```
l_before-iff  THM

⊢  ∀T:Type, L:T List, x,y:T.
|  x before y ∈ L
|  ⟺
|  (∃L1,L2,L3:T List. L = (L1 @ [x] @ L2 @ [y] @ L3))
| by InductionOnList
|
|\
| 1. T : Type
| 2. T List
| ⊢  ∀x,y:T.
|    x before y ∈ nil
|    ⟺
|    (∃L1,L2,L3:T List. nil = (L1 @ [x] @ L2 @ [y] @ L3))
|   by Auto THEN Obvious
|
.\
. 1. T : Type
. 2. T List
. 3. u : T
. 4. v : T List
. 5. ∀x,y:T.
. |  x before y ∈ v
. |  ⟺
. |  (∃L1,L2,L3:T List. v = (L1 @ [x] @ L2 @ [y] @ L3))
. ⊢  ∀x,y:T.
. |  x before y ∈ [u / v]
. |  ⟺
. |  (∃L1,L2,L3:T List. [u / v] = (L1 @ [x] @ L2 @ [y] @ L3))
. | by UnivCD
. |
. 1. T : Type
. 2. T List
. 3. u : T
. 4. v : T List
. 5. ∀x,y:T.
. |  x before y ∈ v
. |  ⟺
. |  (∃L1,L2,L3:T List. v = (L1 @ [x] @ L2 @ [y] @ L3))
. 6. x : T
. 7. y : T
. ⊢  x before y ∈ [u / v]
. |  ⟺
. |  (∃L1,L2,L3:T List. [u / v] = (L1 @ [x] @ L2 @ [y] @ L3))
. | by RWO
. |    Thm* ∀l:T List, a,x,y:T.
. |         x before y ∈ [a / l] ⟺ x = a & (y ∈ l) ∨ x before y ∈ l
. |    {cons_before}
. |    0
. |
. |\
. | 1. T : Type
. | 2. T List
. | 3. u : T
```

```
. | 4. v : T List
. | 5. ∀x,y:T.
. | |  x before y ∈ v
. | |  ⟺
. | |  (∃L1,L2,L3:T List. v = (L1 @ [x] @ L2 @ [y] @ L3))
. | 6. x : T
. | 7. y : T
. | 8. x = u & (y ∈ v) ∨ x before y ∈ v
. | ⊢  ∃L1,L2,L3:T List. [u / v] = (L1 @ [x] @ L2 @ [y] @ L3)
. | | by Analyze -1
. | |
. | |\
. | | 1. T : Type
. | | 2. T List
. | | 3. u : T
. | | 4. v : T List
. | | 5. ∀x,y:T.
. | | |  x before y ∈ v
. | | |  ⟺
. | | |  (∃L1,L2,L3:T List. v = (L1 @ [x] @ L2 @ [y] @ L3))
. | | 6. x : T
. | | 7. y : T
. | | 8. x = u
. | | 9. (y ∈ v)
. | | ⊢  ∃L1,L2,L3:T List. [u / v] = (L1 @ [x] @ L2 @ [y] @ L3)
. | | | by (Inst
. | | |     Thm* ∀s:T List, z:T.
. | | |          (z ∈ s) ⇒ (∃s1,s2:T List. s = (s1 @ [z / s2]))
. | | |     {l_member_decomp}
. | | |     [T;v;y])
. | | |    THEN
. | | |    ExRepD
. | | |    THEN
. | | |    (InstConcl [nil;s1;s2])
. | | |    THEN
. | | |    (Reduce 0)
. | | |
. | | 1. T : Type
. | | 2. T List
. | | 3. u : T
. | | 4. v : T List
. | | 5. ∀x,y:T.
. | |    x before y ∈ v
. | |    ⟺
. | |    (∃L1,L2,L3:T List. v = (L1 @ [x] @ L2 @ [y] @ L3))
. | | 6. x : T
. | | 7. y : T
. | | 8. x = u
. | | 9. (y ∈ v)
. | | 10. s1 : T List
. | | 11. s2 : T List
. | | 12. v = (s1 @ [y / s2])
. | | ⊢  [u / v] = [x / (s1 @ [y / s2])]  by Analyze
. | |
. | .\
```

```
. | . 1. T : Type
. | . 2. T List
. | . 3. u : T
. | . 4. v : T List
. | . 5. ∀x,y:T.
. | . |  x before y ∈ v
. | . |  ⟺
. | . |  (∃L1,L2,L3:T List. v = (L1 @ [x] @ L2 @ [y] @ L3))
. | . 6. x : T
. | . 7. y : T
. | . 8. x before y ∈ v
. | . ⊢  ∃L1,L2,L3:T List. [u / v] = (L1 @ [x] @ L2 @ [y] @ L3)
. | . | by (InstHyp [x;y] -4) THEN ThinTrivial THEN (Analyze -1)
. | . |    THEN
. | . |    (InstConcl [[u / L1]])
. | . |
. | . 1. T : Type
. | . 2. T List
. | . 3. u : T
. | . 4. v : T List
. | . 5. ∀x,y:T.
. | . |  x before y ∈ v
. | . |  ⟺
. | . |  (∃L1,L2,L3:T List. v = (L1 @ [x] @ L2 @ [y] @ L3))
. | . 6. x : T
. | . 7. y : T
. | . 8. x before y ∈ v
. | . 9. x before y ∈ v ⇐ (∃L1,L2,L3:T List.
. | . |  v = (L1 @ [x] @ L2 @ [y] @ L3))
. | . 10. L1 : T List
. | . 11. ∃L2,L3:T List. v = (L1 @ [x] @ L2 @ [y] @ L3)
. | . ⊢  ∃L2,L3:T List. [u / v] = ([u / L1] @ [x] @ L2 @ [y] @ L3)
. | . | by RepeatFor 2 (ParallelOp -1)
. | . |
. | . 1. T : Type
. | . 2. T List
. | . 3. u : T
. | . 4. v : T List
. | . 5. ∀x,y:T.
. | .    x before y ∈ v
. | .    ⟺
. | .    (∃L1,L2,L3:T List. v = (L1 @ [x] @ L2 @ [y] @ L3))
. | . 6. x : T
. | . 7. y : T
. | . 8. x before y ∈ v
. | . 9. x before y ∈ v ⇐ (∃L1,L2,L3:T List.
. | .    v = (L1 @ [x] @ L2 @ [y] @ L3))
. | . 10. L1 : T List
. | . 11. L2 : T List
. | . 12. L3 : T List
. | . 13. v = (L1 @ [x] @ L2 @ [y] @ L3)
. | . ⊢  [u / v] = ([u / L1] @ [x] @ L2 @ [y] @ L3)
. | .   by (All Reduce) THEN Analyze
. |
. .\
```

```
. . 1. T : Type
. . 2. T List
. . 3. u : T
. . 4. v : T List
. . 5. ∀x,y:T.
. . |  x before y ∈ v
. . |  ⟺
. . |  (∃L1,L2,L3:T List. v = (L1 @ [x] @ L2 @ [y] @ L3))
. . 6. x : T
. . 7. y : T
. . 8. ∃L1,L2,L3:T List. [u / v] = (L1 @ [x] @ L2 @ [y] @ L3)
. . ⊢  x = u & (y ∈ v) ∨ x before y ∈ v
. . | by ExRepD THEN (Analyze -4) THEN (All Reduce)
. . |    THEN
. . |    (SplitCons -1)
. . |
. . |\
. . | 1. T : Type
. . | 2. T List
. . | 3. u : T
. . | 4. v : T List
. . | 5. ∀x,y:T.
. . |    x before y ∈ v
. . |    ⟺
. . |    (∃L1,L2,L3:T List. v = (L1 @ [x / (L2 @ [y / L3])]))
. . | 6. x : T
. . | 7. y : T
. . | 8. L2 : T List
. . | 9. L3 : T List
. . | 10. u = x
. . | 11. v = (L2 @ [y / L3])
. . | ⊢  x = u & (y ∈ v) ∨ x before y ∈ v
. . |   by OrLeft THEN (HypSubst -1 0) THEN ObviousConcl
. . |
. . .\
. . . 1. T : Type
. . . 2. T List
. . . 3. u : T
. . . 4. v : T List
. . . 5. ∀x,y:T.
. . . |  x before y ∈ v
. . . |  ⟺
. . . |  (∃L1,L2,L3:T List. v = (L1 @ [x / (L2 @ [y / L3])]))
. . . 6. x : T
. . . 7. y : T
. . . 8. u1 : T
. . . 9. v1 : T List
. . . 10. L2 : T List
. . . 11. L3 : T List
. . . 12. u = u1
. . . 13. v = (v1 @ [x / (L2 @ [y / L3])])
. . . ⊢  x = u & (y ∈ v) ∨ x before y ∈ v
. . . | by OrRight THEN BackThruSomeHyp
. . . |
. . . 1. T : Type
```

```
. . . 2. T List
. . . 3. u : T
. . . 4. v : T List
. . . 5. ∀x,y:T.
. . .    x before y ∈ v
. . .    ⟺
. . .    (∃L1,L2,L3:T List. v = (L1 @ [x / (L2 @ [y / L3])]))
. . . 6. x : T
. . . 7. y : T
. . . 8. u1 : T
. . . 9. v1 : T List
. . . 10. L2 : T List
. . . 11. L3 : T List
. . . 12. u = u1
. . . 13. v = (v1 @ [x / (L2 @ [y / L3])])
. . . ⊢  ∃L1,L2,L3:T List. v = (L1 @ [x / (L2 @ [y / L3])])
. . .   by ObviousFrom [-1]
END
```