```
decls_mng_singleton  THM

⊢  ∀d:dec(), rho:Decl, s:{[[d]] rho}. s ∈ {[[<d>]] rho}
| by D 0 THENA Auto THEN
Unfolds ``record decls_mng`` 0 THEN Reduce 0
|
THEN Auto
|
.....unproved Inclusion subgoal .....
1. d : dec()
2. rho : Decl
3. l:Label→decl_type([[d]] rho;l)
4. x1 : Label
⊢  decl_type([[d]] rho;x1)
|  ⊆ (∩d:{d@0:dec()| d@0 ∈ <d> }. decl_type([[d]] rho;x1))
| by Unfold `subtype` 0 THEN Auto
|
1. d : dec()
2. rho : Decl
3. l:Label→decl_type([[d]] rho;l)
4. x1 : Label
5. x : decl_type([[d]] rho;x1)
6. d1 : {d@0:dec()| d@0 ∈ <d> }
⊢  x ∈ decl_type([[d1]] rho;x1)  by D (-1) THEN RW ColMemberC (-1)
|
1. d : dec()
2. rho : Decl
3. l:Label→decl_type([[d]] rho;l)
4. x1 : Label
5. x : decl_type([[d]] rho;x1)
6. d1 : dec()
7. d1 = d
⊢  x ∈ decl_type([[d1]] rho;x1)
| by All (Unfolds ``decl_type dec_mng``)  THEN All (Unfold `dbase`)

|    THEN D (-2) THEN D 1 THEN All Reduce
|
1. l1 : Label
2. d3 : SimpleType
3. rho : Decl
4. l:Label→if l =₁ l1→ [[d3]] rho else Top fi
5. x1 : Label
6. x : if x1 =₁ l1→ [[d3]] rho else Top fi
7. lbl : Label
8. d2 : SimpleType
9. <lbl,d2> = <l1,d3>
⊢  x ∈ if x1 =₁ lbl→ [[d2]] rho else Top fi
| by Assert lbl = l1 & d2 = d3
|
|\
| .....assertion .....
| 1. l1 : Label
| 2. d3 : SimpleType
| 3. rho : Decl
| 4. l:Label→if l =₁ l1→ [[d3]] rho else Top fi
```

```
| 5. x1 : Label
| 6. if x1 =_1 l1→ [[d3]] rho else Top fi
| 7. lbl : Label
| 8. d2 : SimpleType
| 9. <lbl,d2> = <l1,d3>
| ⊢  lbl = l1 & d2 = d3  by Fold ‘mk_dec‘ (-1) THEN Auto
| |
| |\
| | 1. l1 : Label
| | 2. d3 : SimpleType
| | 3. rho : Decl
| | 4. l:Label→if l =_1 l1→ [[d3]] rho else Top fi
| | 5. x1 : Label
| | 6. if x1 =_1 l1→ [[d3]] rho else Top fi
| | 7. lbl : Label
| | 8. d2 : SimpleType
| | 9. mk_dec(lbl, d2) = mk_dec(l1, d3)
| | ⊢  lbl = l1
| |   by ApFunToHypEquands ‘z’ z.lbl Label
| |       (-1) THEN All Reduce THEN Auto
| |
| .\
| . 1. l1 : Label
| . 2. d3 : SimpleType
| . 3. rho : Decl
| . 4. l:Label→if l =_1 l1→ [[d3]] rho else Top fi
| . 5. x1 : Label
| . 6. if x1 =_1 l1→ [[d3]] rho else Top fi
| . 7. lbl : Label
| . 8. d2 : SimpleType
| . 9. mk_dec(lbl, d2) = mk_dec(l1, d3)
| . ⊢  d2 = d3
| .   by ApFunToHypEquands ‘z’ z.typ SimpleType
| .       (-1) THEN All Reduce THEN Auto
|
.\
. 1. l1 : Label
. 2. d3 : SimpleType
. 3. rho : Decl
. 4. l:Label→if l =_1 l1→ [[d3]] rho else Top fi
. 5. x1 : Label
. 6. x : if x1 =_1 l1→ [[d3]] rho else Top fi
. 7. lbl : Label
. 8. d2 : SimpleType
. 9. <lbl,d2> = <l1,d3>
. 10. lbl = l1 & d2 = d3
. ⊢  x ∈ if x1 =_1 lbl→ [[d2]] rho else Top fi
. | by RepD THEN SplitOnConclITE THENA Auto
. |
. |\
. | .....truecase .....
. | 1. l1 : Label
. | 2. d3 : SimpleType
. | 3. rho : Decl
. | 4. l:Label→if l =_1 l1→ [[d3]] rho else Top fi
```

```
. | 5. x1 : Label
. | 6. x : if x1 =_1 l1→ [[d3]] rho else Top fi
. | 7. lbl : Label
. | 8. d2 : SimpleType
. | 9. <lbl,d2> = <l1,d3>
. | 10. lbl = l1
. | 11. d2 = d3
. | 12. x1 =_1 lbl
. | ⊢  x ∈ [[d2]] rho  by Subst x1 ∼ l1 6 THENA Auto
. | |
. | |\
. | | 1. l1 : Label
. | | 2. d3 : SimpleType
. | | 3. rho : Decl
. | | 4. l:Label→if l =_1 l1→ [[d3]] rho else Top fi
. | | 5. x1 : Label
. | | 6. if x1 =_1 l1→ [[d3]] rho else Top fi
. | | 7. lbl : Label
. | | 8. d2 : SimpleType
. | | 9. <lbl,d2> = <l1,d3>
. | | 10. lbl = l1
. | | 11. d2 = d3
. | | 12. x1 =_1 lbl
. | | ⊢  x1 = l1  by EqLblFwd THEN RelRST
. | |
. | .\
. | . 1. l1 : Label
. | . 2. d3 : SimpleType
. | . 3. rho : Decl
. | . 4. l:Label→if l =_1 l1→ [[d3]] rho else Top fi
. | . 5. x1 : Label
. | . 6. x : if l1 =_1 l1→ [[d3]] rho else Top fi
. | . 7. lbl : Label
. | . 8. d2 : SimpleType
. | . 9. <lbl,d2> = <l1,d3>
. | . 10. lbl = l1
. | . 11. d2 = d3
. | . 12. x1 =_1 lbl
. | . ⊢  x ∈ [[d2]] rho
. | .   by EqLblReflexive 6
. | . THEN Subst d2 ∼ d3 0
. | . THEN Auto
. |
. .\
. . .....falsecase .....
. . 1. l1 : Label
. . 2. d3 : SimpleType
. . 3. rho : Decl
. . 4. l:Label→if l =_1 l1→ [[d3]] rho else Top fi
. . 5. x1 : Label
. . 6. x : if x1 =_1 l1→ [[d3]] rho else Top fi
. . 7. lbl : Label
. . 8. d2 : SimpleType
. . 9. <lbl,d2> = <l1,d3>
. . 10. lbl = l1
```

```
. . 11. d2 = d3
. . 12. ¬x1 =₁ lbl
. . ⊢  x ∈ Top  by Unfold ‘top‘ 0 THEN Auto
END
```