```
ma-init_wf  THM

⊢  ∀M:MsgA, x:Id, v:M.ds(x). M.init(x,v) ∈ Prop
| by All_MsgA THEN Unfold ‘fpf-val‘ 0
|
.....subterm .....
1. ds : x:Id fp-> Type
2. da : a:Knd fp-> Type
3. init : x:Id fp-> ds(x)?Void
4. a:Id fp-> State(ds)→ma-valtype(da; locl(a))→Prop
5. kx:Knd×Id fp-> State(ds)→ma-valtype(da; 1of(kx))→ds(2of(kx))?Void
6. kl:Knd×IdLnk fp-> (tg:Id
|                     ×State(ds)→ma-valtype(da; 1of(kl))→
|                     (da(rcv(2of(kl); tg))?Void List)) List
7. x:Id fp-> Knd List
8. ltg:IdLnk×Id fp-> Knd List
9. Top
10. x : Id
11. v : ds(x)?Top
12. x ∈ dom(init)
⊢  v ∈ ds(x)?Void
| by DoSubsume THEN Unfold ‘fpf-cap‘ 0 THEN SplitOnConclITE
|    THEN
|    Analyze 0
|
1. ds : x:Id fp-> Type
2. da : a:Knd fp-> Type
3. init : x:Id fp-> ds(x)?Void
4. a:Id fp-> State(ds)→ma-valtype(da; locl(a))→Prop
5. kx:Knd×Id fp-> State(ds)→ma-valtype(da; 1of(kx))→ds(2of(kx))?Void
6. kl:Knd×IdLnk fp-> (tg:Id
|                     ×State(ds)→ma-valtype(da; 1of(kl))→
|                     (da(rcv(2of(kl); tg))?Void List)) List
7. x:Id fp-> Knd List
8. ltg:IdLnk×Id fp-> Knd List
9. Top
10. x : Id
11. ds(x)?Top
12. x ∈ dom(init)
13. ¬x ∈ dom(ds)
14. x1 : Top
⊢  x1 ∈ Void  by Assert Void THEN UseWitness init(x) THEN DoSubsume
|
.....subtype_rel .....
1. ds : x:Id fp-> Type
2. da : a:Knd fp-> Type
3. init : x:Id fp-> ds(x)?Void
4. a:Id fp-> State(ds)→ma-valtype(da; locl(a))→Prop
5. kx:Knd×Id fp-> State(ds)→ma-valtype(da; 1of(kx))→ds(2of(kx))?Void
6. kl:Knd×IdLnk fp-> (tg:Id
                     ×State(ds)→ma-valtype(da; 1of(kl))→
                     (da(rcv(2of(kl); tg))?Void List)) List
7. x:Id fp-> Knd List
8. ltg:IdLnk×Id fp-> Knd List
9. Top
```

```
10. x : Id
11. ds(x)?Top
12. x ∈ dom(init)
13. ¬x ∈ dom(ds)
14. Top
⊢  ds(x)?Void ⊆r Void  by Unfold ‘fpf-cap‘ 0 THEN SplitOnConclITE
END
```