```
possible-world_wf  THM

⊢  ∀D:Dsys, w:World. PossibleWorld(D;w) ∈ Prop
| by let T0 ≡ Complete (Auto THEN DoSubsume THEN BackThruSomeHyp) ,
|        T1
|        ≡ DoSubsume
|          THEN
|          Subst (locl(act(kind(a(i;t)))) ~ kind(a(i;t))) 0
|          THENL
|          [MoveToConcl -1 THEN GenConclAtAddr [1;1;1]
|           THEN
|           KindCase -2
|          ;BackThruSomeHyp] ,
|        T2
|        ≡ Try
|          (Unfold `ma-v` 0 THEN Unfold `ma-valtype` 0
|           THEN
|           Fold `ma-da` 0)
|          THEN
|          DoSubsume
|          THEN
|          BackThruSomeHyp ,
|        T3
|        ≡ Unfold `ma-st` 0 THEN Unfold `ma-state` 0
|          THEN
|          Fold `ma-ds` 0
|          THEN
|          T0 ,
|        T4 ≡ Unfold `w-Msg` 0 ,
|        T
|        ≡ Complete
|          (Auto THEN (T0 ORELSE T1 ORELSE T2 ORELSE T3 ORELSE T4)) in
|    UnivCD THEN Unfold `possible-world` 0
|    THEN
|    Repeat
|    (Repeat (Analyze THENL [T;Id])
|     THEN
|     Repeat (Analyze THENL [Id;T]))
|    THEN
|    Analyze
|    THEN
|    Try T
|
.....subterm .....
1. D : Dsys
2. w : World
3. ∀i,x:Id. vartype(i;x) ⊆r M(i).ds(x)
4. ∀i:Id, a:Action(i).
|  ¬isnull(a) ⇒ (valtype(i;a) ⊆r M(i).da(kind(a)))
5. ∀l:IdLnk, tg:Id. (w.M(l,tg)) ⊆r M(source(l)).da(rcv(l; tg))
6. ∀i,x:Id. M(i).init(x,s(i;0).x)
7. i : Id
8. t : ℕ
9. ¬isnull(a(i;t))
10. l : IdLnk
```

```
⊢  withlnk(l;m(i;t))
|  ∈ (tg:Id×if source(l) = i→ M(i).da(rcv(l; tg)) else Top fi) List
| by Using [‘the_w’,w] Auto
|
|\
| .....unproved Inclusion subgoal .....
| 1. D : Dsys
| 2. w : World
| 3. ∀i,x:Id. vartype(i;x) ⊆r M(i).ds(x)
| 4. ∀i:Id, a:Action(i).
|    ¬isnull(a) ⇒ (valtype(i;a) ⊆r M(i).da(kind(a)))
| 5. ∀l:IdLnk, tg:Id. (w.M(l,tg)) ⊆r M(source(l)).da(rcv(l; tg))
| 6. ∀i,x:Id. M(i).init(x,s(i;0).x)
| 7. i : Id
| 8. t : ℕ
| 9. ¬isnull(a(i;t))
| 10. IdLnk
| 11. m(i;t) = m(i;t)
| 12. {m:Msg(w.M)| source(mlnk(m)) = i } List
| 13. {m:Msg(w.M)| source(mlnk(m)) = i } List
| 14. u : {m:Msg(w.M)| source(mlnk(m)) = i }
| ⊢  u ∈ Msg  by Analyze -1 THEN Unfold ‘w-Msg‘ 0
|
.\
. .....unproved Inclusion subgoal .....
. 1. D : Dsys
. 2. w : World
. 3. ∀i,x:Id. vartype(i;x) ⊆r M(i).ds(x)
. 4. ∀i:Id, a:Action(i).
. |  ¬isnull(a) ⇒ (valtype(i;a) ⊆r M(i).da(kind(a)))
. 5. ∀l:IdLnk, tg:Id. (w.M(l,tg)) ⊆r M(source(l)).da(rcv(l; tg))
. 6. ∀i,x:Id. M(i).init(x,s(i;0).x)
. 7. i : Id
. 8. t : ℕ
. 9. ¬isnull(a(i;t))
. 10. l : IdLnk
. 11. withlnk(l;m(i;t)) = withlnk(l;m(i;t))
. 12. (t:Id×w.M(l,t)) List
. 13. (t:Id×w.M(l,t)) List
. 14. t:Id×w.M(l,t)
. 15. t1 : Id
. ⊢  (w.M(l,t1)) ⊆ if source(l) = i→ M(i).da(rcv(l; t1)) else Top fi
. | by Analyze 0 THEN DoSubsume THEN SplitOnConclITE
. |
. |\
. | .....truecase .....
. | 1. D : Dsys
. | 2. w : World
. | 3. ∀i,x:Id. vartype(i;x) ⊆r M(i).ds(x)
. | 4. ∀i:Id, a:Action(i).
. |    ¬isnull(a) ⇒ (valtype(i;a) ⊆r M(i).da(kind(a)))
. | 5. ∀l:IdLnk, tg:Id. (w.M(l,tg)) ⊆r M(source(l)).da(rcv(l; tg))
. | 6. ∀i,x:Id. M(i).init(x,s(i;0).x)
. | 7. i : Id
. | 8. t : ℕ
```

```
. | 9. ¬isnull(a(i;t))
. | 10. l : IdLnk
. | 11. withlnk(l;m(i;t)) = withlnk(l;m(i;t))
. | 12. (t:Id×w.M(l,t)) List
. | 13. (t:Id×w.M(l,t)) List
. | 14. t:Id×w.M(l,t)
. | 15. t1 : Id
. | 16. w.M(l,t1)
. | 17. source(l) = i
. | ⊢  (w.M(l,t1)) ⊆r M(i).da(rcv(l; t1))
. |   by RevHypSubst -1 0 THEN BackThruSomeHyp
. |
. .\
. . .....falsecase .....
. . 1. D : Dsys
. . 2. w : World
. . 3. ∀i,x:Id. vartype(i;x) ⊆r M(i).ds(x)
. . 4. ∀i:Id, a:Action(i).
. .     ¬isnull(a) ⇒ (valtype(i;a) ⊆r M(i).da(kind(a)))
. . 5. ∀l:IdLnk, tg:Id. (w.M(l,tg)) ⊆r M(source(l)).da(rcv(l; tg))
. . 6. ∀i,x:Id. M(i).init(x,s(i;0).x)
. . 7. i : Id
. . 8. t : ℕ
. . 9. ¬isnull(a(i;t))
. . 10. l : IdLnk
. . 11. withlnk(l;m(i;t)) = withlnk(l;m(i;t))
. . 12. (t:Id×w.M(l,t)) List
. . 13. (t:Id×w.M(l,t)) List
. . 14. t:Id×w.M(l,t)
. . 15. t1 : Id
. . 16. w.M(l,t1)
. . 17. ¬2source(l) = i
. . ⊢  (w.M(l,t1)) ⊆r Top  by Analyze 0
END
```