```
l_before_append_iff  THM

⊢  ∀T:Type, A,B:T List, x,y:T.
|  x before y ∈ A @ B
|  ⟺
|  x before y ∈ A ∨ x before y ∈ B ∨ (x ∈ A) & (y ∈ B)
| by (Unfold ‘l_before‘ 0)
|    THEN
|    (RWO
|     Thm* ∀C,A,B:T List.
|          C ⊆ A @ B
|          ⟺
|          (∃A’,B’:T List. C = (A’ @ B’) & A’ ⊆ A & B’ ⊆ B)
|     {sublist_append_iff}
|     0)
|
|\
| 1. T : Type
| 2. A : T List
| 3. B : T List
| 4. x : T
| 5. y : T
| 6. ∃A’,B’:T List. [x; y] = (A’ @ B’) & A’ ⊆ A & B’ ⊆ B
| ⊢  [x; y] ⊆ A ∨ [x; y] ⊆ B ∨ (x ∈ A) & (y ∈ B)
| | by ExRepD THEN (Analyze -5) THEN (All Reduce)
| |
| |\
| | 1. T : Type
| | 2. A : T List
| | 3. B : T List
| | 4. x : T
| | 5. y : T
| | 6. B’ : T List
| | 7. [x; y] = B’
| | 8. nil ⊆ A
| | 9. B’ ⊆ B
| | ⊢  [x; y] ⊆ A ∨ [x; y] ⊆ B ∨ (x ∈ A) & (y ∈ B)  by Obvious
| |
| .\
| . 1. T : Type
| . 2. A : T List
| . 3. B : T List
| . 4. x : T
| . 5. y : T
| . 6. u : T
| . 7. v : T List
| . 8. B’ : T List
| . 9. [x; y] = [u / (v @ B’)]
| . 10. [u / v] ⊆ A
| . 11. B’ ⊆ B
| . ⊢  [x; y] ⊆ A ∨ [x; y] ⊆ B ∨ (x ∈ A) & (y ∈ B)
| . | by (Analyze -5) THEN (All Reduce)
| . |
| . |\
| . | 1. T : Type
```

```
| . | 2. A : T List
| . | 3. B : T List
| . | 4. x : T
| . | 5. y : T
| . | 6. u : T
| . | 7. B' : T List
| . | 8. [x; y] = [u / B']
| . | 9. [u] ⊆ A
| . | 10. B' ⊆ B
| . | ⊢  [x; y] ⊆ A ∨ [x; y] ⊆ B ∨ (x ∈ A) & (y ∈ B)
| . |   by (OrRight THEN OrRight)
| . |      THEN
| . |      (RWO
| . |       Thm* ∀x:T, L:T List. (x ∈ L) ⟺ [x] ⊆ L
| . |       {member_iff_sublist}
| . |       0)
| . |      THEN
| . |      (Try Obvious)
| . |
| . .\
| . . 1. T : Type
| . . 2. A : T List
| . . 3. B : T List
| . . 4. x : T
| . . 5. y : T
| . . 6. u : T
| . . 7. u1 : T
| . . 8. v1 : T List
| . . 9. B' : T List
| . . 10. [x; y] = [u; u1 / (v1 @ B')]
| . . 11. [u; u1 / v1] ⊆ A
| . . 12. B' ⊆ B
| . . ⊢  [x; y] ⊆ A ∨ [x; y] ⊆ B ∨ (x ∈ A) & (y ∈ B)
| . . | by OrLeft
| . . |    THEN
| . . |    (Using [`L2',[u; u1 / v1]]
| . . |     (BackThru
| . . |      Thm* ∀L1,L2,L3:T List. L1 ⊆ L2 ⇒ L2 ⊆ L3 ⇒ L1 ⊆ L3
| . . |      {sublist_transitivity}))
| . . |
| . . 1. T : Type
| . . 2. A : T List
| . . 3. B : T List
| . . 4. x : T
| . . 5. y : T
| . . 6. u : T
| . . 7. u1 : T
| . . 8. v1 : T List
| . . 9. B' : T List
| . . 10. [x; y] = [u; u1 / (v1 @ B')]
| . . 11. [u; u1 / v1] ⊆ A
| . . 12. B' ⊆ B
| . . ⊢  [x; y] ⊆ [u; u1 / v1]
| . .   by (((RWO
| . .          Thm* ∀x1,x2:T, L1,L2:T List.
```

```
| . .                 [x1 / L1] ⊆ [x2 / L2]
| . .                 ⟺
| . .                 x1 = x2 & L1 ⊆ L2 ∨ [x1 / L1] ⊆ L2
| . .            {cons_sublist_cons}
| . .            0)
| . .           THEN
| . .           OrLeft)
| . .          THEN
| . .          (Try Obvious)
| . .          THEN
| . .          (RWO
| . .           Thm* ∀x1,x2:T, L1,L2:T List.
| . .                [x1 / L1] ⊆ [x2 / L2]
| . .                ⟺
| . .                x1 = x2 & L1 ⊆ L2 ∨ [x1 / L1] ⊆ L2
| . .           {cons_sublist_cons}
| . .           0))
| . .         THEN
| . .         OrLeft)
| . .        THEN
| . .        (Thin -1)
| . .        THEN
| . .        (Try Obvious)
|
.\
. 1. T : Type
. 2. A : T List
. 3. B : T List
. 4. x : T
. 5. y : T
. 6. [x; y] ⊆ A ∨ [x; y] ⊆ B ∨ (x ∈ A) & (y ∈ B)
. ⊢  ∃A',B':T List. [x; y] = (A' @ B') & A' ⊆ A & B' ⊆ B
. | by SplitOrHyps
. |
. |\
. | 1. T : Type
. | 2. A : T List
. | 3. B : T List
. | 4. x : T
. | 5. y : T
. | 6. [x; y] ⊆ A
. | ⊢  ∃A',B':T List. [x; y] = (A' @ B') & A' ⊆ A & B' ⊆ B
. |   by (InstConcl [[x; y];nil]) THEN (Reduce 0) THEN (Try Obvious)
. |
. |\
. | 1. T : Type
. | 2. A : T List
. | 3. B : T List
. | 4. x : T
. | 5. y : T
. | 6. [x; y] ⊆ B
. | ⊢  ∃A',B':T List. [x; y] = (A' @ B') & A' ⊆ A & B' ⊆ B
. |   by (InstConcl [nil;[x; y]]) THEN (Reduce 0) THEN (Try Obvious)
. |
. .\
```

```
. . 1. T : Type
. . 2. A : T List
. . 3. B : T List
. . 4. x : T
. . 5. y : T
. . 6. (x ∈ A) & (y ∈ B)
. . ⊢  ∃A',B':T List. [x; y] = (A' @ B') & A' ⊆ A & B' ⊆ B
. .   by (InstConcl [[x];[y]]) THEN (Reduce 0)
. .      THEN
. .      (RWO "member_iff_sublist<" 0)
END
```