```
list_all_is_member_lemma  THM

⊢  ∀eq:{T=₂}, P:(T→Prop), L:T List.
|  ∀x∈L.P(x) ⟺ (∀z:T. z(∈eq) L ⇒ P(z))  by UnivCD
|
1. T : Type
2. eq : {T=₂}
3. P : T→Prop
4. L : T List
⊢  ∀x∈L.P(x) ⟺ (∀z:T. z(∈eq) L ⇒ P(z))
| by DiscreteEq 2 THEN GenUnivCD
|
|\
| 3. eq ∈ T→T→𝔹
| 4. ∀x,y:T. eq(x,y) ⟺ x = y
| 5. P : T→Prop
| 6. L : T List
| 7. ∀x∈L.P(x)
| 8. z : T
| 9. z(∈eq) L
| ⊢  P(z)  by ListInd 6
| |
| |\
| | 7. z : T
| | ⊢  ∀x∈nil.P(x) ⇒ z(∈eq) nil ⇒ P(z)
| |   by Rewrite
| |      (HigherC list_all_unrollC THENC HigherC is_member_unrollC
| |       THENC
| |       HigherC assert_evalC)
| |      0
| |
| .\
| . 7. z : T
| . 8. u : T
| . 9. v : T List
| . 10. ∀x∈v.P(x) ⇒ z(∈eq) v ⇒ P(z)
| . ⊢  ∀x∈(u.v).P(x) ⇒ z(∈eq) (u.v) ⇒ P(z)
| . | by UnivCD THEN Decide (eq(z,u))
| . |
| . |\
| . | 11. ∀x∈(u.v).P(x)
| . | 12. z(∈eq) (u.v)
| . | 13. eq(z,u)
| . | ⊢  P(z)  by FwdThru 4 [13]
| . | |
| . | 14. z = u
| . | ⊢  P(z)
| . | | by Rewrite (HigherC list_all_unrollC) 11 THEN Analyze 11
| . | |
| . | 11. P(u)
| . | 12. ∀x∈v.P(x)
| . | 13. z(∈eq) (u.v)
| . | 14. eq(z,u)
| . | 15. z = u
| . | ⊢  P(z)  by HypSubst -1 0 THEN Trivial
```

```
| . |
| . .\
| . . 11. ∀x∈(u.v).P(x)
| . . 12. z(∈eq) (u.v)
| . . 13. ¬eq(z,u)
| . . ⊢  P(z)
| . . | by Rewrite (HigherC list_all_unrollC) 11 THEN Analyze 11
| . . |
| . . 11. P(u)
| . . 12. ∀x∈v.P(x)
| . . 13. z(∈eq) (u.v)
| . . 14. ¬eq(z,u)
| . . ⊢  P(z)  by BackThru 10
| . . |
| . . |\
| . . | ⊢  ∀x∈v.P(x)  by Trivial
| . . |
| . . .\
| . . . ⊢  z(∈eq) v
| . . . | by Rewrite (HigherC is_member_unrollC) 13
| . . . |    THEN
| . . . |    FwdThru
| . . . |    Thm* ∀P:𝔹. ¬P ⟺ P = false₂    {not_assert_iff_bfalse}
| . . . |    [14]
| . . . |
| . . . 13. if eq(z,u)→ true₂ else z(∈eq) v fi
| . . . 14. ¬eq(z,u)
| . . . 15. eq(z,u) = false₂
| . . . ⊢  z(∈eq) v
| . . . | by HypSubst 15 13
| . . . |    THEN
| . . . |    Rewrite (HigherC ifthenelse_evalC) 13
| . . . |
| . . . 13. z(∈eq) v
| . . . 14. ¬eq(z,u)
| . . . 15. eq(z,u) = false₂
| . . . ⊢  z(∈eq) v  by Trivial
|
.\
. 3. eq ∈ T→T→𝔹
. 4. ∀x,y:T. eq(x,y) ⟺ x = y
. 5. P : T→Prop
. 6. L : T List
. 7. ∀z:T. z(∈eq) L ⇒ P(z)
. ⊢  ∀x∈L.P(x)  by ListInd 6
. |
. |\
. | 6. T List
. | ⊢  (∀z:T. z(∈eq) nil ⇒ P(z)) ⇒ ∀x∈nil.P(x)
. |   by Rewrite (HigherC list_all_unrollC) 0
. |
. .\
. . 7. u : T
. . 8. v : T List
. . 9. (∀z:T. z(∈eq) v ⇒ P(z)) ⇒ ∀x∈v.P(x)
```

```
. . ⊢  (∀z:T. z(∈eq) (u.v) ⇒ P(z)) ⇒ ∀x∈(u.v).P(x)  by UnivCD
. . |
. . 10. ∀z:T. z(∈eq) (u.v) ⇒ P(z)
. . ⊢  ∀x∈(u.v).P(x)
. . | by Rewrite (HigherC list_all_unrollC) 0 THEN Analyze 0
. . |
. . |\
. . | ⊢  P(u)  by BackThru -1
. . | |
. . | ⊢  u(∈eq) (u.v)  by Rewrite (HigherC is_member_unrollC) 0
. . | |
. . | ⊢  if eq(u,u)→ true₂ else u(∈eq) v fi
. . | | by With u (Analyze 4) THEN With u (Analyze -1) THEN Analyze -1
. . | |    THEN
. . | |    Thin -2
. . | |
. . | 4. P : T→Prop
. . | 5. T List
. . | 6. u : T
. . | 7. v : T List
. . | 8. (∀z:T. z(∈eq) v ⇒ P(z)) ⇒ ∀x∈v.P(x)
. . | 9. ∀z:T. z(∈eq) (u.v) ⇒ P(z)
. . | 10. (eq(u,u)) ⇐ (u = u)
. . | ⊢  if eq(u,u)→ true₂ else u(∈eq) v fi
. . | | by Assert (u = u) THENL [Auto;Id]
. . | |
. . | 11. u = u
. . | ⊢  if eq(u,u)→ true₂ else u(∈eq) v fi
. . | | by FwdThru 10 [11]
. . | |    THEN
. . | |    FwdThru Thm* ∀P:𝔹. P ⟺ P = true₂    {assert_iff_btrue}
. . | |    [-1]
. . | |
. . | 12. eq(u,u)
. . | 13. eq(u,u) = true₂
. . | ⊢  if eq(u,u)→ true₂ else u(∈eq) v fi
. . |   by HypSubst 13 0
. . |      THEN
. . |      Rewrite (HigherC ifthenelse_evalC THENC assert_evalC) 0
. . |      THEN
. . |      Trivial
. . |
. . .\
. . . ⊢  ∀x∈v.P(x)  by BackThru 9
. . . |
. . . ⊢  ∀z:T. z(∈eq) v ⇒ P(z)  by UnivCD
. . . |
. . . 11. z : T
. . . 12. z(∈eq) v
. . . ⊢  P(z)  by BackThru 10
. . . |
. . . ⊢  z(∈eq) (u.v)  by Rewrite (HigherC is_member_unrollC) 0
. . . |
. . . ⊢  if eq(z,u)→ true₂ else z(∈eq) v fi
. . .   by GenConcl (eq(z,u) = b) THEN BoolInd -2
```

```
. . .      THEN
. . .      Rewrite (HigherC ifthenelse_evalC THENC TryC assert_evalC)
. . .      0
. . .      THEN
. . .      Try Trivial
END
```