```
is_member_append_lemma  THM

⊢  ∀eq:{T=₂}, L:T List, x:T.
|  x(∈eq) L ⇒ (∃M,N:T List. L = (M @ (x.N)))
| by RepeatFor 2 (Analyze 0) THEN DiscreteEq -1 THEN UnivCD
|
1. T : Type
2. eq : {T=₂}
3. eq ∈ T→T→𝔹
4. ∀x,y:T. eq(x,y) ⟺ x = y
5. L : T List
6. x : T
7. x(∈eq) L
⊢  ∃M,N:T List. L = (M @ (x.N))
| by ListInd 5
|    THEN
|    Rewrite
|    (HigherC is_member_unrollC THENC TryC (HigherC assert_evalC))
|    0
|
|\
| 6. x : T
| ⊢  False ⇒ (∃M,N:T List. nil = (M @ (x.N)))  by Auto
|
.\
. 7. u : T
. 8. v : T List
. 9. x(∈eq) v ⇒ (∃M,N:T List. v = (M @ (x.N)))
. ⊢  if eq(x,u)→ true₂ else x(∈eq) v fi
. |  ⇒
. |  (∃M,N:T List. u.v = (M @ (x.N)))
. | by GenConcl (eq(x,u) = b) THEN BoolInd -2
. |    THEN
. |    Rewrite
. |    (HigherC ifthenelse_evalC THENC TryC (HigherC assert_evalC))
. |    0
. |
. |\
. | 10. eq(x,u) = true₂
. | ⊢  True ⇒ (∃M,N:T List. u.v = (M @ (x.N)))
. | | by Analyze 0 THEN Thin -1
. | |
. | ⊢  ∃M,N:T List. u.v = (M @ (x.N))  by Witness nil THEN Witness v
. | |
. | ⊢  u.v = (nil @ (x.v))  by Reduce 0
. | |
. | ⊢  u.v = x.v  by MemberEqCD
. | |
. | ⊢  u = x  by Assert (x = u)
. | |
. | .....assertion .....
. | ⊢  x = u  by HypBackchain
. | |
. | ⊢  eq(x,u)  by HypSubst -1 0 THEN Reduce 0
. |
```

```
. .\
. . 10. eq(x,u) = false₂
. . ⊢  x(∈eq) v ⇒ (∃M,N:T List. u.v = (M @ (x.N)))  by Analyze 0
. . |
. . 11. x(∈eq) v
. . ⊢  ∃M,N:T List. u.v = (M @ (x.N))
. . | by RenameVar ‘xx’ 11 THEN Witness9 xx THEN Analyze -1
. . |    THEN
. . |    Analyze -1
. . |
. . 9. eq(x,u) = false₂
. . 10. x(∈eq) v
. . 11. M : T List
. . 12. N : T List
. . 13. v = (M @ (x.N))
. . ⊢  ∃M,N:T List. u.v = (M @ (x.N))  by Witness u.M THEN Witness N
. . |
. . ⊢  u.v = ((u.M) @ (x.N))  by Reduce 0 THEN HypSubst 13 0
END
```