | | . | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | |   =
. . . | . . | | | . | | |   tagged-list-messages(λx.
. . . | . . | | | . | | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | 30. source(l) = i
. . . | . . | | | . | | 31. ({m:Msg(w.M)
. . . | . . | | | . | | |     | source(mlnk(m)) = i } List) ⊆r (Msg
. . . | . . | | | . | | |   ((λl,tg. da(rcv(l; tg))?Top)) List)
. . . | . . | | | . | | ⊢  mapfilter((λx.<l,x>) o (λms.2of(ms));λms.
. . . | . . | | | . | |    mlnk(ms) = l;m(i;t))
. . . | . . | | | . | |    =
. . . | . . | | | . | |    filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . | | by Unfold ‘mapfilter‘ 0
```

```
. . . | . . | | | . | | |
. . . | . . | | | . | | 1. i : Id
. . . | . . | | | . | | 2. k : Knd
. . . | . . | | | . | | 3. l : IdLnk
. . . | . . | | | . | | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | 5. da : a:Knd fp-> Type
. . . | . . | | | . | | 6. f :
. . . | . . | | | . | | |  (tg:Id
. . . | . . | | | . | | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | 7. source(l) = i
. . . | . . | | | . | | 8. w : World
. . . | . . | | | . | | 9. p : FairFifo
. . . | . . | | | . | | 10. FairFifo
. . . | . . | | | . | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | |                       (source(l@0)
. . . | . . | | | . | | |                       ,i)→
. . . | . . | | | . | | |                       <ds
. . . | . . | | | . | | |                       ,da
. . . | . . | | | . | | |                       ,
. . . | . . | | | . | | |                       ,
. . . | . . | | | . | | |                       ,
. . . | . . | | | . | | |                       ,<[<k,l>],λx.f>
. . . | . . | | | . | | |                       ,
. . . | . . | | | . | | |                       ,
. . . | . . | | | . | | |                       ,·>
. . . | . . | | | . | | |                     else  fi.da(rcv(l@0; tg))
. . . | . . | | | . | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | 14. ∀e:E.
. . . | . . | | | . | | |   loc(e) = i
. . . | . . | | | . | | |   ⇒
. . . | . . | | | . | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | 15. ∀e:E.
. . . | . . | | | . | | |   isrcv(e)
. . . | . . | | | . | | |   ⇒
. . . | . . | | | . | | |   lnk(e) = l
. . . | . . | | | . | | |   ⇒
. . . | . . | | | . | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | 17. ∀a:Action(i).
. . . | . . | | | . | | |   ¬isnull(a)
. . . | . . | | | . | | |   ⇒
. . . | . . | | | . | | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | 18. ∀x:Id.
. . . | . . | | | . | | |   vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | . | | |                                        ds))→
. . . | . . | | | . | | |                       2of(ds)(x)
. . . | . . | | | . | | |                     else Top fi
. . . | . . | | | . | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | 20. j : Id
. . . | . . | | | . | | 21. t : ℕ
. . . | . . | | | . | | 22. ¬isnull(a(j;t))
```

```
. . . | . . | | | . | | 23. True
. . . | . . | | | . | | 24. j = i
. . . | . . | | | . | | 25. kind(<j,t>) = k
. . . | . . | | | . | | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . | | |    (<k,l>
. . . | . . | | | . | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | |   ⇒
. . . | . . | | | . | | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | |   =
. . . | . . | | | . | | |   if source(l@0) = i→
. . . | . . | | | . | | |      tagged-list-messages(λx.
. . . | . . | | | . | | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | |   else nil fi
. . . | . . | | | . | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | |   =
. . . | . . | | | . | | |   tagged-list-messages(λx.
. . . | . . | | | . | | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | 30. source(l) = i
. . . | . . | | | . | | 31. ({m:Msg(w.M)
. . . | . . | | | . | | |     | source(mlnk(m)) = i } List) ⊆r (Msg
. . . | . . | | | . | | |   ((λl,tg. da(rcv(l; tg))?Top)) List)
. . . | . . | | | . | | ⊢  map((λx.<l,x>) o (λms.2of(ms));filter(λms.
. . . | . . | | | . | |  |  mlnk(ms) = l;m(i;t)))
. . . | . . | | | . | |  |  =
. . . | . . | | | . | |  |  filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . | |  | by GenConcl
. . . | . . | | | . | |  |    (filter(λms.mlnk(ms) = l;m(i;t)) = L)
. . . | . . | | | . | |  |
. . . | . . | | | . | |  |\
. . . | . . | | | . | |  | .....wf .....
. . . | . . | | | . | |  | 1. i : Id
. . . | . . | | | . | |  | 2. k : Knd
. . . | . . | | | . | |  | 3. l : IdLnk
. . . | . . | | | . | |  | 4. ds : x:Id fp-> Type
. . . | . . | | | . | |  | 5. da : a:Knd fp-> Type
. . . | . . | | | . | |  | 6. f :
. . . | . . | | | . | |  |  | (tg:Id
. . . | . . | | | . | |  |  |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | |  |  |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | |  | 7. source(l) = i
. . . | . . | | | . | |  | 8. w : World
. . . | . . | | | . | |  | 9. p : FairFifo
. . . | . . | | | . | |  | 10. FairFifo
. . . | . . | | | . | |  | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | |  |  |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | |  |  |                       (source(l@0)
. . . | . . | | | . | |  |  |                       ,i)→
. . . | . . | | | . | |  |  |                       <ds
. . . | . . | | | . | |  |  |                       ,da
. . . | . . | | | . | |  |  |                       ,
. . . | . . | | | . | |  |  |                       ,
```

```
. . . | . . | | | . | | | |                        ,
. . . | . . | | | . | | | |                        ,<[<k,l>],λx.f>
. . . | . . | | | . | | | |                        ,
. . . | . . | | | . | | | |                        ,
. . . | . . | | | . | | | |                        ,·>
. . . | . . | | | . | | | |                    else  fi.da(rcv
. . . | . . | | | . | | | |    (l@0; tg))
. . . | . . | | | . | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | | 14. ∀e:E.
. . . | . . | | | . | | | |   loc(e) = i
. . . | . . | | | . | | | |   ⇒
. . . | . . | | | . | | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | | 15. ∀e:E.
. . . | . . | | | . | | | |   isrcv(e)
. . . | . . | | | . | | | |   ⇒
. . . | . . | | | . | | | |   lnk(e) = l
. . . | . . | | | . | | | |   ⇒
. . . | . . | | | . | | | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | | 17. ∀a:Action(i).
. . . | . . | | | . | | | |   ¬isnull(a)
. . . | . . | | | . | | | |   ⇒
. . . | . . | | | . | | | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | | 18. ∀x:Id.
. . . | . . | | | . | | | |   vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | | |                      2of(ds)(x)
. . . | . . | | | . | | | |                   else Top fi
. . . | . . | | | . | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | | 20. j : Id
. . . | . . | | | . | | | 21. t : ℕ
. . . | . . | | | . | | | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | | 23. True
. . . | . . | | | . | | | 24. j = i
. . . | . . | | | . | | | 25. kind(<j,t>) = k
. . . | . . | | | . | | | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | | |   (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . | | | |    (<k,l>
. . . | . . | | | . | | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | | |   ⇒
. . . | . . | | | . | | | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | | |   =
. . . | . . | | | . | | | |   if source(l@0) = i→
. . . | . . | | | . | | | |      tagged-list-messages(λx.
. . . | . . | | | . | | | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | | |   else nil fi
. . . | . . | | | . | | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | | |   =
. . . | . . | | | . | | | |   tagged-list-messages(λx.
. . . | . . | | | . | | | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | | 30. source(l) = i
```

```
. . . | . . | | | . | | | 31. ({m:Msg(w.M)
. . . | . . | | | . | | | |     | source(mlnk(m)) = i } List) ⊆r (Msg
. . . | . . | | | . | | | |   ((λl,tg. da(rcv(l; tg))?Top)) List)
. . . | . . | | | . | | | ⊢  filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . | | | |  ∈ {x:Msg| (λms.mlnk(ms) = l)(x) } List
. . . | . . | | | . | | | by BackThru
. . . | . . | | | . | | | |    Thm* ∀T:Type, P:(T→𝔹), l:T List.
. . . | . . | | | . | | | |         filter(P;l) ∈ {x:T| P(x) } List
. . . | . . | | | . | | | |    {filter_type}
. . . | . . | | | . | | |
. . . | . . | | | . | | |\
. . . | . . | | | . | | | .....subterm .....
. . . | . . | | | . | | | 1. i : Id
. . . | . . | | | . | | | 2. k : Knd
. . . | . . | | | . | | | 3. l : IdLnk
. . . | . . | | | . | | | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | | 5. da : a:Knd fp-> Type

. . . | . . | | | . | | | 6. f :
. . . | . . | | | . | | | |    (tg:Id
. . . | . . | | | . | | | |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | | |     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | | 7. source(l) = i
. . . | . . | | | . | | | 8. w : World
. . . | . . | | | . | | | 9. p : FairFifo
. . . | . . | | | . | | | 10. FairFifo
. . . | . . | | | . | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | | |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | | |                          (source(l@0)
. . . | . . | | | . | | | |                          ,i)→
. . . | . . | | | . | | | |                          <ds
. . . | . . | | | . | | | |                          ,da
. . . | . . | | | . | | | |                          ,
. . . | . . | | | . | | | |                          ,
. . . | . . | | | . | | | |                          ,
. . . | . . | | | . | | | |                          ,<[<k,l>],λx.f>
. . . | . . | | | . | | | |                          ,
. . . | . . | | | . | | | |                          ,
. . . | . . | | | . | | | |                          ,·>
. . . | . . | | | . | | | |                       else  fi.da(rcv(l@0;
. . . | . . | | | . | | | |                                       tg))
. . . | . . | | | . | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | | 14. ∀e:E.
. . . | . . | | | . | | | |     loc(e) = i
. . . | . . | | | . | | | |     ⇒
. . . | . . | | | . | | | |     (valtype(e) ⊆r ma-valtype
. . . | . . | | | . | | | |      (da; kind(e)))
. . . | . . | | | . | | | 15. ∀e:E.
. . . | . . | | | . | | | |     isrcv(e)
. . . | . . | | | . | | | |     ⇒
. . . | . . | | | . | | | |     lnk(e) = l
. . . | . . | | | . | | | |     ⇒
. . . | . . | | | . | | | |     (valtype(e) ⊆r ma-valtype
. . . | . . | | | . | | | |      (da; kind(e)))
. . . | . . | | | . | | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
```

```
. . . | . . | | | . | | | |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | | | 17. ∀a:Action(i).
. . . | . . | | | . | | | |     ¬isnull(a)
. . . | . . | | | . | | | |     ⇒
. . . | . . | | | . | | | |     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | | | 18. ∀x:Id.
. . . | . . | | | . | | | |     vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | | |                       2of(ds)(x)
. . . | . . | | | . | | | |                     else Top fi
. . . | . . | | | . | | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | | | 20. j : Id
. . . | . . | | | . | | | | 21. t : ℕ
. . . | . . | | | . | | | | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | | | 23. True
. . . | . . | | | . | | | | 24. j = i
. . . | . . | | | . | | | | 25. kind(<j,t>) = k
. . . | . . | | | . | | | | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | | | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | | |     (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . | | | |      (<k,l>
. . . | . . | | | . | | | |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | | |     ⇒
. . . | . . | | | . | | | |     map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | | |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | | |     =
. . . | . . | | | . | | | |     if source(l@0) = i→
. . . | . . | | | . | | | |        tagged-list-messages(λx.
. . . | . . | | | . | | | |        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | | |     else nil fi
. . . | . . | | | . | | | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | | |     =
. . . | . . | | | . | | | |     tagged-list-messages(λx.
. . . | . . | | | . | | | |     s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | | | 30. source(l) = i
. . . | . . | | | . | | | | 31. ({m:Msg(w.M)
. . . | . . | | | . | | | |       | source(mlnk(m)) = i } List) ⊆r ...
. . . | . . | | | . | | | | 32. ms : Msg
. . . | . . | | | . | | | | ⊢  mlnk(ms) ∈ IdLnk
. . . | . . | | | . | | | |   by Analyze -1 THEN Unfold ‘mlnk‘ 0
. . . | . . | | | . | | | |      THEN
. . . | . . | | | . | | | |      Reduce 0
. . . | . . | | | . | | | |
. . . | . . | | | . | | | .\
. . . | . . | | | . | | | . .....unproved Inclusion subgoal .....
. . . | . . | | | . | | | . 1. i : Id
. . . | . . | | | . | | | . 2. k : Knd
. . . | . . | | | . | | | . 3. l : IdLnk
. . . | . . | | | . | | | . 4. ds : x:Id fp-> Type
. . . | . . | | | . | | | . 5. da : a:Knd fp-> Type
. . . | . . | | | . | | | . 6. f :
. . . | . . | | | . | | | .    (tg:Id
. . . | . . | | | . | | | .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | | .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | | . 7. source(l) = i
```

```
. . . | . . | | | . | | | . 8. w : World
. . . | . . | | | . | | | . 9. p : FairFifo
. . . | . . | | | . | | | . 10. FairFifo
. . . | . . | | | . | | | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | | .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | | .                         (source(l@0)
. . . | . . | | | . | | | .                         ,i)→
. . . | . . | | | . | | | .                         <ds
. . . | . . | | | . | | | .                         ,da
. . . | . . | | | . | | | .                         ,
. . . | . . | | | . | | | .                         ,
. . . | . . | | | . | | | .                         ,
. . . | . . | | | . | | | .                         ,<[<k,l>],λx.f>
. . . | . . | | | . | | | .                         ,
. . . | . . | | | . | | | .                         ,
. . . | . . | | | . | | | .                         ,.>
. . . | . . | | | . | | | .                       else  fi.da(rcv(l@0;
. . . | . . | | | . | | | .                                       tg))
. . . | . . | | | . | | | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | | . 14. ∀e:E.
. . . | . . | | | . | | | .     loc(e) = i
. . . | . . | | | . | | | .     ⇒
. . . | . . | | | . | | | .     (valtype(e) ⊆r ma-valtype
. . . | . . | | | . | | | .      (da; kind(e)))
. . . | . . | | | . | | | . 15. ∀e:E.
. . . | . . | | | . | | | .     isrcv(e)
. . . | . . | | | . | | | .     ⇒
. . . | . . | | | . | | | .     lnk(e) = l
. . . | . . | | | . | | | .     ⇒
. . . | . . | | | . | | | .     (valtype(e) ⊆r ma-valtype
. . . | . . | | | . | | | .      (da; kind(e)))
. . . | . . | | | . | | | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | | .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | | . 17. ∀a:Action(i).
. . . | . . | | | . | | | .     ¬isnull(a)
. . . | . . | | | . | | | .     ⇒
. . . | . . | | | . | | | .     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | | . 18. ∀x:Id.
. . . | . . | | | . | | | .     vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | | .                          2of(ds)(x)
. . . | . . | | | . | | | .                       else Top fi
. . . | . . | | | . | | | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | | . 20. j : Id
. . . | . . | | | . | | | . 21. t : ℕ
. . . | . . | | | . | | | . 22. ¬isnull(a(j;t))
. . . | . . | | | . | | | . 23. True
. . . | . . | | | . | | | . 24. j = i
. . . | . . | | | . | | | . 25. kind(<j,t>) = k
. . . | . . | | | . | | | . 26. ¬isnull(a(i;t))
. . . | . . | | | . | | | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | | . 28. ∀l@0:IdLnk.
. . . | . . | | | . | | | .     (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . | | | .      (<k,l>
. . . | . . | | | . | | | .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
```

```
. . . | . . | | | . | | | .     ⇒
. . . | . . | | | . | | | .     map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | | .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | | .     =
. . . | . . | | | . | | | .     if source(l@0) = i→
. . . | . . | | | . | | | .        tagged-list-messages(λx.
. . . | . . | | | . | | | .        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | | .     else nil fi
. . . | . . | | | . | | | . 29. withlnk(l;m(i;t))
. . . | . . | | | . | | | .     =
. . . | . . | | | . | | | .     tagged-list-messages(λx.
. . . | . . | | | . | | | .     s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | | . 30. source(l) = i
. . . | . . | | | . | | | . 31. ({m:Msg(w.M)
. . . | . . | | | . | | | .       | source(mlnk(m)) = i } List) ⊆r ...
. . . | . . | | | . | | | . 32. m(i;t) = m(i;t)
. . . | . . | | | . | | | . 33. {m:Msg(w.M)
. . . | . . | | | . | | | .      | source(mlnk(m)) = i } List
. . . | . . | | | . | | | . 34. {m:Msg(w.M)
. . . | . . | | | . | | | .      | source(mlnk(m)) = i } List
. . . | . . | | | . | | | . 35. u : {m:Msg(w.M)| source(mlnk(m)) = i }
. . . | . . | | | . | | | . ⊢  u ∈ Msg
. . . | . . | | | . | | | .   by Analyze -1 THEN Unfold ‘w-Msg‘ 0
. . . | . . | | | . | | |
. . . | . . | | | . | | |\
. . . | . . | | | . | | | .....wf .....
. . . | . . | | | . | | | 1. i : Id
. . . | . . | | | . | | | 2. k : Knd
. . . | . . | | | . | | | 3. l : IdLnk
. . . | . . | | | . | | | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | | 5. da : a:Knd fp-> Type
. . . | . . | | | . | | | 6. f :
. . . | . . | | | . | | |    (tg:Id
. . . | . . | | | . | | |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | |     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | | 7. source(l) = i
. . . | . . | | | . | | | 8. w : World
. . . | . . | | | . | | | 9. p : FairFifo
. . . | . . | | | . | | | 10. FairFifo
. . . | . . | | | . | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | |                          (source(l@0)
. . . | . . | | | . | | |                          ,i)→
. . . | . . | | | . | | |                          <ds
. . . | . . | | | . | | |                          ,da
. . . | . . | | | . | | |                          ,
. . . | . . | | | . | | |                          ,
. . . | . . | | | . | | |                          ,
. . . | . . | | | . | | |                          ,<[<k,l>],λx.f>
. . . | . . | | | . | | |                          ,
. . . | . . | | | . | | |                          ,
. . . | . . | | | . | | |                          ,·>

. . . | . . | | | . | | |                       else  fi.da(rcv
. . . | . . | | | . | | |     (l@0; tg))
. . . | . . | | | . | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
```

```
. . . | . . | | | . | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | | 14. ∀e:E.
. . . | . . | | | . | | |     loc(e) = i
. . . | . . | | | . | | |     ⇒
. . . | . . | | | . | | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | | 15. ∀e:E.
. . . | . . | | | . | | |     isrcv(e)
. . . | . . | | | . | | |     ⇒
. . . | . . | | | . | | |     lnk(e) = l
. . . | . . | | | . | | |     ⇒
. . . | . . | | | . | | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | | 17. ∀a:Action(i).
. . . | . . | | | . | | |     ¬isnull(a)
. . . | . . | | | . | | |     ⇒
. . . | . . | | | . | | |     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | | 18. ∀x:Id.
. . . | . . | | | . | | |     vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | |                         2of(ds)(x)
. . . | . . | | | . | | |                       else Top fi
. . . | . . | | | . | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | | 20. j : Id
. . . | . . | | | . | | | 21. t : ℕ
. . . | . . | | | . | | | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | | 23. True
. . . | . . | | | . | | | 24. j = i
. . . | . . | | | . | | | 25. kind(<j,t>) = k
. . . | . . | | | . | | | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | |     (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . | | |      (<k,l>
. . . | . . | | | . | | |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | |     ⇒
. . . | . . | | | . | | |     map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | |     =
. . . | . . | | | . | | |     if source(l@0) = i→
. . . | . . | | | . | | |        tagged-list-messages(λx.
. . . | . . | | | . | | |        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | |     else nil fi
. . . | . . | | | . | | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | |     =
. . . | . . | | | . | | |     tagged-list-messages(λx.
. . . | . . | | | . | | |     s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | | 30. source(l) = i
. . . | . . | | | . | | | 31. ({m:Msg(w.M)
. . . | . . | | | . | | |       | source(mlnk(m)) = i } List) ⊆r (Msg
. . . | . . | | | . | | |     ((λl,tg. da(rcv(l; tg))?Top)) List)
. . . | . . | | | . | | | ⊢  {x:Msg| (λms.mlnk(ms) = l)(x) } List
. . . | . . | | | . | | |    ∈ Type
. . . | . . | | | . | | |   by Reduce 0 THEN Analyze -1
. . . | . . | | | . | | |      THEN
. . . | . . | | | . | | |      Unfold ‘mlnk‘ 0
```

```
. . . | . . | | | . | | |      THEN
. . . | . . | | | . | | |      Reduce 0
. . . | . . | | | . | | |
. . . | . . | | | . | | .\
. . . | . . | | | . | | . 1. i : Id
. . . | . . | | | . | | . 2. k : Knd
. . . | . . | | | . | | . 3. l : IdLnk
. . . | . . | | | . | | . 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . 6. f :
. . . | . . | | | . | | . |  (tg:Id
. . . | . . | | | . | | . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | . |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . 7. source(l) = i
. . . | . . | | | . | | . 8. w : World
. . . | . . | | | . | | . 9. p : FairFifo
. . . | . . | | | . | | . 10. FairFifo
. . . | . . | | | . | | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | . |                       (source(l@0)
. . . | . . | | | . | | . |                       ,i)→
. . . | . . | | | . | | . |                       <ds
. . . | . . | | | . | | . |                       ,da
. . . | . . | | | . | | . |                       ,
. . . | . . | | | . | | . |                       ,
. . . | . . | | | . | | . |                       ,
. . . | . . | | | . | | . |                       ,<[<k,l>],λx.f>
. . . | . . | | | . | | . |                       ,
. . . | . . | | | . | | . |                       ,
. . . | . . | | | . | | . |                       ,·>
. . . | . . | | | . | | . |                    else  fi.da(rcv
. . . | . . | | | . | | . |   (l@0; tg))
. . . | . . | | | . | | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . 14. ∀e:E.
. . . | . . | | | . | | . |   loc(e) = i
. . . | . . | | | . | | . |   ⇒
. . . | . . | | | . | | . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | . 15. ∀e:E.
. . . | . . | | | . | | . |   isrcv(e)
. . . | . . | | | . | | . |   ⇒
. . . | . . | | | . | | . |   lnk(e) = l

. . . | . . | | | . | | . |   ⇒
. . . | . . | | | . | | . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . 17. ∀a:Action(i).
. . . | . . | | | . | | . |   ¬isnull(a)
. . . | . . | | | . | | . |   ⇒
. . . | . . | | | . | | . |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . 18. ∀x:Id.
. . . | . . | | | . | | . |   vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | . |                       2of(ds)(x)
. . . | . . | | | . | | . |                    else Top fi
. . . | . . | | | . | | . 19. IdDeq ∈ EqDecider(Id)
```

```
. . . | . . | | | . | | . 20. j : Id
. . . | . . | | | . | | . 21. t : ℕ
. . . | . . | | | . | | . 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . 23. True
. . . | . . | | | . | | . 24. j = i
. . . | . . | | | . | | . 25. kind(<j,t>) = k
. . . | . . | | | . | | . 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . |    (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . | | . |     (<k,l>
. . . | . . | | | . | | . |     ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | . |    ⇒
. . . | . . | | | . | | . |    map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . |    mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . |    =
. . . | . . | | | . | | . |    if source(l@0) = i→
. . . | . . | | | . | | . |       tagged-list-messages(λx.
. . . | . . | | | . | | . |       s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . |    else nil fi
. . . | . . | | | . | | . 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . |    =
. . . | . . | | | . | | . |    tagged-list-messages(λx.
. . . | . . | | | . | | . |    s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . 30. source(l) = i
. . . | . . | | | . | | . 31. ({m:Msg(w.M)
. . . | . . | | | . | | . |      | source(mlnk(m)) = i } List) ⊆r (Msg
. . . | . . | | | . | | . |    ((λl,tg. da(rcv(l; tg))?Top)) List)
. . . | . . | | | . | | . 32. L : {x:Msg| (λms.mlnk(ms) = l)(x) } List
. . . | . . | | | . | | . 33. filter(λms.mlnk(ms) = l;m(i;t)) = L
. . . | . . | | | . | | . ⊢  map((λx.<l,x>) o (λms.2of(ms));L) = L
. . . | . . | | | . | | . | by All Reduce
. . . | . . | | | . | | . |    THEN
. . . | . . | | | . | | . |    Inst
. . . | . . | | | . | | . |    Thm* ∀a:T List, f:(T→T).
. . . | . . | | | . | | . |         (∀x:T. (x ∈ a) ⇒ f(x) = x)
. . . | . . | | | . | | . |         ⇒
. . . | . . | | | . | | . |         map(f;a) = a
. . . | . . | | | . | | . |    {trivial_map}
. . . | . . | | | . | | . |    [{x:Msg| mlnk(x) = l }
. . . | . . | | | . | | . |    ;L
. . . | . . | | | . | | . |    ;(λx.<l,x>) o (λms.2of(ms))]
. . . | . . | | | . | | . |    THEN
. . . | . . | | | . | | . |    Try
. . . | . . | | | . | | . |    (Analyze -1 THEN Unfold `mlnk` 0
. . . | . . | | | . | | . |     THEN
. . . | . . | | | . | | . |     Reduce 0
. . . | . . | | | . | | . |     THEN
. . . | . . | | | . | | . |     Complete Auto)
. . . | . . | | | . | | . |
. . . | . . | | | . | | . |\
. . . | . . | | | . | | . | .....wf .....
. . . | . . | | | . | | . | 1. i : Id
. . . | . . | | | . | | . | 2. k : Knd
. . . | . . | | | . | | . | 3. l : IdLnk
```

```
. . . | . . | | | . | | . | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . | 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . | 6. f :
. . . | . . | | | . | | . | |  (tg:Id
. . . | . . | | | . | | . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . | 7. source(l) = i
. . . | . . | | | . | | . | 8. w : World
. . . | . . | | | . | | . | 9. p : FairFifo
. . . | . . | | | . | | . | 10. FairFifo
. . . | . . | | | . | | . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | . | |                       (source(l@0)
. . . | . . | | | . | | . | |                       ,i)→
. . . | . . | | | . | | . | |                       <ds
. . . | . . | | | . | | . | |                       ,da
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,<[<k,l>],λx.f>
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,·>
. . . | . . | | | . | | . | |                    else  fi.da(rcv(l@0;
. . . | . . | | | . | | . | |                                    tg))
. . . | . . | | | . | | . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . | 14. ∀e:E.
. . . | . . | | | . | | . | |   loc(e) = i
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   (valtype(e) ⊆r ma-valtype
. . . | . . | | | . | | . | |    (da; kind(e)))
. . . | . . | | | . | | . | 15. ∀e:E.
. . . | . . | | | . | | . | |   isrcv(e)
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   lnk(e) = l
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   (valtype(e) ⊆r ma-valtype
. . . | . . | | | . | | . | |    (da; kind(e)))
. . . | . . | | | . | | . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . | 17. ∀a:Action(i).
. . . | . . | | | . | | . | |   ¬isnull(a)
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . | 18. ∀x:Id.
. . . | . . | | | . | | . | |   vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | . | |                        2of(ds)(x)
. . . | . . | | | . | | . | |                     else Top fi
. . . | . . | | | . | | . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . | 20. j : Id
. . . | . . | | | . | | . | 21. t : ℕ
. . . | . . | | | . | | . | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . | 23. True
. . . | . . | | | . | | . | 24. j = i
```

```
. . . | . . | | | . | | . | 25. kind(<j,t>) = k
. . . | . . | | | . | | . | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . | |   (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . | | . | |    (<k,l>
. . . | . . | | | . | | . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . | |   =
. . . | . . | | | . | | . | |   if source(l@0) = i→
. . . | . . | | | . | | . | |      tagged-list-messages(λx.
. . . | . . | | | . | | . | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | |   else nil fi
. . . | . . | | | . | | . | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . | |   =
. . . | . . | | | . | | . | |   tagged-list-messages(λx.
. . . | . . | | | . | | . | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | 30. source(l) = i
. . . | . . | | | . | | . | 31. ({m:Msg(w.M)
. . . | . . | | | . | | . | |     | source(mlnk(m)) = i } List) ⊆r ...
. . . | . . | | | . | | . | 32. L : {x:Msg| mlnk(x) = l } List
. . . | . . | | | . | | . | 33. filter(λms.mlnk(ms) = l;m(i;t)) = L
. . . | . . | | | . | | . | ⊢  (λx.<l,x>) o (λms.2of(ms))
. . . | . . | | | . | | . | |  ∈ {x:Msg| mlnk(x) = l }→
. . . | . . | | | . | | . | |    {x:Msg| mlnk(x) = l }
. . . | . . | | | . | | . | | by Unfold ‘compose‘ 0 THEN Reduce 0
. . . | . . | | | . | | . | |
. . . | . . | | | . | | . | |\
. . . | . . | | | . | | . | | .....subterm .....
. . . | . . | | | . | | . | | 1. i : Id
. . . | . . | | | . | | . | | 2. k : Knd
. . . | . . | | | . | | . | | 3. l : IdLnk
. . . | . . | | | . | | . | | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . | | 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . | | 6. f :
. . . | . . | | | . | | . | | |  (tg:Id
. . . | . . | | | . | | . | | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | . | | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . | | 7. source(l) = i
. . . | . . | | | . | | . | | 8. w : World
. . . | . . | | | . | | . | | 9. p : FairFifo
. . . | . . | | | . | | . | | 10. FairFifo
. . . | . . | | | . | | . | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | . | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | . | | |                      (source(l@0)
. . . | . . | | | . | | . | | |                      ,i)→
. . . | . . | | | . | | . | | |                      <ds
. . . | . . | | | . | | . | | |                      ,da
. . . | . . | | | . | | . | | |                      ,
. . . | . . | | | . | | . | | |                      ,
. . . | . . | | | . | | . | | |                      ,
. . . | . . | | | . | | . | | |                      ,<[<k,l>],λx.f>
. . . | . . | | | . | | . | | |                      ,
```

```
. . . | . . | | | . | | . | | |                        ,
. . . | . . | | | . | | . | | |                        ,·>
. . . | . . | | | . | | . | | |                    else  fi.da(rcv
. . . | . . | | | . | | . | | |   (l@0; tg))
. . . | . . | | | . | | . | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . | | | 14. ∀e:E.
. . . | . . | | | . | | . | | |   loc(e) = i
. . . | . . | | | . | | . | | |   ⇒
. . . | . . | | | . | | . | | |   (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | | |                              kind(e)))
. . . | . . | | | . | | . | | | 15. ∀e:E.
. . . | . . | | | . | | . | | |   isrcv(e)
. . . | . . | | | . | | . | | |   ⇒
. . . | . . | | | . | | . | | |   lnk(e) = l
. . . | . . | | | . | | . | | |   ⇒
. . . | . . | | | . | | . | | |   (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | | |                              kind(e)))
. . . | . . | | | . | | . | | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . | | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . | | | 17. ∀a:Action(i).
. . . | . . | | | . | | . | | |   ¬isnull(a)

. . . | . . | | | . | | . | | |   ⇒
. . . | . . | | | . | | . | | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . | | | 18. ∀x:Id.
. . . | . . | | | . | | . | | |   vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | . | | |                        2of(ds)(x)
. . . | . . | | | . | | . | | |                    else Top fi
. . . | . . | | | . | | . | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . | | | 20. j : Id
. . . | . . | | | . | | . | | | 21. t : ℕ
. . . | . . | | | . | | . | | | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . | | | 23. True
. . . | . . | | | . | | . | | | 24. j = i
. . . | . . | | | . | | . | | | 25. kind(<j,t>) = k
. . . | . . | | | . | | . | | | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . | | | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . | | |   (eqof(product-deq(Knd;...;...;...))
. . . | . . | | | . | | . | | |    (<k,l>
. . . | . . | | | . | | . | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | . | | |   ⇒
. . . | . . | | | . | | . | | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . | | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . | | |   =
. . . | . . | | | . | | . | | |   if source(l@0) = i→
. . . | . . | | | . | | . | | |      tagged-list-messages(λx.
. . . | . . | | | . | | . | | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | | |   else nil fi
. . . | . . | | | . | | . | | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . | | |   =
. . . | . . | | | . | | . | | |   tagged-list-messages(λx.
. . . | . . | | | . | | . | | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | | | 30. source(l) = i
. . . | . . | | | . | | . | | | 31. (... List) ⊆r (Msg
```

```
. . . | . . | | | . | | . | | |   ((λl,tg. da(rcv(l; tg))?Top)) List)
. . . | . . | | | . | | . | | | 32. L : {x:Msg| mlnk(x) = l } List
. . . | . . | | | . | | . | | | 33. filter(λms.mlnk(ms) = l;m(i;t)) = L
. . . | . . | | | . | | . | | | 34. x : {x:Msg| mlnk(x) = l }
. . . | . . | | | . | | . | | | ⊢  <l,2of(x)> ∈ {x:Msg| mlnk(x) = l }
. . . | . . | | | . | | . | | | | by Analyze -1 THEN Analyze -2
. . . | . . | | | . | | . | | | |    THEN
. . . | . . | | | . | | . | | | |    Analyze -2
. . . | . . | | | . | | . | | | |    THEN
. . . | . . | | | . | | . | | | |    All Reduce
. . . | . . | | | . | | . | | | |    THEN
. . . | . . | | | . | | . | | | |    Analyze
. . . | . . | | | . | | . | | | |
. . . | . . | | | . | | . | | | |\
. . . | . . | | | . | | . | | | | 1. i : Id
. . . | . . | | | . | | . | | | | 2. k : Knd
. . . | . . | | | . | | . | | | | 3. l : IdLnk
. . . | . . | | | . | | . | | | | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . | | | | 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . | | | | 6. f :
. . . | . . | | | . | | . | | | | |  (tg:Id
. . . | . . | | | . | | . | | | | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | . | | | | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . | | | | 7. source(l) = i
. . . | . . | | | . | | . | | | | 8. w : World
. . . | . . | | | . | | . | | | | 9. p : FairFifo
. . . | . . | | | . | | . | | | | 10. FairFifo
. . . | . . | | | . | | . | | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | . | | | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | . | | | | |                     (source(l@0)
. . . | . . | | | . | | . | | | | |                     ,i)→
. . . | . . | | | . | | . | | | | |                     <ds
. . . | . . | | | . | | . | | | | |                     ,da
. . . | . . | | | . | | . | | | | |                     ,
. . . | . . | | | . | | . | | | | |                     ,
. . . | . . | | | . | | . | | | | |                     ,
. . . | . . | | | . | | . | | | | |                     ,<[<k,l>]
. . . | . . | | | . | | . | | | | |                      ,λx.f>
. . . | . . | | | . | | . | | | | |                     ,
. . . | . . | | | . | | . | | | | |                     ,
. . . | . . | | | . | | . | | | | |                     ,·>
. . . | . . | | | . | | . | | | | |                   else  fi.da(rcv
. . . | . . | | | . | | . | | | | |   (l@0; tg))
. . . | . . | | | . | | . | | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . | | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . | | | | 14. ∀e:E.
. . . | . . | | | . | | . | | | | |   loc(e) = i
. . . | . . | | | . | | . | | | | |   ⇒
. . . | . . | | | . | | . | | | | |   (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | | | | |                             kind(e)
. . . | . . | | | . | | . | | | | |                             ))
. . . | . . | | | . | | . | | | | 15. ∀e:E.
. . . | . . | | | . | | . | | | | |   isrcv(e)
. . . | . . | | | . | | . | | | | |   ⇒
. . . | . . | | | . | | . | | | | |   lnk(e) = l
```

```
. . . | . . | | | . | | . | | | |   ⇒
. . . | . . | | | . | | . | | | |   (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | | | |                            kind(e)
. . . | . . | | | . | | . | | | |                            ))
. . . | . . | | | . | | . | | | 16. {m:Msg
. . . | . . | | | . | | . | | | |    | source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . | | | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . | | | 17. ∀a:Action(i).
. . . | . . | | | . | | . | | | |   ¬isnull(a)
. . . | . . | | | . | | . | | | |   ⇒
. . . | . . | | | . | | . | | | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . | | | 18. ∀x:Id.
. . . | . . | | | . | | . | | | |   vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | . | | | |                    2of(ds)(x)
. . . | . . | | | . | | . | | | |                  else Top fi
. . . | . . | | | . | | . | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . | | | 20. j : Id
. . . | . . | | | . | | . | | | 21. t : ℕ
. . . | . . | | | . | | . | | | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . | | | 23. True
. . . | . . | | | . | | . | | | 24. j = i
. . . | . . | | | . | | . | | | 25. kind(<j,t>) = k
. . . | . . | | | . | | . | | | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . | | | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . | | | |   (eqof(...)
. . . | . . | | | . | | . | | | |    (<k,l>
. . . | . . | | | . | | . | | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | . | | | |   ⇒
. . . | . . | | | . | | . | | | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . | | | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . | | | |   =
. . . | . . | | | . | | . | | | |   if source(l@0) = i→
. . . | . . | | | . | | . | | | |      tagged-list-messages(λx.
. . . | . . | | | . | | . | | | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | | | |   else nil fi
. . . | . . | | | . | | . | | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . | | | |   =
. . . | . . | | | . | | . | | | |   tagged-list-messages(λx.
. . . | . . | | | . | | . | | | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | | | 30. source(l) = i
. . . | . . | | | . | | . | | | 31. (... List) ⊆r (Msg
. . . | . . | | | . | | . | | | |   ((λl,tg. da(rcv(l; tg))?Top)
. . . | . . | | | . | | . | | | |    ) List)
. . . | . . | | | . | | . | | | 32. L : {x:Msg| mlnk(x) = l } List
. . . | . . | | | . | | . | | | 33. filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . | | . | | | |   =
. . . | . . | | | . | | . | | | |   L
. . . | . . | | | . | | . | | | 34. l1 : IdLnk
. . . | . . | | | . | | . | | | 35. t1 : Id
. . . | . . | | | . | | . | | | 36. x2 : w.M(l1,t1)
. . . | . . | | | . | | . | | | 37. mlnk(<l1,t1,x2>) = l
. . . | . . | | | . | | . | | | ⊢  <l,t1,x2> ∈ Msg
. . . | . . | | | . | | . | | | | by Fold ‘msg‘ 0
. . . | . . | | | . | | . | | | |    THEN
```

```
. . . | . . | | | . | | . | | | |    Unfold ‘w-Msg‘ 0
. . . | . . | | | . | | . | | | |
. . . | . . | | | . | | . | | | | .....subterm .....
. . . | . . | | | . | | . | | | | 1. i : Id
. . . | . . | | | . | | . | | | | 2. k : Knd
. . . | . . | | | . | | . | | | | 3. l : IdLnk
. . . | . . | | | . | | . | | | | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . | | | | 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . | | | | 6. f :
. . . | . . | | | . | | . | | | |  (tg:Id
. . . | . . | | | . | | . | | | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | . | | | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . | | | | 7. source(l) = i
. . . | . . | | | . | | . | | | | 8. w : World
. . . | . . | | | . | | . | | | | 9. p : FairFifo
. . . | . . | | | . | | . | | | | 10. FairFifo
. . . | . . | | | . | | . | | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | . | | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | . | | | |                       (source(l@0)
. . . | . . | | | . | | . | | | |                       ,i)→
. . . | . . | | | . | | . | | | |                       <ds
. . . | . . | | | . | | . | | | |                       ,da
. . . | . . | | | . | | . | | | |                       ,
. . . | . . | | | . | | . | | | |                       ,
. . . | . . | | | . | | . | | | |                       ,
. . . | . . | | | . | | . | | | |                       ,<[<k,l>]
. . . | . . | | | . | | . | | | |                        ,λx.f>
. . . | . . | | | . | | . | | | |                       ,
. . . | . . | | | . | | . | | | |                       ,
. . . | . . | | | . | | . | | | |                       ,·>
. . . | . . | | | . | | . | | | |                    else  fi.da(rcv
. . . | . . | | | . | | . | | | |   (l@0; tg))
. . . | . . | | | . | | . | | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . | | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . | | | | 14. ∀e:E.
. . . | . . | | | . | | . | | | |   loc(e) = i
. . . | . . | | | . | | . | | | |   ⇒
. . . | . . | | | . | | . | | | |   (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | | | |                              kind(e)
. . . | . . | | | . | | . | | | |                              ))
. . . | . . | | | . | | . | | | | 15. ∀e:E.
. . . | . . | | | . | | . | | | |   isrcv(e)
. . . | . . | | | . | | . | | | |   ⇒
. . . | . . | | | . | | . | | | |   lnk(e) = l
. . . | . . | | | . | | . | | | |   ⇒
. . . | . . | | | . | | . | | | |   (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | | | |                              kind(e)
. . . | . . | | | . | | . | | | |                              ))
. . . | . . | | | . | | . | | | | 16. {m:Msg
. . . | . . | | | . | | . | | | |    | source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . | | | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . | | | | 17. ∀a:Action(i).
. . . | . . | | | . | | . | | | |   ¬isnull(a)
. . . | . . | | | . | | . | | | |   ⇒
. . . | . . | | | . | | . | | | |   (valtype(i;a) ⊆r da(kind(a))?Top)
```

```
. . . | . . | | | . | | . | | | 18. ∀x:Id.
. . . | . . | | | . | | . | | | |   vartype(i;x) ⊆r if ...→

. . . | . . | | | . | | . | | | |                     2of(ds)(x)
. . . | . . | | | . | | . | | | |                   else Top fi
. . . | . . | | | . | | . | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . | | | 20. j : Id
. . . | . . | | | . | | . | | | 21. t : ℕ
. . . | . . | | | . | | . | | | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . | | | 23. True
. . . | . . | | | . | | . | | | 24. j = i
. . . | . . | | | . | | . | | | 25. kind(<j,t>) = k
. . . | . . | | | . | | . | | | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . | | | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . | | | |   (eqof(...)
. . . | . . | | | . | | . | | | |    (<k,l>
. . . | . . | | | . | | . | | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | . | | | |   ⇒
. . . | . . | | | . | | . | | | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . | | | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . | | | |   =
. . . | . . | | | . | | . | | | |   if source(l@0) = i→
. . . | . . | | | . | | . | | | |      tagged-list-messages(λx.
. . . | . . | | | . | | . | | | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | | | |   else nil fi
. . . | . . | | | . | | . | | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . | | | |   =
. . . | . . | | | . | | . | | | |   tagged-list-messages(λx.
. . . | . . | | | . | | . | | | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | | | 30. source(l) = i
. . . | . . | | | . | | . | | | 31. (... List) ⊆r (Msg
. . . | . . | | | . | | . | | | |   ((λl,tg. da(rcv(l; tg))?Top)
. . . | . . | | | . | | . | | | |    ) List)
. . . | . . | | | . | | . | | | 32. L : {x:Msg| mlnk(x) = l } List
. . . | . . | | | . | | . | | | 33. filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . | | . | | | |   =
. . . | . . | | | . | | . | | | |   L
. . . | . . | | | . | | . | | | 34. l1 : IdLnk
. . . | . . | | | . | | . | | | 35. t1 : Id
. . . | . . | | | . | | . | | | 36. x2 : w.M(l1,t1)
. . . | . . | | | . | | . | | | 37. mlnk(<l1,t1,x2>) = l
. . . | . . | | | . | | . | | | ⊢  x2 ∈ w.M(l,t1)
. . . | . . | | | . | | . | | | | by Unfold ‘mlnk‘ -1 THEN Reduce -1
. . . | . . | | | . | | . | | | |    THEN
. . . | . . | | | . | | . | | | |    RW assert_pushdownC -1
. . . | . . | | | . | | . | | | |
. . . | . . | | | . | | . | | | 1. i : Id
. . . | . . | | | . | | . | | | 2. k : Knd
. . . | . . | | | . | | . | | | 3. l : IdLnk
. . . | . . | | | . | | . | | | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . | | | 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . | | | 6. f :
. . . | . . | | | . | | . | | | |    (tg:Id
. . . | . . | | | . | | . | | | |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | . | | | |     (da(rcv(l; tg))?Void List)) List
```

```
. . . | . . | | | . | | . | | | 7. source(l) = i
. . . | . . | | | . | | . | | | 8. w : World
. . . | . . | | | . | | . | | | 9. p : FairFifo
. . . | . . | | | . | | . | | | 10. FairFifo
. . . | . . | | | . | | . | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | . | | |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | . | | |                          (source(l@0)
. . . | . . | | | . | | . | | |                          ,i)→
. . . | . . | | | . | | . | | |                          <ds
. . . | . . | | | . | | . | | |                          ,da
. . . | . . | | | . | | . | | |                          ,
. . . | . . | | | . | | . | | |                          ,
. . . | . . | | | . | | . | | |                          ,
. . . | . . | | | . | | . | | |                          ,<[<k,l>]
. . . | . . | | | . | | . | | |                           ,λx.f>
. . . | . . | | | . | | . | | |                          ,
. . . | . . | | | . | | . | | |                          ,
. . . | . . | | | . | | . | | |                          ,·>
. . . | . . | | | . | | . | | |                       else  fi.da(rcv
. . . | . . | | | . | | . | | |     (l@0; tg))
. . . | . . | | | . | | . | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . | | | 14. ∀e:E.
. . . | . . | | | . | | . | | |     loc(e) = i
. . . | . . | | | . | | . | | |     ⇒
. . . | . . | | | . | | . | | |     (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | | |                                kind(e)
. . . | . . | | | . | | . | | |                                ))
. . . | . . | | | . | | . | | | 15. ∀e:E.
. . . | . . | | | . | | . | | |     isrcv(e)
. . . | . . | | | . | | . | | |     ⇒
. . . | . . | | | . | | . | | |     lnk(e) = l
. . . | . . | | | . | | . | | |     ⇒
. . . | . . | | | . | | . | | |     (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | | |                                kind(e)
. . . | . . | | | . | | . | | |                                ))
. . . | . . | | | . | | . | | | 16. {m:Msg
. . . | . . | | | . | | . | | |      | source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . | | |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . | | | 17. ∀a:Action(i).
. . . | . . | | | . | | . | | |     ¬isnull(a)
. . . | . . | | | . | | . | | |     ⇒
. . . | . . | | | . | | . | | |     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . | | | 18. ∀x:Id.
. . . | . . | | | . | | . | | |     vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | . | | |                          2of(ds)(x)
. . . | . . | | | . | | . | | |                       else Top fi

. . . | . . | | | . | | . | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . | | | 20. j : Id
. . . | . . | | | . | | . | | | 21. t : ℕ
. . . | . . | | | . | | . | | | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . | | | 23. True
. . . | . . | | | . | | . | | | 24. j = i
. . . | . . | | | . | | . | | | 25. kind(<j,t>) = k
. . . | . . | | | . | | . | | | 26. ¬isnull(a(i;t))
```

```
. . . | . . | | | . | | . | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . | | | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . | | |     (eqof(...)
. . . | . . | | | . | | . | | |      (<k,l>
. . . | . . | | | . | | . | | |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | . | | |     ⇒
. . . | . . | | | . | | . | | |     map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . | | |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . | | |     =
. . . | . . | | | . | | . | | |     if source(l@0) = i→
. . . | . . | | | . | | . | | |        tagged-list-messages(λx.
. . . | . . | | | . | | . | | |        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | | |     else nil fi
. . . | . . | | | . | | . | | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . | | |     =
. . . | . . | | | . | | . | | |     tagged-list-messages(λx.
. . . | . . | | | . | | . | | |     s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | | | 30. source(l) = i
. . . | . . | | | . | | . | | | 31. (... List) ⊆r (Msg
. . . | . . | | | . | | . | | |     ((λl,tg. da(rcv(l; tg))?Top)
. . . | . . | | | . | | . | | |      ) List)
. . . | . . | | | . | | . | | | 32. L : {x:Msg| mlnk(x) = l } List
. . . | . . | | | . | | . | | | 33. filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . | | . | | |     =
. . . | . . | | | . | | . | | |     L
. . . | . . | | | . | | . | | | 34. l1 : IdLnk
. . . | . . | | | . | | . | | | 35. t1 : Id
. . . | . . | | | . | | . | | | 36. x2 : w.M(l1,t1)
. . . | . . | | | . | | . | | | 37. l1 = l
. . . | . . | | | . | | . | | | ⊢  x2 ∈ w.M(l,t1)
. . . | . . | | | . | | . | | |   by DoSubsume THEN HypSubst -1 0
. . . | . . | | | . | | . | | |
. . . | . . | | | . | | . | | |\
. . . | . . | | | . | | . | | | .....set predicate .....
. . . | . . | | | . | | . | | | 1. i : Id
. . . | . . | | | . | | . | | | 2. k : Knd
. . . | . . | | | . | | . | | | 3. l : IdLnk
. . . | . . | | | . | | . | | | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . | | | 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . | | | 6. f :
. . . | . . | | | . | | . | | |    (tg:Id
. . . | . . | | | . | | . | | |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | . | | |     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . | | | 7. source(l) = i
. . . | . . | | | . | | . | | | 8. w : World
. . . | . . | | | . | | . | | | 9. p : FairFifo
. . . | . . | | | . | | . | | | 10. FairFifo
. . . | . . | | | . | | . | | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | . | | |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | . | | |                          (source(l@0)
. . . | . . | | | . | | . | | |                          ,i)→
. . . | . . | | | . | | . | | |                          <ds
. . . | . . | | | . | | . | | |                          ,da
. . . | . . | | | . | | . | | |                          ,
. . . | . . | | | . | | . | | |                          ,
```

```
. . . | . . | | | . | | . | | |                          ,
. . . | . . | | | . | | . | | |                          ,<[<k,l>]
. . . | . . | | | . | | . | | |                           ,λx.f>
. . . | . . | | | . | | . | | |                          ,
. . . | . . | | | . | | . | | |                          ,
. . . | . . | | | . | | . | | |                          ,·>
. . . | . . | | | . | | . | | |                       else  fi.da(rcv
. . . | . . | | | . | | . | | |     (l@0; tg))
. . . | . . | | | . | | . | | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . | | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . | | | 14. ∀e:E.
. . . | . . | | | . | | . | | |     loc(e) = i
. . . | . . | | | . | | . | | |     ⇒
. . . | . . | | | . | | . | | |     (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | | |                                kind(e)
. . . | . . | | | . | | . | | |                                ))
. . . | . . | | | . | | . | | | 15. ∀e:E.
. . . | . . | | | . | | . | | |     isrcv(e)
. . . | . . | | | . | | . | | |     ⇒
. . . | . . | | | . | | . | | |     lnk(e) = l
. . . | . . | | | . | | . | | |     ⇒
. . . | . . | | | . | | . | | |     (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | | |                                kind(e)
. . . | . . | | | . | | . | | |                                ))
. . . | . . | | | . | | . | | | 16. {m:Msg
. . . | . . | | | . | | . | | |      | source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . | | |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . | | | 17. ∀a:Action(i).
. . . | . . | | | . | | . | | |     ¬isnull(a)
. . . | . . | | | . | | . | | |     ⇒
. . . | . . | | | . | | . | | |     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . | | | 18. ∀x:Id.
. . . | . . | | | . | | . | | |     vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | . | | |                          2of(ds)(x)
. . . | . . | | | . | | . | | |                       else Top fi
. . . | . . | | | . | | . | | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . | | | 20. j : Id
. . . | . . | | | . | | . | | | 21. t : ℕ
. . . | . . | | | . | | . | | | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . | | | 23. True
. . . | . . | | | . | | . | | | 24. j = i
. . . | . . | | | . | | . | | | 25. kind(<j,t>) = k
. . . | . . | | | . | | . | | | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . | | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . | | | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . | | |     (eqof(...)
. . . | . . | | | . | | . | | |      (<k,l>
. . . | . . | | | . | | . | | |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | . | | |     ⇒
. . . | . . | | | . | | . | | |     map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . | | |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . | | |     =
. . . | . . | | | . | | . | | |     if source(l@0) = i→
. . . | . . | | | . | | . | | |        tagged-list-messages(λx.
. . . | . . | | | . | | . | | |        s(i;t).x;val(a(i;t));f)
```

```
. . . | . . | | | . | | . | | |     else nil fi
. . . | . . | | | . | | . | | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . | | |     =
. . . | . . | | | . | | . | | |     tagged-list-messages(λx.
. . . | . . | | | . | | . | | |     s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | | | 30. source(l) = i
. . . | . . | | | . | | . | | | 31. (... List) ⊆r (Msg
. . . | . . | | | . | | . | | |     ((λl,tg. da(rcv(l; tg))?Top)
. . . | . . | | | . | | . | | |      ) List)
. . . | . . | | | . | | . | | | 32. L : {x:Msg| mlnk(x) = l } List
. . . | . . | | | . | | . | | | 33. filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . | | . | | |     =
. . . | . . | | | . | | . | | |     L
. . . | . . | | | . | | . | | | 34. l1 : IdLnk
. . . | . . | | | . | | . | | | 35. t1 : Id
. . . | . . | | | . | | . | | | 36. x2 : w.M(l1,t1)
. . . | . . | | | . | | . | | | 37. mlnk(<l1,t1,x2>) = l
. . . | . . | | | . | | . | | | ⊢  mlnk(<l,t1,x2>) = l
. . . | . . | | | . | | . | | |   by Unfold `mlnk` 0 THEN Reduce 0
. . . | . . | | | . | | . | | |      THEN
. . . | . . | | | . | | . | | |      RW assert_pushdownC 0
. . . | . . | | | . | | . | | |
. . . | . . | | | . | | . | | .\
. . . | . . | | | . | | . | | . .....wf .....
. . . | . . | | | . | | . | | . 1. i : Id
. . . | . . | | | . | | . | | . 2. k : Knd
. . . | . . | | | . | | . | | . 3. l : IdLnk
. . . | . . | | | . | | . | | . 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . | | . 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . | | . 6. f :
. . . | . . | | | . | | . | | .    (tg:Id
. . . | . . | | | . | | . | | .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | . | | .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . | | . 7. source(l) = i
. . . | . . | | | . | | . | | . 8. w : World
. . . | . . | | | . | | . | | . 9. p : FairFifo
. . . | . . | | | . | | . | | . 10. FairFifo
. . . | . . | | | . | | . | | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | . | | .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | . | | .                         (source(l@0)
. . . | . . | | | . | | . | | .                         ,i)→
. . . | . . | | | . | | . | | .                         <ds
. . . | . . | | | . | | . | | .                         ,da
. . . | . . | | | . | | . | | .                         ,
. . . | . . | | | . | | . | | .                         ,
. . . | . . | | | . | | . | | .                         ,
. . . | . . | | | . | | . | | .                         ,<[<k,l>]
. . . | . . | | | . | | . | | .                          ,λx.f>
. . . | . . | | | . | | . | | .                         ,
. . . | . . | | | . | | . | | .                         ,
. . . | . . | | | . | | . | | .                         ,·>
. . . | . . | | | . | | . | | .                       else  fi.da(rcv
. . . | . . | | | . | | . | | .     (l@0; tg))
. . . | . . | | | . | | . | | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . | | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
```

. . . | . . | | | . | | . | | . 14. ∀e:E.
. . . | . . | | | . | | . | | .   loc(e) = i
. . . | . . | | | . | | . | | .   ⇒
. . . | . . | | | . | | . | | .   (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | | .   kind(e)
. . . | . . | | | . | | . | | .   ))
. . . | . . | | | . | | . | | . 15. ∀e:E.
. . . | . . | | | . | | . | | .   isrcv(e)
. . . | . . | | | . | | . | | .   ⇒
. . . | . . | | | . | | . | | .   lnk(e) = l
. . . | . . | | | . | | . | | .   ⇒
. . . | . . | | | . | | . | | .   (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | | .   kind(e)
. . . | . . | | | . | | . | | .   ))
. . . | . . | | | . | | . | | . 16. {m:Msg
. . . | . . | | | . | | . | | . | source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . | | .   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . | | . 17. ∀a:Action(i).
. . . | . . | | | . | | . | | .   ¬isnull(a)
. . . | . . | | | . | | . | | .   ⇒
. . . | . . | | | . | | . | | .   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . | | . 18. ∀x:Id.
. . . | . . | | | . | | . | | .   vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | . | | .   2of(ds)(x)
. . . | . . | | | . | | . | | .   else Top fi
. . . | . . | | | . | | . | | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . | | . 20. j : Id
. . . | . . | | | . | | . | | . 21. t : ℕ
. . . | . . | | | . | | . | | . 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . | | . 23. True
. . . | . . | | | . | | . | | . 24. j = i
. . . | . . | | | . | | . | | . 25. kind(<j,t>) = k
. . . | . . | | | . | | . | | . 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . | | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . | | . 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . | | .   (eqof(...)
. . . | . . | | | . | | . | | .   (<k,l>
. . . | . . | | | . | | . | | .   ,<kind(a(i;t)),l@0>) ∨$_2$ false$_2$)
. . . | . . | | | . | | . | | .   ⇒
. . . | . . | | | . | | . | | .   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . | | .   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . | | .   =
. . . | . . | | | . | | . | | .   if source(l@0) = i→
. . . | . . | | | . | | . | | .   tagged-list-messages(λx.
. . . | . . | | | . | | . | | .   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | | .   else nil fi
. . . | . . | | | . | | . | | . 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . | | .   =
. . . | . . | | | . | | . | | .   tagged-list-messages(λx.
. . . | . . | | | . | | . | | .   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | | . 30. source(l) = i
. . . | . . | | | . | | . | | . 31. (... List) ⊆r (Msg
. . . | . . | | | . | | . | | .   ((λl,tg. da(rcv(l; tg))?Top)
. . . | . . | | | . | | . | | .   ) List)
. . . | . . | | | . | | . | | . 32. L : {x:Msg| mlnk(x) = l } List

```
. . . | . . | | | . | | . | | . 33. filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . | | . | | .     =
. . . | . . | | | . | | . | | .     L
. . . | . . | | | . | | . | | . 34. l1 : IdLnk
. . . | . . | | | . | | . | | . 35. t1 : Id
. . . | . . | | | . | | . | | . 36. x2 : w.M(l1,t1)
. . . | . . | | | . | | . | | . 37. mlnk(<l1,t1,x2>) = l
. . . | . . | | | . | | . | | . 38. x : Msg
. . . | . . | | | . | | . | | . ⊢  mlnk(x) = l ∈ Type
. . . | . . | | | . | | . | | .   by Analyze -1 THEN Unfold ‘mlnk‘ 0
. . . | . . | | | . | | . | |
. . . | . . | | | . | | . | .\
. . . | . . | | | . | | . | . .....subterm .....
. . . | . . | | | . | | . | . 1. i : Id
. . . | . . | | | . | | . | . 2. k : Knd
. . . | . . | | | . | | . | . 3. l : IdLnk
. . . | . . | | | . | | . | . 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . | . 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . | . 6. f :
. . . | . . | | | . | | . | .    (tg:Id
. . . | . . | | | . | | . | .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | . | .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . | . 7. source(l) = i
. . . | . . | | | . | | . | . 8. w : World
. . . | . . | | | . | | . | . 9. p : FairFifo
. . . | . . | | | . | | . | . 10. FairFifo
. . . | . . | | | . | | . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | . | .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | . | .                         (source(l@0)
. . . | . . | | | . | | . | .                         ,i)→
. . . | . . | | | . | | . | .                         <ds
. . . | . . | | | . | | . | .                         ,da
. . . | . . | | | . | | . | .                         ,
. . . | . . | | | . | | . | .                         ,
. . . | . . | | | . | | . | .                         ,
. . . | . . | | | . | | . | .                         ,<[<k,l>],λx.f>
. . . | . . | | | . | | . | .                         ,
. . . | . . | | | . | | . | .                         ,
. . . | . . | | | . | | . | .                         ,•>
. . . | . . | | | . | | . | .                    else  fi.da(rcv
. . . | . . | | | . | | . | .     (l@0; tg))
. . . | . . | | | . | | . | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . | . 14. ∀e:E.
. . . | . . | | | . | | . | .     loc(e) = i
. . . | . . | | | . | | . | .     ⇒
. . . | . . | | | . | | . | .     (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | .                                kind(e)))
. . . | . . | | | . | | . | . 15. ∀e:E.
. . . | . . | | | . | | . | .     isrcv(e)
. . . | . . | | | . | | . | .     ⇒
. . . | . . | | | . | | . | .     lnk(e) = l
. . . | . . | | | . | | . | .     ⇒
. . . | . . | | | . | | . | .     (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | .                                kind(e)))
```

```
. . . | . . | | | . | | . | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . | .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . | . 17. ∀a:Action(i).
. . . | . . | | | . | | . | .     ¬isnull(a)
. . . | . . | | | . | | . | .     ⇒
. . . | . . | | | . | | . | .     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . | . 18. ∀x:Id.
. . . | . . | | | . | | . | .     vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | . | .                        2of(ds)(x)
. . . | . . | | | . | | . | .                      else Top fi
. . . | . . | | | . | | . | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . | . 20. j : Id
. . . | . . | | | . | | . | . 21. t : ℕ
. . . | . . | | | . | | . | . 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . | . 23. True
. . . | . . | | | . | | . | . 24. j = i
. . . | . . | | | . | | . | . 25. kind(<j,t>) = k

. . . | . . | | | . | | . | . 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . | . 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . | .     (eqof(product-deq(Knd;...;...;...))
. . . | . . | | | . | | . | .      (<k,l>
. . . | . . | | | . | | . | .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | . | .     ⇒
. . . | . . | | | . | | . | .     map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . | .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . | .     =
. . . | . . | | | . | | . | .     if source(l@0) = i→
. . . | . . | | | . | | . | .        tagged-list-messages(λx.
. . . | . . | | | . | | . | .        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | .     else nil fi
. . . | . . | | | . | | . | . 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . | .     =
. . . | . . | | | . | | . | .     tagged-list-messages(λx.
. . . | . . | | | . | | . | .     s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | . 30. source(l) = i
. . . | . . | | | . | | . | . 31. (... List) ⊆r (Msg
. . . | . . | | | . | | . | .     ((λl,tg. da(rcv(l; tg))?Top)) List)
. . . | . . | | | . | | . | . 32. L : {x:Msg| mlnk(x) = l } List
. . . | . . | | | . | | . | . 33. filter(λms.mlnk(ms) = l;m(i;t)) = L
. . . | . . | | | . | | . | . 34. x : Msg
. . . | . . | | | . | | . | . ⊢ mlnk(x) ∈ IdLnk
. . . | . . | | | . | | . | .   by Analyze -1 THEN Unfold `mlnk` 0
. . . | . . | | | . | | . | .      THEN
. . . | . . | | | . | | . | .      Reduce 0
. . . | . . | | | . | | . |
. . . | . . | | | . | | . |\
. . . | . . | | | . | | . | 1. i : Id
. . . | . . | | | . | | . | 2. k : Knd
. . . | . . | | | . | | . | 3. l : IdLnk
. . . | . . | | | . | | . | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . | 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . | 6. f :
. . . | . . | | | . | | . | |  (tg:Id
. . . | . . | | | . | | . | |   ×State(ds)→ma-valtype(da; k)→
```

```
. . . | . . | | | . | | . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . | 7. source(l) = i
. . . | . . | | | . | | . | 8. w : World
. . . | . . | | | . | | . | 9. p : FairFifo
. . . | . . | | | . | | . | 10. FairFifo
. . . | . . | | | . | | . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | . | |                       (source(l@0)
. . . | . . | | | . | | . | |                       ,i)→
. . . | . . | | | . | | . | |                       <ds
. . . | . . | | | . | | . | |                       ,da
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,<[<k,l>],λx.f>
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,•>
. . . | . . | | | . | | . | |                   else  fi.da(rcv(l@0;
. . . | . . | | | . | | . | |                                   tg))
. . . | . . | | | . | | . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . | 14. ∀e:E.
. . . | . . | | | . | | . | |   loc(e) = i
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   (valtype(e) ⊆r ma-valtype
. . . | . . | | | . | | . | |    (da; kind(e)))
. . . | . . | | | . | | . | 15. ∀e:E.
. . . | . . | | | . | | . | |   isrcv(e)
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   lnk(e) = l
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   (valtype(e) ⊆r ma-valtype
. . . | . . | | | . | | . | |    (da; kind(e)))
. . . | . . | | | . | | . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . | 17. ∀a:Action(i).
. . . | . . | | | . | | . | |   ¬isnull(a)
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . | 18. ∀x:Id.
. . . | . . | | | . | | . | |   vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | . | |                       2of(ds)(x)
. . . | . . | | | . | | . | |                   else Top fi
. . . | . . | | | . | | . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . | 20. j : Id
. . . | . . | | | . | | . | 21. t : ℕ
. . . | . . | | | . | | . | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . | 23. True
. . . | . . | | | . | | . | 24. j = i
. . . | . . | | | . | | . | 25. kind(<j,t>) = k
. . . | . . | | | . | | . | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . | |   (eqof(product-deq(Knd;IdLnk;...;...))
```

```
. . . | . . | | | . | | . | |    (<k,l>
. . . | . . | | | . | | . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . | |   =
. . . | . . | | | . | | . | |   if source(l@0) = i→

. . . | . . | | | . | | . | |      tagged-list-messages(λx.
. . . | . . | | | . | | . | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | |   else nil fi
. . . | . . | | | . | | . | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . | |   =
. . . | . . | | | . | | . | |   tagged-list-messages(λx.
. . . | . . | | | . | | . | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | 30. source(l) = i
. . . | . . | | | . | | . | 31. ({m:Msg(w.M)
. . . | . . | | | . | | . | |     | source(mlnk(m)) = i } List) ⊆r ...
. . . | . . | | | . | | . | 32. L : {x:Msg| mlnk(x) = l } List
. . . | . . | | | . | | . | 33. filter(λms.mlnk(ms) = l;m(i;t)) = L
. . . | . . | | | . | | . | 34. x : {x:Msg| mlnk(x) = l }
. . . | . . | | | . | | . | 35. (x ∈ L)
. . . | . . | | | . | | . | ⊢  ((λx.<l,x>) o (λms.2of(ms)))(x) = x
. . . | . . | | | . | | . | | by Reduce 0
. . . | . . | | | . | | . | |
. . . | . . | | | . | | . | 1. i : Id
. . . | . . | | | . | | . | 2. k : Knd
. . . | . . | | | . | | . | 3. l : IdLnk
. . . | . . | | | . | | . | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . | 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . | 6. f :
. . . | . . | | | . | | . | |  (tg:Id
. . . | . . | | | . | | . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . | 7. source(l) = i
. . . | . . | | | . | | . | 8. w : World
. . . | . . | | | . | | . | 9. p : FairFifo
. . . | . . | | | . | | . | 10. FairFifo
. . . | . . | | | . | | . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | . | |                       (source(l@0)
. . . | . . | | | . | | . | |                       ,i)→
. . . | . . | | | . | | . | |                       <ds
. . . | . . | | | . | | . | |                       ,da
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,<[<k,l>],λx.f>
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,
. . . | . . | | | . | | . | |                       ,·>
. . . | . . | | | . | | . | |                    else  fi.da(rcv(l@0;
. . . | . . | | | . | | . | |                                  tg))
. . . | . . | | | . | | . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . | 14. ∀e:E.
```

```
. . . | . . | | | . | | . | |   loc(e) = i
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   (valtype(e) ⊆r ma-valtype
. . . | . . | | | . | | . | |    (da; kind(e)))
. . . | . . | | | . | | . | 15. ∀e:E.
. . . | . . | | | . | | . | |   isrcv(e)
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   lnk(e) = l
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   (valtype(e) ⊆r ma-valtype
. . . | . . | | | . | | . | |    (da; kind(e)))
. . . | . . | | | . | | . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . | 17. ∀a:Action(i).
. . . | . . | | | . | | . | |   ¬isnull(a)
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . | 18. ∀x:Id.
. . . | . . | | | . | | . | |   vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | . | |                      2of(ds)(x)
. . . | . . | | | . | | . | |                    else Top fi
. . . | . . | | | . | | . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . | 20. j : Id
. . . | . . | | | . | | . | 21. t : ℕ
. . . | . . | | | . | | . | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . | 23. True
. . . | . . | | | . | | . | 24. j = i
. . . | . . | | | . | | . | 25. kind(<j,t>) = k
. . . | . . | | | . | | . | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . | |   (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . | | . | |    (<k,l>
. . . | . . | | | . | | . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | . | |   ⇒
. . . | . . | | | . | | . | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . | |   =
. . . | . . | | | . | | . | |   if source(l@0) = i→
. . . | . . | | | . | | . | |      tagged-list-messages(λx.
. . . | . . | | | . | | . | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | |   else nil fi
. . . | . . | | | . | | . | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . | |   =
. . . | . . | | | . | | . | |   tagged-list-messages(λx.
. . . | . . | | | . | | . | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | 30. source(l) = i
. . . | . . | | | . | | . | 31. ({m:Msg(w.M)
. . . | . . | | | . | | . | |     | source(mlnk(m)) = i } List) ⊆r ...
. . . | . . | | | . | | . | 32. L : {x:Msg| mlnk(x) = l } List
. . . | . . | | | . | | . | 33. filter(λms.mlnk(ms) = l;m(i;t)) = L
. . . | . . | | | . | | . | 34. x : {x:Msg| mlnk(x) = l }
. . . | . . | | | . | | . | 35. (x ∈ L)
. . . | . . | | | . | | . | ⊢  <l,2of(x)> = x
. . . | . . | | | . | | . | | by Analyze -2 THEN Analyze -3
```

```
. . . | . . | | | . | | . | |    THEN
. . . | . . | | | . | | . | |    Reduce 0
. . . | . . | | | . | | . | |    THEN
. . . | . . | | | . | | . | |    Analyze
. . . | . . | | | . | | . | |    THEN
. . . | . . | | | . | | . | |    Try
. . . | . . | | | . | | . | |    (Analyze -1 THEN Unfold `mlnk` 0
. . . | . . | | | . | | . | |     THEN
. . . | . . | | | . | | . | |     Reduce 0)
. . . | . . | | | . | | . | |
. . . | . . | | | . | | . | |\
. . . | . . | | | . | | . | | 1. i : Id
. . . | . . | | | . | | . | | 2. k : Knd
. . . | . . | | | . | | . | | 3. l : IdLnk
. . . | . . | | | . | | . | | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . | | 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . | | 6. f :
. . . | . . | | | . | | . | | |  (tg:Id
. . . | . . | | | . | | . | | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | . | | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . | | 7. source(l) = i
. . . | . . | | | . | | . | | 8. w : World
. . . | . . | | | . | | . | | 9. p : FairFifo
. . . | . . | | | . | | . | | 10. FairFifo
. . . | . . | | | . | | . | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | . | | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | . | | |                       (source(l@0)
. . . | . . | | | . | | . | | |                       ,i)→
. . . | . . | | | . | | . | | |                       <ds
. . . | . . | | | . | | . | | |                       ,da
. . . | . . | | | . | | . | | |                       ,
. . . | . . | | | . | | . | | |                       ,
. . . | . . | | | . | | . | | |                       ,
. . . | . . | | | . | | . | | |                       ,<[<k,l>],λx.f>
. . . | . . | | | . | | . | | |                       ,
. . . | . . | | | . | | . | | |                       ,
. . . | . . | | | . | | . | | |                       ,·>
. . . | . . | | | . | | . | | |                     else  fi.da(rcv
. . . | . . | | | . | | . | | |   (l@0; tg))
. . . | . . | | | . | | . | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . | | 14. ∀e:E.
. . . | . . | | | . | | . | | |   loc(e) = i
. . . | . . | | | . | | . | | |   ⇒
. . . | . . | | | . | | . | | |   (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | | |                              kind(e)))
. . . | . . | | | . | | . | | 15. ∀e:E.
. . . | . . | | | . | | . | | |   isrcv(e)
. . . | . . | | | . | | . | | |   ⇒
. . . | . . | | | . | | . | | |   lnk(e) = l
. . . | . . | | | . | | . | | |   ⇒
. . . | . . | | | . | | . | | |   (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | | |                              kind(e)))
. . . | . . | | | . | | . | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . | | |   ((λl,tg. da(rcv(l; tg))?Top))
```

```
. . . | . . | | | . | | . | | 17. ∀a:Action(i).
. . . | . . | | | . | | . | | |   ¬isnull(a)
. . . | . . | | | . | | . | | |   ⇒
. . . | . . | | | . | | . | | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . | | 18. ∀x:Id.
. . . | . . | | | . | | . | | |   vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | . | | |                     2of(ds)(x)
. . . | . . | | | . | | . | | |                   else Top fi
. . . | . . | | | . | | . | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . | | 20. j : Id
. . . | . . | | | . | | . | | 21. t : ℕ
. . . | . . | | | . | | . | | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . | | 23. True
. . . | . . | | | . | | . | | 24. j = i
. . . | . . | | | . | | . | | 25. kind(<j,t>) = k
. . . | . . | | | . | | . | | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . | | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . | | |   (eqof(product-deq(Knd;...;...;...))
. . . | . . | | | . | | . | | |    (<k,l>
. . . | . . | | | . | | . | | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | . | | |   ⇒
. . . | . . | | | . | | . | | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . | | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . | | |   =
. . . | . . | | | . | | . | | |   if source(l@0) = i→
. . . | . . | | | . | | . | | |      tagged-list-messages(λx.
. . . | . . | | | . | | . | | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | | |   else nil fi
. . . | . . | | | . | | . | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . | | |   =
. . . | . . | | | . | | . | | |   tagged-list-messages(λx.
. . . | . . | | | . | | . | | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | | 30. source(l) = i
. . . | . . | | | . | | . | | 31. (... List) ⊆r (Msg
. . . | . . | | | . | | . | | |   ((λl,tg. da(rcv(l; tg))?Top)) List)
. . . | . . | | | . | | . | | 32. L : {x:Msg| mlnk(x) = l } List
. . . | . . | | | . | | . | | 33. filter(λms.mlnk(ms) = l;m(i;t)) = L
. . . | . . | | | . | | . | | 34. l1 : IdLnk
. . . | . . | | | . | | . | | 35. x1 : t:Id×w.M(l1,t)
. . . | . . | | | . | | . | | 36. mlnk(<l1,x1>) = l
. . . | . . | | | . | | . | | 37. (<l1,x1> ∈ L)

. . . | . . | | | . | | . | | ⊢  <l,x1> = <l1,x1>
. . . | . . | | | . | | . | | | by Unfold `mlnk` -2 THEN Reduce -2
. . . | . . | | | . | | . | | |    THEN
. . . | . . | | | . | | . | | |    RW assert_pushdownC -2
. . . | . . | | | . | | . | | |
. . . | . . | | | . | | . | | 1. i : Id
. . . | . . | | | . | | . | | 2. k : Knd
. . . | . . | | | . | | . | | 3. l : IdLnk
. . . | . . | | | . | | . | | 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . | | 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . | | 6. f :
. . . | . . | | | . | | . | | |    (tg:Id
. . . | . . | | | . | | . | | |     ×State(ds)→ma-valtype(da; k)→
```

```
. . . | . . | | | . | | . | |     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . | | 7. source(l) = i
. . . | . . | | | . | | . | | 8. w : World
. . . | . . | | | . | | . | | 9. p : FairFifo
. . . | . . | | | . | | . | | 10. FairFifo
. . . | . . | | | . | | . | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | . | |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | . | |                          (source(l@0)
. . . | . . | | | . | | . | |                          ,i)→
. . . | . . | | | . | | . | |                          <ds
. . . | . . | | | . | | . | |                          ,da
. . . | . . | | | . | | . | |                          ,
. . . | . . | | | . | | . | |                          ,
. . . | . . | | | . | | . | |                          ,
. . . | . . | | | . | | . | |                          ,<[<k,l>],λx.f>
. . . | . . | | | . | | . | |                          ,
. . . | . . | | | . | | . | |                          ,
. . . | . . | | | . | | . | |                          ,•>
. . . | . . | | | . | | . | |                       else  fi.da(rcv
. . . | . . | | | . | | . | |     (l@0; tg))
. . . | . . | | | . | | . | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . | | 14. ∀e:E.
. . . | . . | | | . | | . | |     loc(e) = i
. . . | . . | | | . | | . | |     ⇒
. . . | . . | | | . | | . | |     (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | |                                kind(e)))
. . . | . . | | | . | | . | | 15. ∀e:E.
. . . | . . | | | . | | . | |     isrcv(e)
. . . | . . | | | . | | . | |     ⇒
. . . | . . | | | . | | . | |     lnk(e) = l
. . . | . . | | | . | | . | |     ⇒
. . . | . . | | | . | | . | |     (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | |                                kind(e)))
. . . | . . | | | . | | . | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . | |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . | | 17. ∀a:Action(i).
. . . | . . | | | . | | . | |     ¬isnull(a)
. . . | . . | | | . | | . | |     ⇒
. . . | . . | | | . | | . | |     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . | | 18. ∀x:Id.
. . . | . . | | | . | | . | |     vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | . | |                          2of(ds)(x)
. . . | . . | | | . | | . | |                       else Top fi
. . . | . . | | | . | | . | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . | | 20. j : Id
. . . | . . | | | . | | . | | 21. t : ℕ
. . . | . . | | | . | | . | | 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . | | 23. True
. . . | . . | | | . | | . | | 24. j = i
. . . | . . | | | . | | . | | 25. kind(<j,t>) = k
. . . | . . | | | . | | . | | 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . | | 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . | |     (eqof(product-deq(Knd;...;...;...))
```

```
. . . | . . | | | . | | . | |      (<k,l>
. . . | . . | | | . | | . | |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | . | |     ⇒
. . . | . . | | | . | | . | |     map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . | |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . | |     =
. . . | . . | | | . | | . | |     if source(l@0) = i→
. . . | . . | | | . | | . | |        tagged-list-messages(λx.
. . . | . . | | | . | | . | |        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | |     else nil fi
. . . | . . | | | . | | . | | 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . | |     =
. . . | . . | | | . | | . | |     tagged-list-messages(λx.
. . . | . . | | | . | | . | |     s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | | 30. source(l) = i
. . . | . . | | | . | | . | | 31. (... List) ⊆r (Msg
. . . | . . | | | . | | . | |     ((λl,tg. da(rcv(l; tg))?Top)) List)
. . . | . . | | | . | | . | | 32. L : {x:Msg| mlnk(x) = l } List
. . . | . . | | | . | | . | | 33. filter(λms.mlnk(ms) = l;m(i;t)) = L
. . . | . . | | | . | | . | | 34. l1 : IdLnk
. . . | . . | | | . | | . | | 35. x1 : t:Id×w.M(l1,t)
. . . | . . | | | . | | . | | 36. l1 = l
. . . | . . | | | . | | . | | 37. (<l1,x1> ∈ L)
. . . | . . | | | . | | . | | ⊢  <l,x1> = <l1,x1>
. . . | . . | | | . | | . | |   by Analyze -3 THEN Fold ‘msg‘ 0
. . . | . . | | | . | | . | |      THEN
. . . | . . | | | . | | . | |      Unfold ‘w-Msg‘ 0
. . . | . . | | | . | | . | |      THEN
. . . | . . | | | . | | . | |      Analyze
. . . | . . | | | . | | . | |      THEN
. . . | . . | | | . | | . | |      Fold ‘member‘ 0
. . . | . . | | | . | | . | |      THEN

. . . | . . | | | . | | . | |      DoSubsume
. . . | . . | | | . | | . | |      THEN
. . . | . . | | | . | | . | |      HypSubst -2 0
. . . | . . | | | . | | . | |
. . . | . . | | | . | | . | .\
. . . | . . | | | . | | . | . 1. i : Id
. . . | . . | | | . | | . | . 2. k : Knd
. . . | . . | | | . | | . | . 3. l : IdLnk
. . . | . . | | | . | | . | . 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . | . 5. da : a:Knd fp-> Type
. . . | . . | | | . | | . | . 6. f :
. . . | . . | | | . | | . | .    (tg:Id
. . . | . . | | | . | | . | .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | . | .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . | . 7. source(l) = i
. . . | . . | | | . | | . | . 8. w : World
. . . | . . | | | . | | . | . 9. p : FairFifo
. . . | . . | | | . | | . | . 10. FairFifo
. . . | . . | | | . | | . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | . | .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | . | .                      (source(l@0)
. . . | . . | | | . | | . | .                      ,i)→
. . . | . . | | | . | | . | .                      <ds
```

```
. . . | . . | | | . | | . | .                          ,da
. . . | . . | | | . | | . | .                          ,
. . . | . . | | | . | | . | .                          ,
. . . | . . | | | . | | . | .                          ,
. . . | . . | | | . | | . | .                          ,<[<k,l>],λx.f>
. . . | . . | | | . | | . | .                          ,
. . . | . . | | | . | | . | .                          ,
. . . | . . | | | . | | . | .                          ,·>
. . . | . . | | | . | | . | .                       else  fi.da(rcv
. . . | . . | | | . | | . | .     (l@0; tg))
. . . | . . | | | . | | . | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . | . 14. ∀e:E.
. . . | . . | | | . | | . | .     loc(e) = i
. . . | . . | | | . | | . | .     ⇒
. . . | . . | | | . | | . | .     (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | .                                kind(e)))
. . . | . . | | | . | | . | . 15. ∀e:E.
. . . | . . | | | . | | . | .     isrcv(e)
. . . | . . | | | . | | . | .     ⇒
. . . | . . | | | . | | . | .     lnk(e) = l
. . . | . . | | | . | | . | .     ⇒
. . . | . . | | | . | | . | .     (valtype(e) ⊆r ma-valtype(da;
. . . | . . | | | . | | . | .                                kind(e)))
. . . | . . | | | . | | . | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . | .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . | . 17. ∀a:Action(i).
. . . | . . | | | . | | . | .     ¬isnull(a)
. . . | . . | | | . | | . | .     ⇒
. . . | . . | | | . | | . | .     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . | . 18. ∀x:Id.
. . . | . . | | | . | | . | .     vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | . | .                          2of(ds)(x)
. . . | . . | | | . | | . | .                       else Top fi
. . . | . . | | | . | | . | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . | . 20. j : Id
. . . | . . | | | . | | . | . 21. t : ℕ
. . . | . . | | | . | | . | . 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . | . 23. True
. . . | . . | | | . | | . | . 24. j = i
. . . | . . | | | . | | . | . 25. kind(<j,t>) = k
. . . | . . | | | . | | . | . 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . | . 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . | .     (eqof(product-deq(Knd;...;...;...))
. . . | . . | | | . | | . | .      (<k,l>
. . . | . . | | | . | | . | .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | . | .     ⇒
. . . | . . | | | . | | . | .     map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . | .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . | .     =
. . . | . . | | | . | | . | .     if source(l@0) = i→
. . . | . . | | | . | | . | .        tagged-list-messages(λx.
. . . | . . | | | . | | . | .        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | .     else nil fi
```

```
. . . | . . | | | . | | . | . 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . | .     =
. . . | . . | | | . | | . | .     tagged-list-messages(λx.
. . . | . . | | | . | | . | .     s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . | . 30. source(l) = i
. . . | . . | | | . | | . | . 31. (... List) ⊆r (Msg
. . . | . . | | | . | | . | .     ((λl,tg. da(rcv(l; tg))?Top)) List)
. . . | . . | | | . | | . | . 32. L : {x:Msg| mlnk(x) = l } List
. . . | . . | | | . | | . | . 33. filter(λms.mlnk(ms) = l;m(i;t)) = L
. . . | . . | | | . | | . | . 34. l1 : IdLnk
. . . | . . | | | . | | . | . 35. x1 : t:Id×w.M(l1,t)
. . . | . . | | | . | | . | . 36. mlnk(<l1,x1>) = l
. . . | . . | | | . | | . | . 37. i1 : ℕ
. . . | . . | | | . | | . | . 38. i1<||L|| & <l1,x1> = L[i1]
. . . | . . | | | . | | . | . ⊢  l = l  by RW assert_pushdownC 0
. . . | . . | | | . | | . |
. . . | . . | | | . | | .\
. . . | . . | | | . | | . 1. i : Id
. . . | . . | | | . | | . 2. k : Knd
. . . | . . | | | . | | . 3. l : IdLnk
. . . | . . | | | . | | . 4. ds : x:Id fp-> Type
. . . | . . | | | . | | . 5. da : a:Knd fp-> Type

. . . | . . | | | . | | . 6. f :
. . . | . . | | | . | | .    (tg:Id
. . . | . . | | | . | | .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | | .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | | . 7. source(l) = i
. . . | . . | | | . | | . 8. w : World
. . . | . . | | | . | | . 9. p : FairFifo
. . . | . . | | | . | | . 10. FairFifo
. . . | . . | | | . | | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | | .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | | .                        (source(l@0)
. . . | . . | | | . | | .                        ,i)→
. . . | . . | | | . | | .                        <ds
. . . | . . | | | . | | .                        ,da
. . . | . . | | | . | | .                        ,
. . . | . . | | | . | | .                        ,
. . . | . . | | | . | | .                        ,
. . . | . . | | | . | | .                        ,<[<k,l>],λx.f>
. . . | . . | | | . | | .                        ,
. . . | . . | | | . | | .                        ,
. . . | . . | | | . | | .                        ,·>
. . . | . . | | | . | | .                    else  fi.da(rcv(l@0;
. . . | . . | | | . | | .                                    tg))
. . . | . . | | | . | | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | | . 14. ∀e:E.
. . . | . . | | | . | | .     loc(e) = i
. . . | . . | | | . | | .     ⇒
. . . | . . | | | . | | .     (valtype(e) ⊆r ma-valtype
. . . | . . | | | . | | .      (da; kind(e)))
. . . | . . | | | . | | . 15. ∀e:E.
. . . | . . | | | . | | .     isrcv(e)
. . . | . . | | | . | | .     ⇒
```

```
. . . | . . | | | . | | . .     lnk(e) = l
. . . | . . | | | . | | . .     ⇒
. . . | . . | | | . | | . .     (valtype(e) ⊆r ma-valtype
. . . | . . | | | . | | . .      (da; kind(e)))
. . . | . . | | | . | | . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | | . .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | | . . 17. ∀a:Action(i).
. . . | . . | | | . | | . .     ¬isnull(a)
. . . | . . | | | . | | . .     ⇒
. . . | . . | | | . | | . .     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | | . . 18. ∀x:Id.
. . . | . . | | | . | | . .     vartype(i;x) ⊆r if ...→
. . . | . . | | | . | | . .                        2of(ds)(x)
. . . | . . | | | . | | . .                    else Top fi
. . . | . . | | | . | | . . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | | . . 20. j : Id
. . . | . . | | | . | | . . 21. t : ℕ
. . . | . . | | | . | | . . 22. ¬isnull(a(j;t))
. . . | . . | | | . | | . . 23. True
. . . | . . | | | . | | . . 24. j = i
. . . | . . | | | . | | . . 25. kind(<j,t>) = k
. . . | . . | | | . | | . . 26. ¬isnull(a(i;t))
. . . | . . | | | . | | . . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | | . . 28. ∀l@0:IdLnk.
. . . | . . | | | . | | . .     (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . | | . .      (<k,l>
. . . | . . | | | . | | . .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | | . .     ⇒
. . . | . . | | | . | | . .     map(λms.2of(ms);filter(λms.
. . . | . . | | | . | | . .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | | . .     =
. . . | . . | | | . | | . .     if source(l@0) = i→
. . . | . . | | | . | | . .        tagged-list-messages(λx.
. . . | . . | | | . | | . .        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . .     else nil fi
. . . | . . | | | . | | . . 29. withlnk(l;m(i;t))
. . . | . . | | | . | | . .     =
. . . | . . | | | . | | . .     tagged-list-messages(λx.
. . . | . . | | | . | | . .     s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | | . . 30. source(l) = i
. . . | . . | | | . | | . . 31. ({m:Msg(w.M)
. . . | . . | | | . | | . .       | source(mlnk(m)) = i } List) ⊆r ...
. . . | . . | | | . | | . . 32. L : {x:Msg| mlnk(x) = l } List
. . . | . . | | | . | | . . 33. filter(λms.mlnk(ms) = l;m(i;t)) = L
. . . | . . | | | . | | . . 34. map((λx.<l,x>) o (λms.2of(ms));L) = L
. . . | . . | | | . | | . . ⊢  map((λx.<l,x>) o (λms.2of(ms));L) = L
. . . | . . | | | . | | .   by HypSubst -1 0
. . . | . . | | | . | | .      THEN
. . . | . . | | | . | | .      Try
. . . | . . | | | . | | .      (Analyze -1 THEN Unfold ‘mlnk‘ 0
. . . | . . | | | . | | .       THEN
. . . | . . | | | . | | .       Reduce 0
. . . | . . | | | . | | .       THEN
. . . | . . | | | . | | .       Complete Auto)
. . . | . . | | | . | | .      THEN
```

```
. . . | . . | | | . | | . .      Try (Fold ‘member‘ 0)
. . . | . . | | | . | | . .      THEN
. . . | . . | | | . | | . .      Analyze -1
. . . | . . | | | . | | . .      THEN
. . . | . . | | | . | | . .      Analyze
. . . | . . | | | . | | . .      THEN
. . . | . . | | | . | | . .      Try (Fold ‘w-Msg‘ 0)
. . . | . . | | | . | | . .      THEN
. . . | . . | | | . | | . .      RW assert_pushdownC -1
. . . | . . | | | . | | . .      THEN
. . . | . . | | | . | | . .      Try (HypSubst -1 0)
. . . | . . | | | . | | . .      THEN
. . . | . . | | | . | | . .      DVar ‘u’
. . . | . . | | | . | | . .      THEN
. . . | . . | | | . | | . .      Unfold ‘mlnk‘ 0
. . . | . . | | | . | |
. . . | . . | | | . | .\
. . . | . . | | | . | . .....subterm .....
. . . | . . | | | . | . 1. i : Id
. . . | . . | | | . | . 2. k : Knd
. . . | . . | | | . | . 3. l : IdLnk
. . . | . . | | | . | . 4. ds : x:Id fp-> Type
. . . | . . | | | . | . 5. da : a:Knd fp-> Type
. . . | . . | | | . | . 6. f :
. . . | . . | | | . | .    (tg:Id
. . . | . . | | | . | .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . | .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . | . 7. source(l) = i
. . . | . . | | | . | . 8. w : World
. . . | . . | | | . | . 9. p : FairFifo
. . . | . . | | | . | . 10. FairFifo
. . . | . . | | | . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . | .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . | .                         (source(l@0)
. . . | . . | | | . | .                         ,i)→
. . . | . . | | | . | .                         <ds
. . . | . . | | | . | .                         ,da
. . . | . . | | | . | .                         ,
. . . | . . | | | . | .                         ,
. . . | . . | | | . | .                         ,
. . . | . . | | | . | .                         ,<[<k,l>],λx.f>
. . . | . . | | | . | .                         ,
. . . | . . | | | . | .                         ,
. . . | . . | | | . | .                         ,·>
. . . | . . | | | . | .                       else  fi.da(rcv(l@0; tg))
. . . | . . | | | . | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . | . 14. ∀e:E.
. . . | . . | | | . | .     loc(e) = i
. . . | . . | | | . | .     ⇒
. . . | . . | | | . | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | . 15. ∀e:E.
. . . | . . | | | . | .     isrcv(e)
. . . | . . | | | . | .     ⇒
. . . | . . | | | . | .     lnk(e) = l
```

```
. . . | . . | | | . | .     ⇒
. . . | . . | | | . | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . | .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . | . 17. ∀a:Action(i).
. . . | . . | | | . | .     ¬isnull(a)
. . . | . . | | | . | .     ⇒
. . . | . . | | | . | .     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . | . 18. ∀x:Id.
. . . | . . | | | . | .     vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | . | .                                          ds))→
. . . | . . | | | . | .                          2of(ds)(x)
. . . | . . | | | . | .                       else Top fi
. . . | . . | | | . | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . | . 20. j : Id
. . . | . . | | | . | . 21. t : ℕ
. . . | . . | | | . | . 22. ¬isnull(a(j;t))
. . . | . . | | | . | . 23. True
. . . | . . | | | . | . 24. j = i
. . . | . . | | | . | . 25. kind(<j,t>) = k
. . . | . . | | | . | . 26. ¬isnull(a(i;t))
. . . | . . | | | . | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . | . 28. ∀l@0:IdLnk.
. . . | . . | | | . | .     (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . | .      (<k,l>
. . . | . . | | | . | .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . | .     ⇒
. . . | . . | | | . | .     map(λms.2of(ms);filter(λms.
. . . | . . | | | . | .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . | .     =
. . . | . . | | | . | .     if source(l@0) = i→
. . . | . . | | | . | .        tagged-list-messages(λx.
. . . | . . | | | . | .        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | .     else nil fi
. . . | . . | | | . | . 29. withlnk(l;m(i;t))
. . . | . . | | | . | .     =
. . . | . . | | | . | .     tagged-list-messages(λx.
. . . | . . | | | . | .     s(i;t).x;val(a(i;t));f)
. . . | . . | | | . | . 30. source(l) = i
. . . | . . | | | . | . 31. ({m:Msg(w.M)
. . . | . . | | | . | .       | source(mlnk(m)) = i } List) ⊆r (Msg
. . . | . . | | | . | .     ((λl,tg. da(rcv(l; tg))?Top)) List)
. . . | . . | | | . | . ⊢  (λx.<l,x>) o (λms.2of(ms))
. . . | . . | | | . | .    ∈ {x:{m:Msg(w.M)| source(mlnk(m)) = i }
. . . | . . | | | . | .       | (λms.mlnk(ms) = l)(x) }
. . . | . . | | | . | .      →{m:Msg(w.M)| source(mlnk(m)) = i }
. . . | . . | | | . | .   by Unfold ‘compose‘ 0 THEN Reduce 0
. . . | . . | | | . | .      THEN
. . . | . . | | | . | .      Analyze -1
. . . | . . | | | . | .      THEN
. . . | . . | | | . | .      Analyze -2
. . . | . . | | | . | .      THEN
. . . | . . | | | . | .      Analyze -3
. . . | . . | | | . | .      THEN
. . . | . . | | | . | .      Analyze -3
```

```
. . . | . . | | | . | .      THEN
. . . | . . | | | . | .      Reduce 0

. . . | . . | | | . | .      THEN
. . . | . . | | | . | .      Fold ‘msg‘ 0
. . . | . . | | | . | .      THEN
. . . | . . | | | . | .      Analyze
. . . | . . | | | . | .      THEN
. . . | . . | | | . | .      AllHyps
. . . | . . | | | . | .      (λh.Unfold ‘mlnk‘ h THEN Reduce h)
. . . | . . | | | . | .      THEN
. . . | . . | | | . | .      RW assert_pushdownC -1
. . . | . . | | | . | .      THEN
. . . | . . | | | . | .      Reduce 0
. . . | . . | | | . | .      THEN
. . . | . . | | | . | .      Try DoSubsume
. . . | . . | | | . | .      THEN
. . . | . . | | | . | .      HypSubst -1 0
. . . | . . | | | . |
. . . | . . | | | . .\
. . . | . . | | | . . 1. i : Id
. . . | . . | | | . . 2. k : Knd
. . . | . . | | | . . 3. l : IdLnk
. . . | . . | | | . . 4. ds : x:Id fp-> Type
. . . | . . | | | . . 5. da : a:Knd fp-> Type
. . . | . . | | | . . 6. f :
. . . | . . | | | . . |  (tg:Id
. . . | . . | | | . . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . 7. source(l) = i
. . . | . . | | | . . 8. w : World
. . . | . . | | | . . 9. p : FairFifo
. . . | . . | | | . . 10. FairFifo
. . . | . . | | | . . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . |                       (source(l@0)
. . . | . . | | | . . |                       ,i)→
. . . | . . | | | . . |                       <ds
. . . | . . | | | . . |                       ,da
. . . | . . | | | . . |                       ,
. . . | . . | | | . . |                       ,
. . . | . . | | | . . |                       ,
. . . | . . | | | . . |                       ,<[<k,l>],λx.f>
. . . | . . | | | . . |                       ,
. . . | . . | | | . . |                       ,
. . . | . . | | | . . |                       ,·>
. . . | . . | | | . . |                     else  fi.da(rcv(l@0; tg))
. . . | . . | | | . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . 14. ∀e:E.
. . . | . . | | | . . |   loc(e) = i
. . . | . . | | | . . |   ⇒
. . . | . . | | | . . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . 15. ∀e:E.
. . . | . . | | | . . |   isrcv(e)
. . . | . . | | | . . |   ⇒
```

```
. . . | . . | | | . . |   lnk(e) = l
. . . | . . | | | . . |   ⇒
. . . | . . | | | . . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . 17. ∀a:Action(i).
. . . | . . | | | . . |   ¬isnull(a)
. . . | . . | | | . . |   ⇒
. . . | . . | | | . . |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . 18. ∀x:Id.
. . . | . . | | | . . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | | . . |                                        ds))→
. . . | . . | | | . . |                     2of(ds)(x)
. . . | . . | | | . . |                   else Top fi
. . . | . . | | | . . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . 20. j : Id
. . . | . . | | | . . 21. t : ℕ
. . . | . . | | | . . 22. ¬isnull(a(j;t))
. . . | . . | | | . . 23. True
. . . | . . | | | . . 24. j = i
. . . | . . | | | . . 25. kind(<j,t>) = k
. . . | . . | | | . . 26. ¬isnull(a(i;t))
. . . | . . | | | . . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . 28. ∀l@0:IdLnk.
. . . | . . | | | . . |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . . |    (<k,l>
. . . | . . | | | . . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . |   ⇒
. . . | . . | | | . . |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . . |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . |   =
. . . | . . | | | . . |   if source(l@0) = i→
. . . | . . | | | . . |      tagged-list-messages(λx.
. . . | . . | | | . . |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . |   else nil fi
. . . | . . | | | . . 29. withlnk(l;m(i;t))
. . . | . . | | | . . |   =
. . . | . . | | | . . |   tagged-list-messages(λx.
. . . | . . | | | . . |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . 30. source(l) = i
. . . | . . | | | . . 31. map((λx.<l,x>) o (λms.2of(ms));filter(λms.
. . . | . . | | | . . |   mlnk(ms) = l;m(i;t)))
. . . | . . | | | . . |   =
. . . | . . | | | . . |   filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . . ⊢  filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . | | | . . |  =
. . . | . . | | | . . |  map((λx.<l,x>) o (λms.2of(ms));filter(λms.

. . . | . . | | | . . |  mlnk(ms) = l;m(i;t)))
. . . | . . | | | . . | by Assert
. . . | . . | | | . . |    (filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . | | | . . |     =
. . . | . . | | | . . |     filter(λms.mlnk(ms) = l;m(i;t)))
. . . | . . | | | . . |
. . . | . . | | | . . |\
. . . | . . | | | . . | .....assertion .....
```

```
. . . | . . | | | . . | 1. i : Id
. . . | . . | | | . . | 2. k : Knd
. . . | . . | | | . . | 3. l : IdLnk
. . . | . . | | | . . | 4. ds : x:Id fp-> Type
. . . | . . | | | . . | 5. da : a:Knd fp-> Type
. . . | . . | | | . . | 6. f :
. . . | . . | | | . . | |  (tg:Id
. . . | . . | | | . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . | 7. source(l) = i
. . . | . . | | | . . | 8. w : World
. . . | . . | | | . . | 9. p : FairFifo
. . . | . . | | | . . | 10. FairFifo
. . . | . . | | | . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . | |                       (source(l@0)
. . . | . . | | | . . | |                       ,i)→
. . . | . . | | | . . | |                       <ds
. . . | . . | | | . . | |                       ,da
. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,<[<k,l>],λx.f>
. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,•>
. . . | . . | | | . . | |                    else  fi.da(rcv(l@0; tg))
. . . | . . | | | . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . | 14. ∀e:E.
. . . | . . | | | . . | |   loc(e) = i
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | 15. ∀e:E.
. . . | . . | | | . . | |   isrcv(e)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   lnk(e) = l
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . | 17. ∀a:Action(i).
. . . | . . | | | . . | |   ¬isnull(a)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . | 18. ∀x:Id.
. . . | . . | | | . . | |   vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | . . | |                                        ds))→
. . . | . . | | | . . | |                      2of(ds)(x)
. . . | . . | | | . . | |                   else Top fi
. . . | . . | | | . . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . | 20. j : Id
. . . | . . | | | . . | 21. t : ℕ
. . . | . . | | | . . | 22. ¬isnull(a(j;t))
. . . | . . | | | . . | 23. True
```

```
. . . | . . | | | . . | 24. j = i
. . . | . . | | | . . | 25. kind(<j,t>) = k
. . . | . . | | | . . | 26. ¬isnull(a(i;t))
. . . | . . | | | . . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . | 28. ∀l@0:IdLnk.
. . . | . . | | | . . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . . | |    (<k,l>
. . . | . . | | | . . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . . | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . | |   =
. . . | . . | | | . . | |   if source(l@0) = i→
. . . | . . | | | . . | |      tagged-list-messages(λx.
. . . | . . | | | . . | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . | |   else nil fi
. . . | . . | | | . . | 29. withlnk(l;m(i;t))
. . . | . . | | | . . | |   =
. . . | . . | | | . . | |   tagged-list-messages(λx.
. . . | . . | | | . . | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . | 30. source(l) = i
. . . | . . | | | . . | 31. map((λx.<l,x>) o (λms.2of(ms));filter(λms.
. . . | . . | | | . . | |   mlnk(ms) = l;m(i;t)))
. . . | . . | | | . . | |   =
. . . | . . | | | . . | |   filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . . | ⊢  filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . | | | . . | |  =
. . . | . . | | | . . | |  filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . . | | by Analyze THEN Try (Complete Auto)
. . . | . . | | | . . | |
. . . | . . | | | . . | .....subterm .....
. . . | . . | | | . . | 1. i : Id
. . . | . . | | | . . | 2. k : Knd
. . . | . . | | | . . | 3. l : IdLnk
. . . | . . | | | . . | 4. ds : x:Id fp-> Type
. . . | . . | | | . . | 5. da : a:Knd fp-> Type
. . . | . . | | | . . | 6. f :
. . . | . . | | | . . | |  (tg:Id
. . . | . . | | | . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . | 7. source(l) = i
. . . | . . | | | . . | 8. w : World
. . . | . . | | | . . | 9. p : FairFifo
. . . | . . | | | . . | 10. FairFifo
. . . | . . | | | . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . | |                      (source(l@0)
. . . | . . | | | . . | |                      ,i)→
. . . | . . | | | . . | |                      <ds
. . . | . . | | | . . | |                      ,da
. . . | . . | | | . . | |                      ,
. . . | . . | | | . . | |                      ,
. . . | . . | | | . . | |                      ,
. . . | . . | | | . . | |                      ,<[<k,l>],λx.f>
. . . | . . | | | . . | |                      ,
```

```
. . . | . . | | | . . | |                        ,
. . . | . . | | | . . | |                        ,·>
. . . | . . | | | . . | |                    else  fi.da(rcv(l@0; tg))
. . . | . . | | | . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . | 14. ∀e:E.
. . . | . . | | | . . | |   loc(e) = i
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | 15. ∀e:E.
. . . | . . | | | . . | |   isrcv(e)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   lnk(e) = l
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . | 17. ∀a:Action(i).
. . . | . . | | | . . | |   ¬isnull(a)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . | 18. ∀x:Id.
. . . | . . | | | . . | |   vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | . . | |                                      ds))→
. . . | . . | | | . . | |                     2of(ds)(x)
. . . | . . | | | . . | |                  else Top fi
. . . | . . | | | . . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . | 20. j : Id
. . . | . . | | | . . | 21. t : ℕ
. . . | . . | | | . . | 22. ¬isnull(a(j;t))
. . . | . . | | | . . | 23. True
. . . | . . | | | . . | 24. j = i
. . . | . . | | | . . | 25. kind(<j,t>) = k
. . . | . . | | | . . | 26. ¬isnull(a(i;t))
. . . | . . | | | . . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . | 28. ∀l@0:IdLnk.
. . . | . . | | | . . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . . | |    (<k,l>
. . . | . . | | | . . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . . | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . | |   =
. . . | . . | | | . . | |   if source(l@0) = i→
. . . | . . | | | . . | |      tagged-list-messages(λx.
. . . | . . | | | . . | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . | |   else nil fi
. . . | . . | | | . . | 29. withlnk(l;m(i;t))
. . . | . . | | | . . | |   =
. . . | . . | | | . . | |   tagged-list-messages(λx.
. . . | . . | | | . . | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . | 30. source(l) = i
. . . | . . | | | . . | 31. map((λx.<l,x>) o (λms.2of(ms));filter(λms.
. . . | . . | | | . . | |   mlnk(ms) = l;m(i;t)))
. . . | . . | | | . . | |   =
```

```
. . . | . . | | | . . | |   filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . . | ⊢  m(j;t) = m(i;t)
. . . | . . | | | . . | | by Analyze THEN Fold `member` 0
. . . | . . | | | . . | |
. . . | . . | | | . . | 1. i : Id
. . . | . . | | | . . | 2. k : Knd
. . . | . . | | | . . | 3. l : IdLnk
. . . | . . | | | . . | 4. ds : x:Id fp-> Type
. . . | . . | | | . . | 5. da : a:Knd fp-> Type
. . . | . . | | | . . | 6. f :
. . . | . . | | | . . | |  (tg:Id
. . . | . . | | | . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . | 7. source(l) = i
. . . | . . | | | . . | 8. w : World
. . . | . . | | | . . | 9. p : FairFifo
. . . | . . | | | . . | 10. FairFifo
. . . | . . | | | . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . | |                       (source(l@0)
. . . | . . | | | . . | |                       ,i)→
. . . | . . | | | . . | |                       <ds
. . . | . . | | | . . | |                       ,da
. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,<[<k,l>],λx.f>

. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,
. . . | . . | | | . . | |                       ,.>
. . . | . . | | | . . | |                  else  fi.da(rcv(l@0; tg))
. . . | . . | | | . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . | 14. ∀e:E.
. . . | . . | | | . . | |   loc(e) = i
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | 15. ∀e:E.
. . . | . . | | | . . | |   isrcv(e)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   lnk(e) = l
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . | 17. ∀a:Action(i).
. . . | . . | | | . . | |   ¬isnull(a)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . | 18. ∀x:Id.
. . . | . . | | | . . | |   vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | . . | |                                        ds))→
. . . | . . | | | . . | |                      2of(ds)(x)
. . . | . . | | | . . | |                   else Top fi
. . . | . . | | | . . | 19. IdDeq ∈ EqDecider(Id)
```

```
. . . | . . | | | . . | 20. j : Id
. . . | . . | | | . . | 21. t : ℕ
. . . | . . | | | . . | 22. ¬isnull(a(j;t))
. . . | . . | | | . . | 23. True
. . . | . . | | | . . | 24. j = i
. . . | . . | | | . . | 25. kind(<j,t>) = k
. . . | . . | | | . . | 26. ¬isnull(a(i;t))
. . . | . . | | | . . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . | 28. ∀l@0:IdLnk.
. . . | . . | | | . . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . . | |    (<k,l>
. . . | . . | | | . . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . | |   ⇒
. . . | . . | | | . . | |   map(λms.2of(ms);filter(λms.
. . . | . . | | | . . | |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . | |   =
. . . | . . | | | . . | |   if source(l@0) = i→
. . . | . . | | | . . | |      tagged-list-messages(λx.
. . . | . . | | | . . | |      s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . | |   else nil fi
. . . | . . | | | . . | 29. withlnk(l;m(i;t))
. . . | . . | | | . . | |   =
. . . | . . | | | . . | |   tagged-list-messages(λx.
. . . | . . | | | . . | |   s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . | 30. source(l) = i
. . . | . . | | | . . | 31. map((λx.<l,x>) o (λms.2of(ms));filter(λms.
. . . | . . | | | . . | |   mlnk(ms) = l;m(i;t)))
. . . | . . | | | . . | |   =
. . . | . . | | | . . | |   filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . . | 32. m(j;t) = m(i;t)
. . . | . . | | | . . | 33. {m:Msg(w.M)| source(mlnk(m)) = j } List
. . . | . . | | | . . | 34. {m:Msg(w.M)| source(mlnk(m)) = j } List
. . . | . . | | | . . | 35. u : Msg(w.M)
. . . | . . | | | . . | 36. source(mlnk(u)) = j
. . . | . . | | | . . | ⊢  u ∈ Msg((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . | | by Assert
. . . | . . | | | . . | |    (u ∈ {u:Msg(w.M)| source(mlnk(u)) = i })
. . . | . . | | | . . | |
. . . | . . | | | . . | |\
. . . | . . | | | . . | | .....assertion .....
. . . | . . | | | . . | | 1. i : Id
. . . | . . | | | . . | | 2. k : Knd
. . . | . . | | | . . | | 3. l : IdLnk
. . . | . . | | | . . | | 4. ds : x:Id fp-> Type
. . . | . . | | | . . | | 5. da : a:Knd fp-> Type
. . . | . . | | | . . | | 6. f :
. . . | . . | | | . . | |    (tg:Id
. . . | . . | | | . . | |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . | |     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . | | 7. source(l) = i
. . . | . . | | | . . | | 8. w : World
. . . | . . | | | . . | | 9. p : FairFifo
. . . | . . | | | . . | | 10. FairFifo
. . . | . . | | | . . | | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . | |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
```

```
. . . | . . | | | . . | |                          (source(l@0)
. . . | . . | | | . . | |                          ,i)→
. . . | . . | | | . . | |                          <ds
. . . | . . | | | . . | |                          ,da
. . . | . . | | | . . | |                          ,
. . . | . . | | | . . | |                          ,
. . . | . . | | | . . | |                          ,
. . . | . . | | | . . | |                          ,<[<k,l>],λx.f>
. . . | . . | | | . . | |                          ,
. . . | . . | | | . . | |                          ,
. . . | . . | | | . . | |                          ,·>
. . . | . . | | | . . | |                       else  fi.da(rcv
. . . | . . | | | . . | |     (l@0; tg))
. . . | . . | | | . . | | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . | | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . | | 14. ∀e:E.
. . . | . . | | | . . | |     loc(e) = i

. . . | . . | | | . . | |     ⇒
. . . | . . | | | . . | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | | 15. ∀e:E.
. . . | . . | | | . . | |     isrcv(e)
. . . | . . | | | . . | |     ⇒
. . . | . . | | | . . | |     lnk(e) = l
. . . | . . | | | . . | |     ⇒
. . . | . . | | | . . | |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . | |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . | | 17. ∀a:Action(i).
. . . | . . | | | . . | |     ¬isnull(a)
. . . | . . | | | . . | |     ⇒
. . . | . . | | | . . | |     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . | | 18. ∀x:Id.
. . . | . . | | | . . | |     vartype(i;x) ⊆r if ...→
. . . | . . | | | . . | |                          2of(ds)(x)
. . . | . . | | | . . | |                       else Top fi
. . . | . . | | | . . | | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . | | 20. j : Id
. . . | . . | | | . . | | 21. t : ℕ
. . . | . . | | | . . | | 22. ¬isnull(a(j;t))
. . . | . . | | | . . | | 23. True
. . . | . . | | | . . | | 24. j = i
. . . | . . | | | . . | | 25. kind(<j,t>) = k
. . . | . . | | | . . | | 26. ¬isnull(a(i;t))
. . . | . . | | | . . | | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . | | 28. ∀l@0:IdLnk.
. . . | . . | | | . . | |     (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . . | |      (<k,l>
. . . | . . | | | . . | |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . | |     ⇒
. . . | . . | | | . . | |     map(λms.2of(ms);filter(λms.
. . . | . . | | | . . | |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . | |     =
. . . | . . | | | . . | |     if source(l@0) = i→
. . . | . . | | | . . | |        tagged-list-messages(λx.
. . . | . . | | | . . | |        s(i;t).x;val(a(i;t));f)
```

```
. . . | . . | | | . . | |     else nil fi
. . . | . . | | | . . | | 29. withlnk(l;m(i;t))
. . . | . . | | | . . | |     =
. . . | . . | | | . . | |     tagged-list-messages(λx.
. . . | . . | | | . . | |     s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . | | 30. source(l) = i
. . . | . . | | | . . | | 31. map((λx.<l,x>)
. . . | . . | | | . . | |      o (λms.2of(ms));filter(λms.
. . . | . . | | | . . | |     mlnk(ms) = l;m(i;t)))
. . . | . . | | | . . | |     =
. . . | . . | | | . . | |     filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . . | | 32. m(j;t) = m(i;t)
. . . | . . | | | . . | | 33. {m:Msg(w.M)| source(mlnk(m)) = j } List
. . . | . . | | | . . | | 34. {m:Msg(w.M)| source(mlnk(m)) = j } List
. . . | . . | | | . . | | 35. u : Msg(w.M)
. . . | . . | | | . . | | 36. source(mlnk(u)) = j
. . . | . . | | | . . | | ⊢  u ∈ {u:Msg(w.M)| source(mlnk(u)) = i }
. . . | . . | | | . . | |   by Analyze
. . . | . . | | | . . | |
. . . | . . | | | . . | | .\
. . . | . . | | | . . | | . 1. i : Id
. . . | . . | | | . . | | . 2. k : Knd
. . . | . . | | | . . | | . 3. l : IdLnk
. . . | . . | | | . . | | . 4. ds : x:Id fp-> Type
. . . | . . | | | . . | | . 5. da : a:Knd fp-> Type
. . . | . . | | | . . | | . 6. f :
. . . | . . | | | . . | | .    (tg:Id
. . . | . . | | | . . | | .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . | | .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . | | . 7. source(l) = i
. . . | . . | | | . . | | . 8. w : World
. . . | . . | | | . . | | . 9. p : FairFifo
. . . | . . | | | . . | | . 10. FairFifo
. . . | . . | | | . . | | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . | | .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . | | .                         (source(l@0)
. . . | . . | | | . . | | .                         ,i)→
. . . | . . | | | . . | | .                         <ds
. . . | . . | | | . . | | .                         ,da
. . . | . . | | | . . | | .                         ,
. . . | . . | | | . . | | .                         ,
. . . | . . | | | . . | | .                         ,
. . . | . . | | | . . | | .                         ,<[<k,l>],λx.f>
. . . | . . | | | . . | | .                         ,
. . . | . . | | | . . | | .                         ,
. . . | . . | | | . . | | .                         ,·>
. . . | . . | | | . . | | .                    else  fi.da(rcv
. . . | . . | | | . . | | .     (l@0; tg))
. . . | . . | | | . . | | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | | . . | | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . | | . 14. ∀e:E.
. . . | . . | | | . . | | .     loc(e) = i
. . . | . . | | | . . | | .     ⇒
. . . | . . | | | . . | | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | | . 15. ∀e:E.
```

```
. . . | . . | | | . . | .     isrcv(e)
. . . | . . | | | . . | .     ⇒
. . . | . . | | | . . | .     lnk(e) = l
. . . | . . | | | . . | .     ⇒
. . . | . . | | | . . | .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . | .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . | . 17. ∀a:Action(i).
. . . | . . | | | . . | .     ¬isnull(a)
. . . | . . | | | . . | .     ⇒
. . . | . . | | | . . | .     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . | . 18. ∀x:Id.
. . . | . . | | | . . | .     vartype(i;x) ⊆r if ...→
. . . | . . | | | . . | .                        2of(ds)(x)
. . . | . . | | | . . | .                    else Top fi
. . . | . . | | | . . | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | | . . | . 20. j : Id
. . . | . . | | | . . | . 21. t : ℕ
. . . | . . | | | . . | . 22. ¬isnull(a(j;t))
. . . | . . | | | . . | . 23. True
. . . | . . | | | . . | . 24. j = i
. . . | . . | | | . . | . 25. kind(<j,t>) = k
. . . | . . | | | . . | . 26. ¬isnull(a(i;t))
. . . | . . | | | . . | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . | . 28. ∀l@0:IdLnk.
. . . | . . | | | . . | .     (eqof(product-deq(Knd;IdLnk;...;...))
. . . | . . | | | . . | .      (<k,l>
. . . | . . | | | . . | .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . | .     ⇒
. . . | . . | | | . . | .     map(λms.2of(ms);filter(λms.
. . . | . . | | | . . | .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . | .     =
. . . | . . | | | . . | .     if source(l@0) = i→
. . . | . . | | | . . | .        tagged-list-messages(λx.
. . . | . . | | | . . | .        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . | .     else nil fi
. . . | . . | | | . . | . 29. withlnk(l;m(i;t))
. . . | . . | | | . . | .     =
. . . | . . | | | . . | .     tagged-list-messages(λx.
. . . | . . | | | . . | .     s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . | . 30. source(l) = i
. . . | . . | | | . . | . 31. map((λx.<l,x>)
. . . | . . | | | . . | .      o (λms.2of(ms));filter(λms.
. . . | . . | | | . . | .     mlnk(ms) = l;m(i;t)))
. . . | . . | | | . . | .     =
. . . | . . | | | . . | .     filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . . | . 32. m(j;t) = m(i;t)
. . . | . . | | | . . | . 33. {m:Msg(w.M)| source(mlnk(m)) = j } List
. . . | . . | | | . . | . 34. {m:Msg(w.M)| source(mlnk(m)) = j } List
. . . | . . | | | . . | . 35. u : Msg(w.M)
. . . | . . | | | . . | . 36. source(mlnk(u)) = j
. . . | . . | | | . . | . 37. u ∈ {u:Msg(w.M)| source(mlnk(u)) = i }
. . . | . . | | | . . | . ⊢  u ∈ Msg((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . | .   by DoSubsume
. . . | . . | | | . . | .      THEN
```

```
. . . | . . | | | . . | .      AllHyps
. . . | . . | | | . . | .      (λh.Unfold ‘es-Msg‘ h THEN Reduce h)
. . . | . . | | | . . |
. . . | . . | | | . . .\
. . . | . . | | | . . . 1. i : Id
. . . | . . | | | . . . 2. k : Knd
. . . | . . | | | . . . 3. l : IdLnk
. . . | . . | | | . . . 4. ds : x:Id fp-> Type
. . . | . . | | | . . . 5. da : a:Knd fp-> Type
. . . | . . | | | . . . 6. f :
. . . | . . | | | . . .    (tg:Id
. . . | . . | | | . . .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | | . . .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | | . . . 7. source(l) = i
. . . | . . | | | . . . 8. w : World
. . . | . . | | | . . . 9. p : FairFifo
. . . | . . | | | . . . 10. FairFifo
. . . | . . | | | . . . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | | . . .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)
. . . | . . | | | . . .                         (source(l@0)
. . . | . . | | | . . .                         ,i)→
. . . | . . | | | . . .                         <ds
. . . | . . | | | . . .                         ,da
. . . | . . | | | . . .                         ,
. . . | . . | | | . . .                         ,
. . . | . . | | | . . .                         ,
. . . | . . | | | . . .                         ,<[<k,l>],λx.f>
. . . | . . | | | . . .                         ,
. . . | . . | | | . . .                         ,
. . . | . . | | | . . .                         ,•>
. . . | . . | | | . . .                    else  fi.da(rcv(l@0; tg))
. . . | . . | | | . . . 12. (eqof(IdDeq)(i,i)) ∼ true2
. . . | . . | | | . . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | | . . . 14. ∀e:E.
. . . | . . | | | . . .     loc(e) = i
. . . | . . | | | . . .     ⇒
. . . | . . | | | . . .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . . 15. ∀e:E.
. . . | . . | | | . . .     isrcv(e)
. . . | . . | | | . . .     ⇒
. . . | . . | | | . . .     lnk(e) = l
. . . | . . | | | . . .     ⇒
. . . | . . | | | . . .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | | . . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | | . . .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | | . . . 17. ∀a:Action(i).
. . . | . . | | | . . .     ¬isnull(a)
. . . | . . | | | . . .     ⇒
. . . | . . | | | . . .     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | | . . . 18. ∀x:Id.
. . . | . . | | | . . .     vartype(i;x) ⊆r if deq-member(...;x;1of(
. . . | . . | | | . . .                                       ds))→
. . . | . . | | | . . .                        2of(ds)(x)
. . . | . . | | | . . .                     else Top fi
. . . | . . | | | . . . 19. IdDeq ∈ EqDecider(Id)
```

```
. . . | . . | | | . . . 20. j : Id
. . . | . . | | | . . . 21. t : ℕ
. . . | . . | | | . . . 22. ¬isnull(a(j;t))
. . . | . . | | | . . . 23. True
. . . | . . | | | . . . 24. j = i
. . . | . . | | | . . . 25. kind(<j,t>) = k
. . . | . . | | | . . . 26. ¬isnull(a(i;t))
. . . | . . | | | . . . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | | . . . 28. ∀l@0:IdLnk.
. . . | . . | | | . . .     (eqof(product-deq(Knd;IdLnk;KindDeq;...))
. . . | . . | | | . . .      (<k,l>
. . . | . . | | | . . .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | | . . .     ⇒
. . . | . . | | | . . .     map(λms.2of(ms);filter(λms.
. . . | . . | | | . . .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | | . . .     =
. . . | . . | | | . . .     if source(l@0) = i→
. . . | . . | | | . . .        tagged-list-messages(λx.
. . . | . . | | | . . .        s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . .     else nil fi
. . . | . . | | | . . . 29. withlnk(l;m(i;t))
. . . | . . | | | . . .     =
. . . | . . | | | . . .     tagged-list-messages(λx.
. . . | . . | | | . . .     s(i;t).x;val(a(i;t));f)
. . . | . . | | | . . . 30. source(l) = i
. . . | . . | | | . . . 31. map((λx.<l,x>) o (λms.2of(ms));filter(λms.
. . . | . . | | | . . .     mlnk(ms) = l;m(i;t)))
. . . | . . | | | . . .     =
. . . | . . | | | . . .     filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . . . 32. filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . | | | . . .     =
. . . | . . | | | . . .     filter(λms.mlnk(ms) = l;m(i;t))
. . . | . . | | | . . . ⊢  filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . | | | . . .    =
. . . | . . | | | . . .    map((λx.<l,x>) o (λms.2of(ms));filter(λms.
. . . | . . | | | . . .    mlnk(ms) = l;m(i;t)))  by Auto
. . . | . . | | |
. . . | . . | | .\
. . . | . . | | . .....wf .....
. . . | . . | | . 1. i : Id
. . . | . . | | . 2. k : Knd
. . . | . . | | . 3. l : IdLnk
. . . | . . | | . 4. ds : x:Id fp-> Type
. . . | . . | | . 5. da : a:Knd fp-> Type
. . . | . . | | . 6. f :
. . . | . . | | . |  (tg:Id
. . . | . . | | . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | | . |   (da(rcv(l; tg))?Void List)) List
. . . | . . | | . 7. source(l) = i
. . . | . . | | . 8. w : World
. . . | . . | | . 9. p : FairFifo
. . . | . . | | . 10. FairFifo
. . . | . . | | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | . |                      <ds
```

```
. . . | . . | | . |                        ,da
. . . | . . | | . |                        ,
. . . | . . | | . |                        ,
. . . | . . | | . |                        ,
. . . | . . | | . |                        ,<[<k,l>],λx.f>
. . . | . . | | . |                        ,
. . . | . . | | . |                        ,
. . . | . . | | . |                        ,·>
. . . | . . | | . |                    else  fi.da(rcv(l@0; tg))
. . . | . . | | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | . 14. ∀e:E.
. . . | . . | | . |   loc(e) = i
. . . | . . | | . |   ⇒
. . . | . . | | . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | . 15. ∀e:E.
. . . | . . | | . |   isrcv(e)
. . . | . . | | . |   ⇒
. . . | . . | | . |   lnk(e) = l
. . . | . . | | . |   ⇒
. . . | . . | | . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | . 17. ∀a:Action(i).
. . . | . . | | . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | . 18. ∀x:Id.
. . . | . . | | . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . | | . |                      2of(ds)(x)
. . . | . . | | . |                   else Top fi
. . . | . . | | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | . 20. j : Id
. . . | . . | | . 21. t : ℕ
. . . | . . | | . 22. ¬isnull(a(j;t))
. . . | . . | | . 23. True
. . . | . . | | . 24. j = i
. . . | . . | | . 25. kind(<j,t>) = k
. . . | . . | | . 26. ¬isnull(a(i;t))
. . . | . . | | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | . 28. ∀l@0:IdLnk.
. . . | . . | | . |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | . |    (<k,l>
. . . | . . | | . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | . |   ⇒
. . . | . . | | . |   map(λms.2of(ms);filter(λms.
. . . | . . | | . |   mlnk(ms) = l@0;m(i;t)))
. . . | . . | | . |   =
. . . | . . | | . |   if source(l@0) = i→
. . . | . . | | . |      tagged-list-messages(λx.
. . . | . . | | . |      s(i;t).x;val(a(i;t));f)
. . . | . . | | . |   else nil fi
. . . | . . | | . 29. withlnk(l;m(i;t))
. . . | . . | | . |   =
. . . | . . | | . |   tagged-list-messages(λx.s(i;t).x;val(a(i;t));f)
. . . | . . | | . 30. source(l) = i
. . . | . . | | . 31. z : (tg:Id×da(rcv(l; tg))?Top) List
```

```
. . . | . . | | . ⊢  (filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . | | . |   =
. . . | . . | | . |   map(λx.<l,x>;z))
. . . | . . | | . |  ∈ Prop  by Auto
. . . | . . | | . |
. . . | . . | | . |\
. . . | . . | | . | .....unproved Inclusion subgoal .....
. . . | . . | | . | 1. i : Id
. . . | . . | | . | 2. k : Knd
. . . | . . | | . | 3. l : IdLnk
. . . | . . | | . | 4. ds : x:Id fp-> Type
. . . | . . | | . | 5. da : a:Knd fp-> Type
. . . | . . | | . | 6. f :
. . . | . . | | . |    (tg:Id
. . . | . . | | . |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | . |     (da(rcv(l; tg))?Void List)) List
. . . | . . | | . | 7. source(l) = i
. . . | . . | | . | 8. w : World
. . . | . . | | . | 9. p : FairFifo
. . . | . . | | . | 10. FairFifo
. . . | . . | | . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | . |     (w.M
. . . | . . | | . |      (l@0
. . . | . . | | . |      ,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | . |                  <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . | | . |               else  fi.da(rcv(l@0; tg))
. . . | . . | | . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | . | 14. ∀e:E.
. . . | . . | | . |     loc(e) = i
. . . | . . | | . |     ⇒
. . . | . . | | . |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | . | 15. ∀e:E.
. . . | . . | | . |     isrcv(e)
. . . | . . | | . |     ⇒
. . . | . . | | . |     lnk(e) = l
. . . | . . | | . |     ⇒
. . . | . . | | . |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | . |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | . | 17. ∀a:Action(i).
. . . | . . | | . |     ¬isnull(a)
. . . | . . | | . |     ⇒
. . . | . . | | . |     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | . | 18. ∀x:Id.
. . . | . . | | . |     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | . |                                          ds))→
. . . | . . | | . |                        2of(ds)(x)
. . . | . . | | . |                     else Top fi
. . . | . . | | . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | . | 20. j : Id
. . . | . . | | . | 21. t : ℕ
. . . | . . | | . | 22. ¬isnull(a(j;t))
. . . | . . | | . | 23. True
. . . | . . | | . | 24. j = i
```

```
. . . | . . | | . | 25. kind(<j,t>) = k
. . . | . . | | . | 26. ¬isnull(a(i;t))
. . . | . . | | . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | . | 28. ∀l@0:IdLnk.
. . . | . . | | . |     (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | . |      (<k,l>
. . . | . . | | . |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | . |     ⇒
. . . | . . | | . |     map(λms.2of(ms);filter(λms.
. . . | . . | | . |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | . |     =
. . . | . . | | . |     if source(l@0) = i→
. . . | . . | | . |        tagged-list-messages(λx.
. . . | . . | | . |        s(i;t).x;val(a(i;t));f)
. . . | . . | | . |     else nil fi
. . . | . . | | . | 29. withlnk(l;m(i;t))
. . . | . . | | . |     =
. . . | . . | | . |     tagged-list-messages(λx.
. . . | . . | | . |     s(i;t).x;val(a(i;t));f)
. . . | . . | | . | 30. source(l) = i
. . . | . . | | . | 31. (tg:Id×da(rcv(l; tg))?Top) List
. . . | . . | | . | 32. m(j;t) = m(j;t)
. . . | . . | | . | 33. {m:Msg(w.M)| source(mlnk(m)) = j } List
. . . | . . | | . | 34. {m:Msg(w.M)| source(mlnk(m)) = j } List
. . . | . . | | . | 35. u : Msg(w.M)
. . . | . . | | . | 36. source(mlnk(u)) = j
. . . | . . | | . | ⊢  u ∈ Msg((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | . |   by SubsumeC {u:Msg(w.M)| source(mlnk(u)) = i }
. . . | . . | | . |      THEN
. . . | . . | | . |      AllHyps (λh.Unfold ‘es-Msg‘ h THEN Reduce h)
. . . | . . | | . |
. . . | . . | | .\

. . . | . . | | . . .....subterm .....
. . . | . . | | . . 1. i : Id
. . . | . . | | . . 2. k : Knd
. . . | . . | | . . 3. l : IdLnk
. . . | . . | | . . 4. ds : x:Id fp-> Type
. . . | . . | | . . 5. da : a:Knd fp-> Type
. . . | . . | | . . 6. f :
. . . | . . | | . .    (tg:Id
. . . | . . | | . .     ×State(ds)→ma-valtype(da; k)→
. . . | . . | | . .     (da(rcv(l; tg))?Void List)) List
. . . | . . | | . . 7. source(l) = i
. . . | . . | | . . 8. w : World
. . . | . . | | . . 9. p : FairFifo
. . . | . . | | . . 10. FairFifo
. . . | . . | | . . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | | . .     (w.M
. . . | . . | | . .      (l@0
. . . | . . | | . .      ,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | | . .                  <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . . | . . | | . .               else  fi.da(rcv(l@0; tg))
. . . | . . | | . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | | . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | | . . 14. ∀e:E.
```

```
. . . | . . | | . .     loc(e) = i
. . . | . . | | . .     ⇒
. . . | . . | | . .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | . . 15. ∀e:E.
. . . | . . | | . .     isrcv(e)
. . . | . . | | . .     ⇒
. . . | . . | | . .     lnk(e) = l
. . . | . . | | . .     ⇒
. . . | . . | | . .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | | . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | | . .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | . . 17. ∀a:Action(i).
. . . | . . | | . .     ¬isnull(a)
. . . | . . | | . .     ⇒
. . . | . . | | . .     (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | | . . 18. ∀x:Id.
. . . | . . | | . .     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(
. . . | . . | | . .                                          ds))→
. . . | . . | | . .                          2of(ds)(x)
. . . | . . | | . .                       else Top fi
. . . | . . | | . . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | | . . 20. j : Id
. . . | . . | | . . 21. t : ℕ
. . . | . . | | . . 22. ¬isnull(a(j;t))
. . . | . . | | . . 23. True
. . . | . . | | . . 24. j = i
. . . | . . | | . . 25. kind(<j,t>) = k
. . . | . . | | . . 26. ¬isnull(a(i;t))
. . . | . . | | . . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | | . . 28. ∀l@0:IdLnk.
. . . | . . | | . .     (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | | . .      (<k,l>
. . . | . . | | . .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | | . .     ⇒
. . . | . . | | . .     map(λms.2of(ms);filter(λms.
. . . | . . | | . .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | | . .     =
. . . | . . | | . .     if source(l@0) = i→
. . . | . . | | . .        tagged-list-messages(λx.
. . . | . . | | . .        s(i;t).x;val(a(i;t));f)
. . . | . . | | . .     else nil fi
. . . | . . | | . . 29. withlnk(l;m(i;t))
. . . | . . | | . .     =
. . . | . . | | . .     tagged-list-messages(λx.
. . . | . . | | . .     s(i;t).x;val(a(i;t));f)
. . . | . . | | . . 30. source(l) = i
. . . | . . | | . . 31. (tg:Id×da(rcv(l; tg))?Top) List
. . . | . . | | . . 32. x : tg:Id×da(rcv(l; tg))?Top
. . . | . . | | . . ⊢  <l,x> ∈ Msg((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | | . .   by Analyze -1 THEN Fold ‘msg‘ 0
. . . | . . | |
. . . | . . | .\
. . . | . . | . .....wf .....
. . . | . . | . 1. i : Id
. . . | . . | . 2. k : Knd
```

```
. . . | . . | . 3. l : IdLnk
. . . | . . | . 4. ds : x:Id fp-> Type
. . . | . . | . 5. da : a:Knd fp-> Type
. . . | . . | . 6. f :
. . . | . . | . |  (tg:Id
. . . | . . | . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . | . |   (da(rcv(l; tg))?Void List)) List
. . . | . . | . 7. source(l) = i
. . . | . . | . 8. w : World
. . . | . . | . 9. p : FairFifo
. . . | . . | . 10. FairFifo
. . . | . . | . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | . |                      <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . | . |                  else  fi.da(rcv(l@0; tg))
. . . | . . | . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | . 14. ∀e:E.
. . . | . . | . |   loc(e) = i
. . . | . . | . |   ⇒
. . . | . . | . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | . 15. ∀e:E.
. . . | . . | . |   isrcv(e)
. . . | . . | . |   ⇒
. . . | . . | . |   lnk(e) = l
. . . | . . | . |   ⇒
. . . | . . | . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | . 17. ∀a:Action(i).
. . . | . . | . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | . 18. ∀x:Id.
. . . | . . | . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . | . |                     2of(ds)(x)
. . . | . . | . |                  else Top fi
. . . | . . | . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | . 20. j : Id
. . . | . . | . 21. t : ℕ
. . . | . . | . 22. ¬isnull(a(j;t))
. . . | . . | . 23. True
. . . | . . | . 24. j = i
. . . | . . | . 25. kind(<j,t>) = k
. . . | . . | . 26. ¬isnull(a(i;t))
. . . | . . | . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | . 28. ∀l@0:IdLnk.
. . . | . . | . |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | . |    (<k,l>
. . . | . . | . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | . |   ⇒
. . . | . . | . |   map(λms.2of(ms);filter(λms.mlnk(ms) = l@0;m(i;t)))
. . . | . . | . |   =
. . . | . . | . |   if source(l@0) = i→
. . . | . . | . |      tagged-list-messages(λx.s(i;t).x;val(a(i;t));f)
. . . | . . | . |   else nil fi
. . . | . . | . 29. withlnk(l;m(i;t))
```

```
. . . | . . | . |   =
. . . | . . | . |   concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | . |                          ,x>;2of(tgf)
. . . | . . | . |                              ((λx.s(i;t).x)
. . . | . . | . |                              ,val(a(i;t))));f))
. . . | . . | . 30. source(l) = i
. . . | . . | . 31. z : (t:Id×da(rcv(l; t))?Top) List
. . . | . . | . ⊢  (filter(λms.mlnk(ms) = l;m(j;t)) = map(λx.<l,x>;z))
. . . | . . | . |  ∈ Prop  by Auto
. . . | . . | . |
. . . | . . | . |\
. . . | . . | . | .....unproved Inclusion subgoal .....
. . . | . . | . | 1. i : Id
. . . | . . | . | 2. k : Knd
. . . | . . | . | 3. l : IdLnk
. . . | . . | . | 4. ds : x:Id fp-> Type
. . . | . . | . | 5. da : a:Knd fp-> Type
. . . | . . | . | 6. f :
. . . | . . | . |    (tg:Id
. . . | . . | . |     ×State(ds)→ma-valtype(da; k)→
. . . | . . | . |     (da(rcv(l; tg))?Void List)) List
. . . | . . | . | 7. source(l) = i
. . . | . . | . | 8. w : World
. . . | . . | . | 9. p : FairFifo
. . . | . . | . | 10. FairFifo
. . . | . . | . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | . |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | . |                          <ds
. . . | . . | . |                          ,da
. . . | . . | . |                          ,
. . . | . . | . |                          ,
. . . | . . | . |                          ,
. . . | . . | . |                          ,<[<k,l>],λx.f>
. . . | . . | . |                          ,
. . . | . . | . |                          ,
. . . | . . | . |                          ,•>
. . . | . . | . |                      else  fi.da(rcv(l@0; tg))
. . . | . . | . | 12. (eqof(IdDeq)(i,i)) ~ true₂
. . . | . . | . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | . | 14. ∀e:E.
. . . | . . | . |     loc(e) = i
. . . | . . | . |     ⇒
. . . | . . | . |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | . | 15. ∀e:E.
. . . | . . | . |     isrcv(e)
. . . | . . | . |     ⇒
. . . | . . | . |     lnk(e) = l
. . . | . . | . |     ⇒
. . . | . . | . |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | . |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | . | 17. ∀a:Action(i).
. . . | . . | . |     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | . | 18. ∀x:Id.
. . . | . . | . |     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
```

```
. . . | . . | . |                        2of(ds)(x)
. . . | . . | . |                      else Top fi
. . . | . . | . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . | . | 20. j : Id
. . . | . . | . | 21. t : ℕ
. . . | . . | . | 22. ¬isnull(a(j;t))
. . . | . . | . | 23. True
. . . | . . | . | 24. j = i
. . . | . . | . | 25. kind(<j,t>) = k
. . . | . . | . | 26. ¬isnull(a(i;t))
. . . | . . | . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | . | 28. ∀l@0:IdLnk.
. . . | . . | . |     (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | . |      (<k,l>
. . . | . . | . |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | . |     ⇒
. . . | . . | . |     map(λms.2of(ms);filter(λms.
. . . | . . | . |     mlnk(ms) = l@0;m(i;t)))
. . . | . . | . |     =
. . . | . . | . |     if source(l@0) = i⟶
. . . | . . | . |        tagged-list-messages(λx.
. . . | . . | . |        s(i;t).x;val(a(i;t));f)
. . . | . . | . |     else nil fi
. . . | . . | . | 29. withlnk(l;m(i;t))
. . . | . . | . |     =
. . . | . . | . |     concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | . |                              ,x>;2of(tgf)
. . . | . . | . |                                  ((λx.s(i;t).x)
. . . | . . | . |                                  ,val(a(i;t))));f))
. . . | . . | . | 30. source(l) = i
. . . | . . | . | 31. (t:Id×da(rcv(l; t))?Top) List
. . . | . . | . | 32. m(j;t) = m(j;t)
. . . | . . | . | 33. {m:Msg(w.M)| source(mlnk(m)) = j } List
. . . | . . | . | 34. {m:Msg(w.M)| source(mlnk(m)) = j } List
. . . | . . | . | 35. u : Msg(w.M)
. . . | . . | . | 36. source(mlnk(u)) = j
. . . | . . | . | ⊢ u ∈ Msg((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | .   by SubsumeC {u:Msg(w.M)| source(mlnk(u)) = i }
. . . | . . | .      THEN
. . . | . . | .      AllHyps (λh.Unfold `es-Msg` h THEN Reduce h)
. . . | . . | .
. . . | . . | . .\
. . . | . . | . . .....subterm .....
. . . | . . | . . 1. i : Id
. . . | . . | . . 2. k : Knd
. . . | . . | . . 3. l : IdLnk
. . . | . . | . . 4. ds : x:Id fp-> Type
. . . | . . | . . 5. da : a:Knd fp-> Type
. . . | . . | . . 6. f :
. . . | . . | . .    (tg:Id
. . . | . . | . .     ×State(ds)⟶ma-valtype(da; k)⟶
. . . | . . | . .     (da(rcv(l; tg))?Void List)) List
. . . | . . | . . 7. source(l) = i
. . . | . . | . . 8. w : World
. . . | . . | . . 9. p : FairFifo
```

```
. . . | . . | . . 10. FairFifo
. . . | . . | . . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . | . .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . | . .                         <ds
. . . | . . | . .                         ,da
. . . | . . | . .                         ,
. . . | . . | . .                         ,
. . . | . . | . .                         ,
. . . | . . | . .                         ,<[<k,l>],λx.f>
. . . | . . | . .                         ,
. . . | . . | . .                         ,
. . . | . . | . .                         ,·>
. . . | . . | . .                       else  fi.da(rcv(l@0; tg))
. . . | . . | . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . | . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . | . . 14. ∀e:E.
. . . | . . | . .     loc(e) = i
. . . | . . | . .     ⇒
. . . | . . | . .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | . . 15. ∀e:E.
. . . | . . | . .     isrcv(e)
. . . | . . | . .     ⇒
. . . | . . | . .     lnk(e) = l
. . . | . . | . .     ⇒
. . . | . . | . .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . | . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . | . .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | . . 17. ∀a:Action(i).
. . . | . . | . .     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . | . . 18. ∀x:Id.
. . . | . . | . .     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . | . .                         2of(ds)(x)
. . . | . . | . .                       else Top fi
. . . | . . | . . 19. IdDeq ∈ EqDecider(Id)
. . . | . . | . . 20. j : Id
. . . | . . | . . 21. t : ℕ
. . . | . . | . . 22. ¬isnull(a(j;t))
. . . | . . | . . 23. True
. . . | . . | . . 24. j = i
. . . | . . | . . 25. kind(<j,t>) = k
. . . | . . | . . 26. ¬isnull(a(i;t))
. . . | . . | . . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . | . . 28. ∀l@0:IdLnk.
. . . | . . | . .     (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . | . .      (<k,l>
. . . | . . | . .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . | . .     ⇒
. . . | . . | . .     map(λms.2of(ms);filter(λms.
. . . | . . | . .     mlnk(ms) = l@0;m(i;t)))
. . . | . . | . .     =
. . . | . . | . .     if source(l@0) = i→
. . . | . . | . .        tagged-list-messages(λx.
. . . | . . | . .        s(i;t).x;val(a(i;t));f)
. . . | . . | . .     else nil fi
. . . | . . | . . 29. withlnk(l;m(i;t))
```

```
. . . | . . | . .     =
. . . | . . | . .     concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . | . .                               ,x>;2of(tgf)
. . . | . . | . .                                   ((λx.s(i;t).x)
. . . | . . | . .                                   ,val(a(i;t))));f))
. . . | . . | . . 30. source(l) = i
. . . | . . | . . 31. (t:Id×da(rcv(l; t))?Top) List
. . . | . . | . . 32. x : t:Id×da(rcv(l; t))?Top
. . . | . . | . . ⊢  <l,x> ∈ Msg((λl,tg. da(rcv(l; tg))?Top))
. . . | . . | . .   by Analyze -1 THEN Fold ‘msg‘ 0
. . . | . . |
. . . | . . \
. . . | . . . .....falsecase .....
. . . | . . . 1. i : Id
. . . | . . . 2. k : Knd
. . . | . . . 3. l : IdLnk
. . . | . . . 4. ds : x:Id fp-> Type
. . . | . . . 5. da : a:Knd fp-> Type
. . . | . . . 6. f :
. . . | . . . |  (tg:Id
. . . | . . . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . . |   (da(rcv(l; tg))?Void List)) List
. . . | . . . 7. source(l) = i
. . . | . . . 8. w : World
. . . | . . . 9. p : FairFifo
. . . | . . . 10. FairFifo
. . . | . . . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . . |                      <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . . |                   else  fi.da(rcv(l@0; tg))
. . . | . . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . . 14. ∀e:E.
. . . | . . . |   loc(e) = i
. . . | . . . |   ⇒
. . . | . . . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . . 15. ∀e:E.
. . . | . . . |   isrcv(e)
. . . | . . . |   ⇒
. . . | . . . |   lnk(e) = l
. . . | . . . |   ⇒
. . . | . . . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . . 17. ∀a:Action(i).
. . . | . . . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . . 18. ∀x:Id.
. . . | . . . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . . |                     2of(ds)(x)
. . . | . . . |                  else Top fi
. . . | . . . 19. IdDeq ∈ EqDecider(Id)
. . . | . . . 20. j : Id
. . . | . . . 21. t : ℕ
. . . | . . . 22. ¬isnull(a(j;t))
. . . | . . . 23. True
```

```
. . . | . . . 24. j = i
. . . | . . . 25. kind(<j,t>) = k
. . . | . . . 26. ¬isnull(a(i;t))
. . . | . . . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . . 28. ∀l@0:IdLnk.
. . . | . . . |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . . |    (<k,l>
. . . | . . . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . . |   ⇒
. . . | . . . |   withlnk(l@0;m(i;t))
. . . | . . . |   =
. . . | . . . |   if source(l@0) = i→
. . . | . . . |      concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . . |                              ,x>;2of(tgf)
. . . | . . . |                                  ((λx.s(i;t).x)
. . . | . . . |                                  ,val(a(i;t))));f))
. . . | . . . |   else nil fi
. . . | . . . 29. withlnk(l;m(i;t)) = nil
. . . | . . . 30. ¬source(l) = i
. . . | . . . ⊢  filter(λms.mlnk(ms) = l;m(j;t))
. . . | . . . |  =
. . . | . . . |  tagged-messages(l;λz.s(j;t).z;val(a(j;t));f)
. . . | . . . | by Subst (filter(λms.mlnk(ms) = l;m(j;t)) ∼ nil) 0
. . . | . . . |
. . . | . . . |\
. . . | . . . | .....equality .....
. . . | . . . | 1. i : Id
. . . | . . . | 2. k : Knd
. . . | . . . | 3. l : IdLnk
. . . | . . . | 4. ds : x:Id fp-> Type
. . . | . . . | 5. da : a:Knd fp-> Type
. . . | . . . | 6. f :
. . . | . . . | |  (tg:Id
. . . | . . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . . | 7. source(l) = i
. . . | . . . | 8. w : World
. . . | . . . | 9. p : FairFifo
. . . | . . . | 10. FairFifo
. . . | . . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . . | |                        <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . . | |                    else  fi.da(rcv(l@0; tg))
. . . | . . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . . | 14. ∀e:E.
. . . | . . . | |   loc(e) = i
. . . | . . . | |   ⇒
. . . | . . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . . | 15. ∀e:E.
. . . | . . . | |   isrcv(e)
. . . | . . . | |   ⇒
. . . | . . . | |   lnk(e) = l
. . . | . . . | |   ⇒
. . . | . . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
```

```
. . . | . . . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . . | 17. ∀a:Action(i).
. . . | . . . | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . . | 18. ∀x:Id.
. . . | . . . | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . . | |                     2of(ds)(x)
. . . | . . . | |                  else Top fi
. . . | . . . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . . | 20. j : Id
. . . | . . . | 21. t : ℕ
. . . | . . . | 22. ¬isnull(a(j;t))
. . . | . . . | 23. True
. . . | . . . | 24. j = i
. . . | . . . | 25. kind(<j,t>) = k
. . . | . . . | 26. ¬isnull(a(i;t))
. . . | . . . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . . | 28. ∀l@0:IdLnk.
. . . | . . . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . . | |    (<k,l>
. . . | . . . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . . | |   ⇒
. . . | . . . | |   withlnk(l@0;m(i;t))
. . . | . . . | |   =
. . . | . . . | |   if source(l@0) = i→
. . . | . . . | |      concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . . | |                              ,x>;2of(tgf)
. . . | . . . | |                                  ((λx.s(i;t).x)
. . . | . . . | |                                  ,val(a(i;t))));f))
. . . | . . . | |   else nil fi
. . . | . . . | 29. withlnk(l;m(i;t)) = nil
. . . | . . . | 30. ¬source(l) = i
. . . | . . . | ⊢  filter(λms.mlnk(ms) = l;m(j;t)) ∼ nil
. . . | . . . | | by Unfold ‘w-withlnk‘ -2 THEN Unfold ‘mapfilter‘ -2
. . . | . . . | |
. . . | . . . | 1. i : Id
. . . | . . . | 2. k : Knd
. . . | . . . | 3. l : IdLnk
. . . | . . . | 4. ds : x:Id fp-> Type
. . . | . . . | 5. da : a:Knd fp-> Type
. . . | . . . | 6. f :
. . . | . . . | |  (tg:Id
. . . | . . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . . | 7. source(l) = i
. . . | . . . | 8. w : World
. . . | . . . | 9. p : FairFifo
. . . | . . . | 10. FairFifo
. . . | . . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . . | |                       <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . . | |                    else  fi.da(rcv(l@0; tg))
. . . | . . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . . | 14. ∀e:E.
```

```
. . . | . . . | |   loc(e) = i
. . . | . . . | |   ⇒
. . . | . . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . . | 15. ∀e:E.
. . . | . . . | |   isrcv(e)
. . . | . . . | |   ⇒
. . . | . . . | |   lnk(e) = l
. . . | . . . | |   ⇒
. . . | . . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . . | 17. ∀a:Action(i).
. . . | . . . | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . . | 18. ∀x:Id.
. . . | . . . | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . . | |                      2of(ds)(x)
. . . | . . . | |                  else Top fi
. . . | . . . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . . | 20. j : Id
. . . | . . . | 21. t : ℕ
. . . | . . . | 22. ¬isnull(a(j;t))
. . . | . . . | 23. True
. . . | . . . | 24. j = i
. . . | . . . | 25. kind(<j,t>) = k
. . . | . . . | 26. ¬isnull(a(i;t))
. . . | . . . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . . | 28. ∀l@0:IdLnk.
. . . | . . . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . . | |    (<k,l>
. . . | . . . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . . | |   ⇒
. . . | . . . | |   withlnk(l@0;m(i;t))
. . . | . . . | |   =
. . . | . . . | |   if source(l@0) = i→
. . . | . . . | |      concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . . | |                              ,x>;2of(tgf)
. . . | . . . | |                                  ((λx.s(i;t).x)
. . . | . . . | |                                  ,val(a(i;t))));f))
. . . | . . . | |   else nil fi
. . . | . . . | 29. map(λms.2of(ms);filter(λms.mlnk(ms) = l;m(i;t)))
. . . | . . . | |   =
. . . | . . . | |   nil
. . . | . . . | 30. ¬source(l) = i
. . . | . . . | ⊢  filter(λms.mlnk(ms) = l;m(j;t)) ~ nil
. . . | . . . | | by RW0
. . . | . . . | |    Thm* ∀f:(A→B), l:A List.
. . . | . . . | |         map(f;l) = nil ⟺ l = nil
. . . | . . . | |    {map_is_nil}
. . . | . . . | |    -2
. . . | . . . | |
. . . | . . . | 1. i : Id
. . . | . . . | 2. k : Knd
. . . | . . . | 3. l : IdLnk
. . . | . . . | 4. ds : x:Id fp-> Type
. . . | . . . | 5. da : a:Knd fp-> Type
```

```
. . . | . . . | 6. f :
. . . | . . . | |  (tg:Id
. . . | . . . | |   ×State(ds)→ma-valtype(da; k)→
. . . | . . . | |   (da(rcv(l; tg))?Void List)) List
. . . | . . . | 7. source(l) = i
. . . | . . . | 8. w : World
. . . | . . . | 9. p : FairFifo
. . . | . . . | 10. FairFifo
. . . | . . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . . | |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . . | |                      <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . . | |                   else  fi.da(rcv(l@0; tg))
. . . | . . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . . | 14. ∀e:E.
. . . | . . . | |   loc(e) = i
. . . | . . . | |   ⇒
. . . | . . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . . | 15. ∀e:E.
. . . | . . . | |   isrcv(e)
. . . | . . . | |   ⇒
. . . | . . . | |   lnk(e) = l
. . . | . . . | |   ⇒
. . . | . . . | |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . . | |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . . | 17. ∀a:Action(i).
. . . | . . . | |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . . | 18. ∀x:Id.
. . . | . . . | |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . . | |                     2of(ds)(x)
. . . | . . . | |                  else Top fi
. . . | . . . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . . | 20. j : Id
. . . | . . . | 21. t : ℕ
. . . | . . . | 22. ¬isnull(a(j;t))
. . . | . . . | 23. True
. . . | . . . | 24. j = i
. . . | . . . | 25. kind(<j,t>) = k
. . . | . . . | 26. ¬isnull(a(i;t))
. . . | . . . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . . | 28. ∀l@0:IdLnk.
. . . | . . . | |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . . | |    (<k,l>
. . . | . . . | |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . . | |   ⇒
. . . | . . . | |   withlnk(l@0;m(i;t))
. . . | . . . | |   =
. . . | . . . | |   if source(l@0) = i→
. . . | . . . | |      concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . . | |                              ,x>;2of(tgf)
. . . | . . . | |                                  ((λx.s(i;t).x)
. . . | . . . | |                                  ,val(a(i;t))));f))
. . . | . . . | |   else nil fi
. . . | . . . | 29. filter(λms.mlnk(ms) = l;m(i;t)) = nil
```

```
. . . | . . . | 30. ¬source(l) = i
. . . | . . . | ⊢  filter(λms.mlnk(ms) = l;m(j;t)) ~ nil
. . . | . . . | | by AssertBY (filter(λms.mlnk(ms) = l;m(j;t)) = nil)
. . . | . . . | |    (NthHypEq -2 THEN Repeat Analyze)
. . . | . . . | |    THEN
. . . | . . . | |    MoveToConcl -1
. . . | . . . | |
. . . | . . . | 1. i : Id
. . . | . . . | 2. k : Knd
. . . | . . . | 3. l : IdLnk
. . . | . . . | 4. ds : x:Id fp-> Type
. . . | . . . | 5. da : a:Knd fp-> Type
. . . | . . . | 6. f :
. . . | . . . |    (tg:Id
. . . | . . . |     ×State(ds)→ma-valtype(da; k)→
. . . | . . . |     (da(rcv(l; tg))?Void List)) List
. . . | . . . | 7. source(l) = i
. . . | . . . | 8. w : World
. . . | . . . | 9. p : FairFifo
. . . | . . . | 10. FairFifo
. . . | . . . | 11. ∀l@0:IdLnk, tg:Id.
. . . | . . . |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . . |                         <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . . |                       else  fi.da(rcv(l@0; tg))
. . . | . . . | 12. (eqof(IdDeq)(i,i)) ~ true₂
. . . | . . . | 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . . | 14. ∀e:E.
. . . | . . . |     loc(e) = i
. . . | . . . |     ⇒
. . . | . . . |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . . | 15. ∀e:E.
. . . | . . . |     isrcv(e)
. . . | . . . |     ⇒
. . . | . . . |     lnk(e) = l
. . . | . . . |     ⇒
. . . | . . . |     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . . | 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . . |     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . . | 17. ∀a:Action(i).
. . . | . . . |     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . . | 18. ∀x:Id.
. . . | . . . |     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . . |                        2of(ds)(x)
. . . | . . . |                     else Top fi
. . . | . . . | 19. IdDeq ∈ EqDecider(Id)
. . . | . . . | 20. j : Id
. . . | . . . | 21. t : ℕ
. . . | . . . | 22. ¬isnull(a(j;t))
. . . | . . . | 23. True
. . . | . . . | 24. j = i
. . . | . . . | 25. kind(<j,t>) = k
. . . | . . . | 26. ¬isnull(a(i;t))
. . . | . . . | 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . . | 28. ∀l@0:IdLnk.
. . . | . . . |     (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
```

```
. . . | . . . |      (<k,l>
. . . | . . . |      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . . |     ⇒
. . . | . . . |     withlnk(l@0;m(i;t))
. . . | . . . |     =
. . . | . . . |     if source(l@0) = i→
. . . | . . . |        concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . . |                                ,x>;2of(tgf)
. . . | . . . |                                    ((λx.s(i;t).x)
. . . | . . . |                                    ,val(a(i;t))));f))
. . . | . . . |     else nil fi
. . . | . . . | 29. filter(λms.mlnk(ms) = l;m(i;t)) = nil
. . . | . . . | 30. ¬source(l) = i
. . . | . . . | ⊢  filter(λms.mlnk(ms) = l;m(j;t)) = nil
. . . | . . . |    ⇒
. . . | . . . |    (filter(λms.mlnk(ms) = l;m(j;t)) ∼ nil)
. . . | . . . |   by GenConclAtAddr [1;2] THEN Analyze -2
. . . | . . . |      THEN
. . . | . . . |      Analyze 0
. . . | . . . |      THEN
. . . | . . . |      Try Trivial
. . . | . . . |
. . . | . . . .\
. . . | . . . . 1. i : Id
. . . | . . . . 2. k : Knd
. . . | . . . . 3. l : IdLnk
. . . | . . . . 4. ds : x:Id fp-> Type
. . . | . . . . 5. da : a:Knd fp-> Type
. . . | . . . . 6. f :
. . . | . . . . |  (tg:Id
. . . | . . . . |   ×State(ds)→ma-valtype(da; k)→
. . . | . . . . |   (da(rcv(l; tg))?Void List)) List
. . . | . . . . 7. source(l) = i
. . . | . . . . 8. w : World
. . . | . . . . 9. p : FairFifo
. . . | . . . . 10. FairFifo
. . . | . . . . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . . . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . . . |                        <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . . . |                   else  fi.da(rcv(l@0; tg))
. . . | . . . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . . . 14. ∀e:E.
. . . | . . . . |   loc(e) = i
. . . | . . . . |   ⇒
. . . | . . . . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . . . 15. ∀e:E.
. . . | . . . . |   isrcv(e)
. . . | . . . . |   ⇒
. . . | . . . . |   lnk(e) = l
. . . | . . . . |   ⇒
. . . | . . . . |   (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . . . |   ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . . . 17. ∀a:Action(i).
```

```
. . . | . . . . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . . . 18. ∀x:Id.
. . . | . . . . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . . . |                     2of(ds)(x)
. . . | . . . . |                  else Top fi
. . . | . . . . 19. IdDeq ∈ EqDecider(Id)
. . . | . . . . 20. j : Id
. . . | . . . . 21. t : ℕ
. . . | . . . . 22. ¬isnull(a(j;t))
. . . | . . . . 23. True
. . . | . . . . 24. j = i
. . . | . . . . 25. kind(<j,t>) = k
. . . | . . . . 26. ¬isnull(a(i;t))
. . . | . . . . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . . . 28. ∀l@0:IdLnk.
. . . | . . . . |   (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . . . |    (<k,l>
. . . | . . . . |    ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . . . |   ⇒
. . . | . . . . |   withlnk(l@0;m(i;t))
. . . | . . . . |   =
. . . | . . . . |   if source(l@0) = i→
. . . | . . . . |      concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . . . |                               ,x>;2of(tgf)
. . . | . . . . |                                   ((λx.s(i;t).x)
. . . | . . . . |                                   ,val(a(i;t))));f))
. . . | . . . . |   else nil fi
. . . | . . . . 29. withlnk(l;m(i;t)) = nil
. . . | . . . . 30. ¬source(l) = i
. . . | . . . . ⊢  nil = tagged-messages(l;λz.s(j;t).z;val(a(j;t));f)
. . . | . . . . | by GenConcl (val(a(j;t)) = v)
. . . | . . . . |
. . . | . . . . 1. i : Id
. . . | . . . . 2. k : Knd
. . . | . . . . 3. l : IdLnk
. . . | . . . . 4. ds : x:Id fp-> Type
. . . | . . . . 5. da : a:Knd fp-> Type
. . . | . . . . 6. f :
. . . | . . . .    (tg:Id
. . . | . . . .     ×State(ds)→ma-valtype(da; k)→
. . . | . . . .     (da(rcv(l; tg))?Void List)) List
. . . | . . . . 7. source(l) = i
. . . | . . . . 8. w : World
. . . | . . . . 9. p : FairFifo
. . . | . . . . 10. FairFifo
. . . | . . . . 11. ∀l@0:IdLnk, tg:Id.
. . . | . . . .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . | . . . .                         <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . | . . . .                      else  fi.da(rcv(l@0; tg))
. . . | . . . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . | . . . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . | . . . . 14. ∀e:E.
. . . | . . . .     loc(e) = i
. . . | . . . .     ⇒
. . . | . . . .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
```

```
. . . | . . . . 15. ∀e:E.
. . . | . . . .     isrcv(e)
. . . | . . . .     ⇒
. . . | . . . .     lnk(e) = l
. . . | . . . .     ⇒
. . . | . . . .     (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . | . . . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . | . . . .     ((λl,tg. da(rcv(l; tg))?Top))
. . . | . . . . 17. ∀a:Action(i).
. . . | . . . .     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . | . . . . 18. ∀x:Id.
. . . | . . . .     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . | . . . .                         2of(ds)(x)
. . . | . . . .                     else Top fi
. . . | . . . . 19. IdDeq ∈ EqDecider(Id)
. . . | . . . . 20. j : Id
. . . | . . . . 21. t : ℕ
. . . | . . . . 22. ¬isnull(a(j;t))
. . . | . . . . 23. True
. . . | . . . . 24. j = i
. . . | . . . . 25. kind(<j,t>) = k
. . . | . . . . 26. ¬isnull(a(i;t))
. . . | . . . . 27. (λx.s(i;t).x) ∈ State(ds)
. . . | . . . . 28. ∀l@0:IdLnk.
. . . | . . . .     (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . | . . . .      (<k,l>
. . . | . . . .      ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . | . . . .     ⇒
. . . | . . . .     withlnk(l@0;m(i;t))
. . . | . . . .     =
. . . | . . . .     if source(l@0) = i→
. . . | . . . .        concat(map(λtgf.map(λx.<1of(tgf)
. . . | . . . .                                ,x>;2of(tgf)
. . . | . . . .                                    ((λx.s(i;t).x)
. . . | . . . .                                    ,val(a(i;t))));f))
. . . | . . . .     else nil fi
. . . | . . . . 29. withlnk(l;m(i;t)) = nil
. . . | . . . . 30. ¬source(l) = i
. . . | . . . . 31. v : ma-valtype(da; k)
. . . | . . . . 32. val(a(j;t)) = v
. . . | . . . . ⊢  nil = tagged-messages(l;λz.s(j;t).z;v;f)
. . . | . . . .   by Analyze -3
. . . |
. . . .\
. . . . .....wf .....
. . . . 1. i : Id
. . . . 2. k : Knd
. . . . 3. l : IdLnk
. . . . 4. ds : x:Id fp-> Type
. . . . 5. da : a:Knd fp-> Type
. . . . 6. f :
. . . . |  (tg:Id
. . . . |   ×State(ds)→ma-valtype(da; k)→
. . . . |   (da(rcv(l; tg))?Void List)) List
. . . . 7. source(l) = i
```

```
. . . . 8. w : World
. . . . 9. p : FairFifo
. . . . 10. FairFifo
. . . . 11. ∀l@0:IdLnk, tg:Id.
. . . . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . . |                        <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . . |                    else  fi.da(rcv(l@0; tg))
. . . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . . 13. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . . . |   & (∀e:E.
. . . . |      loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . . |   & (∀e:E.
. . . . |      isrcv(e)
. . . . |      ⇒
. . . . |      lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . . |   & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . . |      ((λl,tg. da(rcv(l; tg))?Top)))
. . . . 14. ∀t:ℕ.
. . . . |   ¬isnull(a(i;t))
. . . . |   ⇒
. . . . |   (islocal(kind(a(i;t)))
. . . . |    ⇒
. . . . |    deq-member(IdDeq;act(kind(a(i;t)));1of())
. . . . |    ⇒
. . . . |    2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . . . |   & (∀x:Id.
. . . . |      deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<kind(...)
. . . . |                                                    ,x>;1of())
. . . . |      ⇒
. . . . |      s(i;t+1).x
. . . . |      =
. . . . |      2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . . . |   & (∀l@0:IdLnk.
. . . . |      (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . . |       (<k,l>
. . . . |       ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . . |      ⇒
. . . . |      withlnk(l@0;m(i;t))
. . . . |      =
. . . . |      if source(l@0) = i→
. . . . |         concat(map(λtgf.map(λx.<1of(tgf)
. . . . |                                  ,x>;2of(tgf)
. . . . |                                      ((λx.s(i;t).x)
. . . . |                                      ,val(a(i;t))));f))
. . . . |      else nil fi)
. . . . |   & (∀x:Id.
. . . . |      ¬(deq-member(IdDeq;x;1of())
. . . . |        ⇒
. . . . |        deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . . . |      ⇒
. . . . |      s(i;t).x = s(i;t+1).x)
. . . . |   & (∀l@0:IdLnk, tg:Id.
. . . . |      ¬(deq-member(product-deq(IdLnk;Id;...;IdDeq);...;1of())
. . . . |        ⇒
. . . . |        deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
```

```
. . . . |      ⇒
. . . . |      w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
. . . . 15. ∀a:Action(i).
. . . . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . . 16. ∀x:Id.
. . . . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→ 2of(ds)(x)
. . . . |                   else Top fi
. . . . 17. IdDeq ∈ EqDecider(Id)
. . . . ⊢  E ∈ Type  by Assert ((λz.(z initially i)) ∈ State(ds))
. . . . |
. . . . |\
. . . . | .....assertion .....
. . . . | 1. i : Id
. . . . | 2. k : Knd
. . . . | 3. l : IdLnk
. . . . | 4. ds : x:Id fp-> Type
. . . . | 5. da : a:Knd fp-> Type
. . . . | 6. f :
. . . . |    (tg:Id
. . . . |     ×State(ds)→ma-valtype(da; k)→
. . . . |     (da(rcv(l; tg))?Void List)) List
. . . . | 7. source(l) = i
. . . . | 8. w : World
. . . . | 9. p : FairFifo
. . . . | 10. FairFifo
. . . . | 11. ∀l@0:IdLnk, tg:Id.
. . . . |     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . . |                          <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . . |                       else  fi.da(rcv(l@0; tg))
. . . . | 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . . | 13. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . . . |     & (∀e:E.
. . . . |        loc(e) = i
. . . . |        ⇒
. . . . |        (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . . |     & (∀e:E.
. . . . |        isrcv(e)
. . . . |        ⇒
. . . . |        lnk(e) = l
. . . . |        ⇒
. . . . |        (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . . |     & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . . |        ((λl,tg. da(rcv(l; tg))?Top)))
. . . . | 14. ∀t:ℕ.
. . . . |     ¬isnull(a(i;t))
. . . . |     ⇒
. . . . |     (islocal(kind(a(i;t)))
. . . . |      ⇒
. . . . |      deq-member(IdDeq;act(kind(a(i;t)));1of())
. . . . |      ⇒
. . . . |      2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . . . |     & (∀x:Id.
. . . . |        deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<...
. . . . |                                                    ,x>;1of(
. . . . |                                                        ))
```

```
. . . . |        ⇒
. . . . |        s(i;t+1).x
. . . . |        =
. . . . |        2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . . . |     & (∀l@0:IdLnk.
. . . . |        (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . . |         (<k,l>
. . . . |         ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . . |        ⇒
. . . . |        withlnk(l@0;m(i;t))
. . . . |        =
. . . . |        if source(l@0) = i→
. . . . |           concat(map(λtgf.map(λx.<1of(tgf)
. . . . |                                ,x>;2of(tgf)
. . . . |                                    ((λx.s(i;t).x)
. . . . |                                    ,val(a(i;t))));f))
. . . . |        else nil fi)
. . . . |     & (∀x:Id.
. . . . |        ¬(deq-member(IdDeq;x;1of())
. . . . |          ⇒
. . . . |          deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . . . |        ⇒
. . . . |        s(i;t).x = s(i;t+1).x)
. . . . |     & (∀l@0:IdLnk, tg:Id.
. . . . |        ¬(deq-member(product-deq(IdLnk;Id;...;IdDeq);...;1of(
. . . . |                                                        ))
. . . . |          ⇒
. . . . |          deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
. . . . |        ⇒
. . . . |        w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
. . . . | 15. ∀a:Action(i).
. . . . |     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . . | 16. ∀x:Id.
. . . . |     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . . |                        2of(ds)(x)
. . . . |                     else Top fi
. . . . | 17. IdDeq ∈ EqDecider(Id)
. . . . | ⊢  (λz.(z initially i)) ∈ State(ds)
. . . . |   by Unfolds [‘w-initially‘;‘ma-state‘] 0 THEN Analyze
. . . . |      THEN
. . . . |      DoSubsume
. . . . |      THEN
. . . . |      Try BackThruSomeHyp
. . . . |      THEN
. . . . |      Try D_ES_Subtype
. . . . |
. . . . .\
. . . . . 1. i : Id
. . . . . 2. k : Knd
. . . . . 3. l : IdLnk
. . . . . 4. ds : x:Id fp-> Type
. . . . . 5. da : a:Knd fp-> Type
. . . . . 6. f :
. . . . . |  (tg:Id
. . . . . |   ×State(ds)→ma-valtype(da; k)→
```

```
. . . . . |   (da(rcv(l; tg))?Void List)) List
. . . . . 7. source(l) = i
. . . . . 8. w : World
. . . . . 9. p : FairFifo
. . . . . 10. FairFifo
. . . . . 11. ∀l@0:IdLnk, tg:Id.
. . . . . |   (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . . . |                        <ds,da,,,,<[<k,l>],λx.f>,,,·>
. . . . . |                    else  fi.da(rcv(l@0; tg))
. . . . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . . . 13. (∀x:Id. vartype(i;x) ⊆r ds(x)?Top)
. . . . . |   & (∀e:E.
. . . . . |      loc(e) = i
. . . . . |      ⇒
. . . . . |      (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . . . |   & (∀e:E.
. . . . . |      isrcv(e)
. . . . . |      ⇒
. . . . . |      lnk(e) = l
. . . . . |      ⇒
. . . . . |      (valtype(e) ⊆r ma-valtype(da; kind(e))))
. . . . . |   & ({m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . . . |      ((λl,tg. da(rcv(l; tg))?Top)))
. . . . . 14. ∀t:ℕ.
. . . . . |   ¬isnull(a(i;t))
. . . . . |   ⇒
. . . . . |   (islocal(kind(a(i;t)))
. . . . . |    ⇒
. . . . . |    deq-member(IdDeq;act(kind(a(i;t)));1of())
. . . . . |    ⇒
. . . . . |    2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . . . . |   & (∀x:Id.
. . . . . |      deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<...
. . . . . |                                                  ,x>;1of(
. . . . . |                                                      ))
. . . . . |      ⇒
. . . . . |      s(i;t+1).x
. . . . . |      =
. . . . . |      2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . . . . |   & (∀l@0:IdLnk.
. . . . . |      (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . . . |       (<k,l>
. . . . . |       ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . . . |      ⇒
. . . . . |      withlnk(l@0;m(i;t))
. . . . . |      =
. . . . . |      if source(l@0) = i→
. . . . . |         concat(map(λtgf.map(λx.<1of(tgf)
. . . . . |                                  ,x>;2of(tgf)
. . . . . |                                      ((λx.s(i;t).x)
. . . . . |                                      ,val(a(i;t))));f))
. . . . . |      else nil fi)
. . . . . |   & (∀x:Id.
. . . . . |      ¬(deq-member(IdDeq;x;1of())
. . . . . |        ⇒
```

```
. . . . . |       deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . . . . |     ⇒
. . . . . |     s(i;t).x = s(i;t+1).x)
. . . . . |   & (∀l@0:IdLnk, tg:Id.
. . . . . |     ¬(deq-member(product-deq(IdLnk;Id;...;IdDeq);...;1of(
. . . . . |                                                      ))
. . . . . |       ⇒
. . . . . |       deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
. . . . . |     ⇒
. . . . . |     w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
. . . . . 15. ∀a:Action(i).
. . . . . |   ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . . . 16. ∀x:Id.
. . . . . |   vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . . . |                      2of(ds)(x)
. . . . . |                  else Top fi
. . . . . 17. IdDeq ∈ EqDecider(Id)
. . . . . 18. (λz.(z initially i)) ∈ State(ds)
. . . . . ⊢  E ∈ Type
. . . . . | by AllHyps
. . . . . |    (λh.
. . . . . |     InstHyp [a;0] h
. . . . . |     THENA
. . . . . |     (Auto THEN All (Unfold `dstype`) THEN All Reduce
. . . . . |      THEN
. . . . . |      Trivial))
. . . . . |    THEN
. . . . . |    ExRepD
. . . . . |    THEN
. . . . . |    SplitOrHyps
. . . . . |
. . . . . 1. i : Id
. . . . . 2. k : Knd
. . . . . 3. l : IdLnk
. . . . . 4. ds : x:Id fp-> Type
. . . . . 5. da : a:Knd fp-> Type
. . . . . 6. f :
. . . . .    (tg:Id
. . . . .     ×State(ds)→ma-valtype(da; k)→
. . . . .     (da(rcv(l; tg))?Void List)) List
. . . . . 7. source(l) = i
. . . . . 8. w : World
. . . . . 9. p : FairFifo
. . . . . 10. FairFifo
. . . . . 11. ∀l@0:IdLnk, tg:Id.
. . . . .     (w.M(l@0,tg)) ⊆r if eqof(IdDeq)(source(l@0),i)→
. . . . .                         <ds,da,,,,<[<k,l>],λx.f>,,,•>
. . . . .                      else  fi.da(rcv(l@0; tg))
. . . . . 12. (eqof(IdDeq)(i,i)) ∼ true₂
. . . . . 13. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . . . 14. ∀e:E.
. . . . .     loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . . . 15. ∀e:E.
. . . . .     isrcv(e)
. . . . .     ⇒
```

```
. . . . .     lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . . . 16. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . . .     ((λl,tg. da(rcv(l; tg))?Top))
. . . . . 17. ∀t:ℕ.
. . . . .     ¬isnull(a(i;t))
. . . . .     ⇒
. . . . .     (islocal(kind(a(i;t)))
. . . . .      ⇒
. . . . .      deq-member(IdDeq;act(kind(a(i;t)));1of())
. . . . .      ⇒
. . . . .      2of()(act(kind(a(i;t))),λx.s(i;t).x,val(a(i;t))))
. . . . .     & (∀x:Id.
. . . . .        deq-member(product-deq(Knd;Id;KindDeq;IdDeq);<...
. . . . .                                                   ,x>;1of(
. . . . .                                                       ))
. . . . .        ⇒
. . . . .        s(i;t+1).x
. . . . .        =
. . . . .        2of()(<kind(a(i;t)),x>,λx.s(i;t).x,val(a(i;t))))
. . . . .     & (∀l@0:IdLnk.
. . . . .        (eqof(product-deq(Knd;IdLnk;KindDeq;IdLnkDeq))
. . . . .         (<k,l>
. . . . .         ,<kind(a(i;t)),l@0>) ∨₂ false₂)
. . . . .        ⇒
. . . . .        withlnk(l@0;m(i;t))
. . . . .        =
. . . . .        if source(l@0) = i→
. . . . .           concat(map(λtgf.map(λx.<1of(tgf)
. . . . .                                  ,x>;2of(tgf)
. . . . .                                      ((λx.s(i;t).x)
. . . . .                                      ,val(a(i;t))));f))
. . . . .        else nil fi)
. . . . .     & (∀x:Id.
. . . . .        ¬(deq-member(IdDeq;x;1of())
. . . . .          ⇒
. . . . .          deq-member(KindDeq;kind(a(i;t));2of()(x)))
. . . . .        ⇒
. . . . .        s(i;t).x = s(i;t+1).x)
. . . . .     & (∀l@0:IdLnk, tg:Id.
. . . . .        ¬(deq-member(product-deq(IdLnk;Id;...;IdDeq);...;1of(
. . . . .                                                       ))
. . . . .          ⇒
. . . . .          deq-member(KindDeq;kind(a(i;t));2of()(<l@0,tg>)))
. . . . .        ⇒
. . . . .        w-tagged(tg; onlnk(l@0;m(i;t))) = nil)
. . . . . 18. ∀a:Action(i).
. . . . .     ¬isnull(a) ⇒ (valtype(i;a) ⊆r da(kind(a))?Top)
. . . . . 19. ∀x:Id.
. . . . .     vartype(i;x) ⊆r if deq-member(IdDeq;x;1of(ds))→
. . . . .                          2of(ds)(x)
. . . . .                       else Top fi
. . . . . 20. IdDeq ∈ EqDecider(Id)
. . . . . 21. (λz.(z initially i)) ∈ State(ds)
. . . . . 22. ∀x:Id. vartype(i;x) ⊆r ds(x)?Top
. . . . . 23. ∀e:E.
```

```
. . . . .     loc(e) = i ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . . . 24. ∀e:E.
. . . . .     isrcv(e)
. . . . .     ⇒
. . . . .     lnk(e) = l ⇒ (valtype(e) ⊆r ma-valtype(da; kind(e)))
. . . . . 25. {m:Msg| source(mlnk(m)) = i } ⊆r Msg
. . . . .     ((λl,tg. da(rcv(l; tg))?Top))
. . . . . ⊢  E ∈ Type  by Auto
END
```