```
l_member_decomp  THM

⊢  ∀T:Type, l:T List, x:T.
|  (x ∈ l) ⟺ (∃l1,l2:T List. l = (l1 @ [x] @ l2))
| by ((InductionOnList
|      THEN
|      RWW
|      Thm* ∀l:T List, a,x:T. (x ∈ [a / l]) ⟺ x = a ∨ (x ∈ l)
|      {cons_member}
|      0)
|     THEN
|     (RWW Thm* ∀x:T. (x ∈ nil) ⟺ False    {nil_member} 0))
|    THEN
|    ExRepD
|
|\
| 1. T : Type
| 2. T List
| 3. x : T
| 4. l1 : T List
| 5. l2 : T List
| 6. nil = (l1 @ [x] @ l2)
| ⊢  False  by Obvious
|
|\
| 1. T : Type
| 2. T List
| 3. u : T
| 4. v : T List
| 5. ∀x:T. (x ∈ v) ⟺ (∃l1,l2:T List. v = (l1 @ [x] @ l2))
| 6. x : T
| 7. x = u ∨ (x ∈ v)
| ⊢  ∃l1,l2:T List. [u / v] = (l1 @ [x] @ l2)  by Analyze -1
| |
| |\
| | 1. T : Type
| | 2. T List
| | 3. u : T
| | 4. v : T List
| | 5. ∀x:T. (x ∈ v) ⟺ (∃l1,l2:T List. v = (l1 @ [x] @ l2))
| | 6. x : T
| | 7. x = u
| | ⊢  ∃l1,l2:T List. [u / v] = (l1 @ [x] @ l2)
| |   by InstConcl [nil;v] THEN Reduce 0 THEN Analyze
| |
| .\
| . 1. T : Type
| . 2. T List
| . 3. u : T
| . 4. v : T List
| . 5. ∀x:T. (x ∈ v) ⟺ (∃l1,l2:T List. v = (l1 @ [x] @ l2))
| . 6. x : T
| . 7. (x ∈ v)
| . ⊢  ∃l1,l2:T List. [u / v] = (l1 @ [x] @ l2)  by FwdThru -3 [-1]
| . |
```

```
| . 1. T : Type
| . 2. T List
| . 3. u : T
| . 4. v : T List
| . 5. ∀x:T. (x ∈ v) ⟺ (∃l1,l2:T List. v = (l1 @ [x] @ l2))
| . 6. x : T
| . 7. (x ∈ v)
| . 8. ∃l1,l2:T List. v = (l1 @ [x] @ l2)
| . ⊢  ∃l1,l2:T List. [u / v] = (l1 @ [x] @ l2)
| . | by ExRepD THEN InstConcl [[u / l1];l2]
| . |
| . 1. T : Type
| . 2. T List
| . 3. u : T
| . 4. v : T List
| . 5. ∀x:T. (x ∈ v) ⟺ (∃l1,l2:T List. v = (l1 @ [x] @ l2))
| . 6. x : T
| . 7. (x ∈ v)
| . 8. l1 : T List
| . 9. l2 : T List
| . 10. v = (l1 @ [x] @ l2)
| . ⊢  [u / v] = ([u / l1] @ [x] @ l2)  by All Reduce THEN Analyze
|
.\
. 1. T : Type
. 2. T List
. 3. u : T
. 4. v : T List
. 5. ∀x:T. (x ∈ v) ⟺ (∃l1,l2:T List. v = (l1 @ [x] @ l2))
. 6. x : T
. 7. l1 : T List
. 8. l2 : T List
. 9. [u / v] = (l1 @ [x] @ l2)
. ⊢  x = u ∨ (x ∈ v)  by Analyze -3 THEN All Reduce
. |
. |\
. | 1. T : Type
. | 2. T List
. | 3. u : T
. | 4. v : T List
. | 5. ∀x:T. (x ∈ v) ⟺ (∃l1,l2:T List. v = (l1 @ [x / l2]))
. | 6. x : T
. | 7. l2 : T List
. | 8. [u / v] = [x / l2]
. | ⊢  x = u ∨ (x ∈ v)  by Obvious
. |
. .\
. . 1. T : Type
. . 2. T List
. . 3. u : T
. . 4. v : T List
. . 5. ∀x:T. (x ∈ v) ⟺ (∃l1,l2:T List. v = (l1 @ [x / l2]))
. . 6. x : T
. . 7. u1 : T
. . 8. v1 : T List
```

```
. . 9. l2 : T List
. . 10. [u / v] = [u1 / (v1 @ [x / l2])]
. . ⊢  x = u ∨ (x ∈ v)  by Subst (v = (v1 @ [x / l2])) 0
. . |
. . |\
. . | .....equality .....
. . | 1. T : Type
. . | 2. T List
. . | 3. u : T
. . | 4. v : T List
. . | 5. ∀x:T. (x ∈ v) ⟺ (∃l1,l2:T List. v = (l1 @ [x / l2]))
. . | 6. x : T
. . | 7. u1 : T
. . | 8. v1 : T List
. . | 9. l2 : T List
. . | 10. [u / v] = [u1 / (v1 @ [x / l2])]
. . | ⊢  v = (v1 @ [x / l2])  by Obvious
. . |
. . .\
. . . 1. T : Type
. . . 2. T List
. . . 3. u : T
. . . 4. v : T List
. . . 5. ∀x:T. (x ∈ v) ⟺ (∃l1,l2:T List. v = (l1 @ [x / l2]))
. . . 6. x : T
. . . 7. u1 : T
. . . 8. v1 : T List
. . . 9. l2 : T List
. . . 10. [u / v] = [u1 / (v1 @ [x / l2])]
. . . ⊢  x = u ∨ (x ∈ v1 @ [x / l2])
. . . | by (RWW
. . . |     Thm* ∀x:T, l1,l2:T List.
. . . |          (x ∈ l1 @ l2) ⟺ (x ∈ l1) ∨ (x ∈ l2)
. . . |     {member_append}
. . . |     0)
. . . |    THEN
. . . |    (RWW
. . . |     Thm* ∀l:T List, a,x:T. (x ∈ [a / l]) ⟺ x = a ∨ (x ∈ l)
. . . |     {cons_member}
. . . |     0)
. . . |
. . . 1. T : Type
. . . 2. T List
. . . 3. u : T
. . . 4. v : T List
. . . 5. ∀x:T. (x ∈ v) ⟺ (∃l1,l2:T List. v = (l1 @ [x / l2]))
. . . 6. x : T
. . . 7. u1 : T
. . . 8. v1 : T List
. . . 9. l2 : T List
. . . 10. [u / v] = [u1 / (v1 @ [x / l2])]
. . . ⊢  x = u ∨ (x ∈ v1) ∨ x = x ∨ (x ∈ l2)  by Obvious
END
```