```
ma-send_wf  THM

⊢  ∀M:MsgA, k:Knd, l:IdLnk, s:M.state, v:M.V(k), i:Id,
|   ms:(tg:Id×if source(l) = i→ M.da(rcv(l; tg)) else Top fi) List.
|  M.send(k;l;s;v;ms;i) ∈ Prop
| by Auto THEN Unfold ‘ma-send‘ 0 THEN Unfold_MsgA -7
|    THEN
|    Unfold ‘fpf-val‘ 0
|    THEN
|    Analyze
|
|\
| .....subterm .....
| 1. ds : x:Id fp-> Type
| 2. da : a:Knd fp-> Type
| 3. x:Id fp-> ds(x)?Void
| 4. a:Id fp-> State(ds)→da(locl(a))?Top→Prop
| 5. kx:Knd×Id fp-> State(ds)→da(1of(kx))?Top→ds(2of(kx))?Void
| 6. send :
|    kl:Knd×IdLnk fp-> (tg:Id
|                       ×State(ds)→da(1of(kl))?Top→
|                       (da(rcv(2of(kl); tg))?Void List)) List
| 7. x:Id fp-> Knd List
| 8. ltg:IdLnk×Id fp-> Knd List
| 9. Top
| 10. k : Knd
| 11. l : IdLnk
| 12. State(ds)
| 13. ma-valtype(da; k)
| 14. i : Id
| 15. (tg:Id×if source(l) = i→ da(rcv(l; tg))?Top else Top fi) List
| ⊢  <k,l> ∈ dom(send) ∈ Prop
|   by DVar ‘send’ THEN Unfold ‘fpf-dom‘ 0 THEN Reduce 0
|
.\
. .....subterm .....
. 1. ds : x:Id fp-> Type
. 2. da : a:Knd fp-> Type
. 3. x:Id fp-> ds(x)?Void
. 4. a:Id fp-> State(ds)→da(locl(a))?Top→Prop
. 5. kx:Knd×Id fp-> State(ds)→da(1of(kx))?Top→ds(2of(kx))?Void
. 6. send :
. |  kl:Knd×IdLnk fp-> (tg:Id
. |                       ×State(ds)→da(1of(kl))?Top→
. |                       (da(rcv(2of(kl); tg))?Void List)) List
. 7. x:Id fp-> Knd List
. 8. ltg:IdLnk×Id fp-> Knd List
. 9. Top
. 10. k : Knd
. 11. l : IdLnk
. 12. s : State(ds)
. 13. v : ma-valtype(da; k)
. 14. i : Id
. 15. ms :
. |   (tg:Id×if source(l) = i→ da(rcv(l; tg))?Top else Top fi) List
```

```
. 16. <k,l> ∈ dom(send)
. ⊢  (ms
. |   =
. |   if source(l) = i→
. |      concat(map(λtgf.map(λx.<1of(tgf),x>;2of(tgf)
. |                                          (s
. |                                          ,v));send(<k,l>)))
. |   else nil fi)
. |  ∈ Prop  by Analyze THEN Try (Complete Auto)
. |
. .....subterm .....
. 1. ds : x:Id fp-> Type
. 2. da : a:Knd fp-> Type
. 3. x:Id fp-> ds(x)?Void
. 4. a:Id fp-> State(ds)→da(locl(a))?Top→Prop
. 5. kx:Knd×Id fp-> State(ds)→da(1of(kx))?Top→ds(2of(kx))?Void
. 6. send :
. |  kl:Knd×IdLnk fp-> (tg:Id
. |                     ×State(ds)→da(1of(kl))?Top→
. |                     (da(rcv(2of(kl); tg))?Void List)) List
. 7. x:Id fp-> Knd List
. 8. ltg:IdLnk×Id fp-> Knd List
. 9. Top
. 10. k : Knd
. 11. l : IdLnk
. 12. s : State(ds)
. 13. v : ma-valtype(da; k)
. 14. i : Id
. 15. (tg:Id×if source(l) = i→ da(rcv(l; tg))?Top else Top fi) List
. 16. <k,l> ∈ dom(send)
. ⊢  if source(l) = i→
. |     concat(map(λtgf.map(λx.<1of(tgf),x>;2of(tgf)
. |                                       (s
. |                                       ,v));send(<k,l>)))
. |  else nil fi
. |  ∈ (tg:Id×if source(l) = i→ da(rcv(l; tg))?Top else Top fi) List
. | by SplitOnConclITE
. |
. |\
. | .....truecase .....
. | 1. ds : x:Id fp-> Type
. | 2. da : a:Knd fp-> Type
. | 3. x:Id fp-> ds(x)?Void
. | 4. a:Id fp-> State(ds)→da(locl(a))?Top→Prop
. | 5. kx:Knd×Id fp-> State(ds)→da(1of(kx))?Top→ds(2of(kx))?Void
. | 6. send :
. | |  kl:Knd×IdLnk fp-> (tg:Id
. | |                     ×State(ds)→da(1of(kl))?Top→
. | |                     (da(rcv(2of(kl); tg))?Void List)) List
. | 7. x:Id fp-> Knd List
. | 8. ltg:IdLnk×Id fp-> Knd List
. | 9. Top
. | 10. k : Knd
. | 11. l : IdLnk
. | 12. s : State(ds)
```

```
. | 13. v : ma-valtype(da; k)
. | 14. i : Id
. | 15. (tg:Id×if source(l) = i→ da(rcv(l; tg))?Top else Top fi) List
. | 16. <k,l> ∈ dom(send)
. | 17. source(l) = i
. | ⊢  concat(map(λtgf.map(λx.<1of(tgf),x>;2of(tgf)
. | |                                    (s
. | |                                    ,v));send(<k,l>)))
. | |  ∈ (tg:Id×da(rcv(l; tg))?Top) List
. | | by Auto THEN All Reduce THEN DoSubsume
. | |
. | .....subtype_rel .....
. | 1. ds : x:Id fp-> Type
. | 2. da : a:Knd fp-> Type
. | 3. x:Id fp-> ds(x)?Void
. | 4. a:Id fp-> State(ds)→da(locl(a))?Top→Prop
. | 5. kx:Knd×Id fp-> State(ds)→da(1of(kx))?Top→ds(2of(kx))?Void
. | 6. send :
. |    kl:Knd×IdLnk fp-> (tg:Id
. |                       ×State(ds)→da(1of(kl))?Top→
. |                       (da(rcv(2of(kl); tg))?Void List)) List
. | 7. x:Id fp-> Knd List
. | 8. ltg:IdLnk×Id fp-> Knd List
. | 9. Top
. | 10. k : Knd
. | 11. l : IdLnk
. | 12. State(ds)
. | 13. ma-valtype(da; k)
. | 14. i : Id
. | 15. (tg:Id×if source(l) = i→ da(rcv(l; tg))?Top else Top fi) List
. | 16. <k,l> ∈ dom(send)
. | 17. source(l) = i
. | 18. tg:Id×State(ds)→da(k)?Top→(da(rcv(l; tg))?Void List)
. | ⊢  ma-valtype(da; k) ⊆r da(k)?Top  by Unfold ‘ma-valtype‘ 0
. |
. .\
. . .....falsecase .....
. . 1. ds : x:Id fp-> Type
. . 2. da : a:Knd fp-> Type
. . 3. x:Id fp-> ds(x)?Void
. . 4. a:Id fp-> State(ds)→da(locl(a))?Top→Prop
. . 5. kx:Knd×Id fp-> State(ds)→da(1of(kx))?Top→ds(2of(kx))?Void
. . 6. send :
. .    kl:Knd×IdLnk fp-> (tg:Id
. .                       ×State(ds)→da(1of(kl))?Top→
. .                       (da(rcv(2of(kl); tg))?Void List)) List
. . 7. x:Id fp-> Knd List
. . 8. ltg:IdLnk×Id fp-> Knd List
. . 9. Top
. . 10. k : Knd
. . 11. l : IdLnk
. . 12. State(ds)
. . 13. ma-valtype(da; k)
. . 14. i : Id
. . 15. (tg:Id×if source(l) = i→ da(rcv(l; tg))?Top else Top fi) List
```

```
. . 16. <k,l> ∈ dom(send)
. . 17. ¬source(l) = i
. . ⊢  nil ∈ (Id×Top) List  by Auto
END
```