```
mapoutl_is_append  THM

⊢  ∀A,B:Type, L:(A+B) List, l1,l2:A List.
|  mapoutl(L) = (l1 @ l2)
|  ⟺
|  (∃L1,L2:(A+B) List.
|   L = (L1 @ L2) & mapoutl(L1) = l1 & mapoutl(L2) = l2)
| by Auto THEN ExRepD
|
|\
| 1. A : Type
| 2. B : Type
| 3. L : (A+B) List
| 4. l1 : A List
| 5. l2 : A List
| 6. mapoutl(L) = (l1 @ l2)
| ⊢  ∃L1,L2:(A+B) List.
| |  L = (L1 @ L2) & mapoutl(L1) = l1 & mapoutl(L2) = l2
| | by (MoveToConcl 3) THEN (ListInd -2) THEN (Reduce 0)
| |
| |\
| | 1. A : Type
| | 2. B : Type
| | 3. A List
| | 4. l2 : A List
| | ⊢  ∀L:(A+B) List.
| | |   mapoutl(L) = l2
| | |   ⇒
| | |   (∃L1,L2:(A+B) List.
| | |    L = (L1 @ L2) & mapoutl(L1) = nil & mapoutl(L2) = l2)
| | |  by Auto THEN (InstConcl [nil;L]) THEN (Reduce 0)
| | |
| | .\
| | . 1. A : Type
| | . 2. B : Type
| | . 3. A List
| | . 4. l2 : A List
| | . 5. u : A
| | . 6. v : A List
| | . 7. ∀L:(A+B) List.
| | . |  mapoutl(L) = (v @ l2)
| | . |  ⇒
| | . |  (∃L1,L2:(A+B) List.
| | . |   L = (L1 @ L2) & mapoutl(L1) = v & mapoutl(L2) = l2)
| | . ⊢  ∀L:(A+B) List.
| | . |  mapoutl(L) = [u / (v @ l2)]
| | . |  ⇒
| | . |  (∃L1,L2:(A+B) List.
| | . |   L = (L1 @ L2) & mapoutl(L1) = [u / v] & mapoutl(L2) = l2)
| | . | by InductionOnList THEN (Reduce 0)
| | . |
| | . |\
| | . | 1. A : Type
| | . | 2. B : Type
| | . | 3. A List
```

```
| . | 4. l2 : A List
| . | 5. u : A
| . | 6. v : A List
| . | 7. ∀L:(A+B) List.
| . |    mapoutl(L) = (v @ l2)
| . |    ⇒
| . |    (∃L1,L2:(A+B) List.
| . |     L = (L1 @ L2) & mapoutl(L1) = v & mapoutl(L2) = l2)
| . | 8. (A+B) List
| . | ⊢  nil = [u / (v @ l2)]
| . |    ⇒
| . |    (∃L1,L2:(A+B) List.
| . |     nil = (L1 @ L2) & mapoutl(L1) = [u / v] & mapoutl(L2) = l2)
| . |   by ObviousConcl
| . |
| . .\
| . . 1. A : Type
| . . 2. B : Type
| . . 3. A List
| . . 4. l2 : A List
| . . 5. u : A
| . . 6. v : A List
| . . 7. ∀L:(A+B) List.
| . . |  mapoutl(L) = (v @ l2)
| . . |  ⇒
| . . |  (∃L1,L2:(A+B) List.
| . . |   L = (L1 @ L2) & mapoutl(L1) = v & mapoutl(L2) = l2)
| . . 8. (A+B) List
| . . 9. u1 : A+B
| . . 10. v1 : (A+B) List
| . . 11. mapoutl(v1) = [u / (v @ l2)]
| . . |   ⇒
| . . |   (∃L1,L2:(A+B) List.
| . . |    v1 = (L1 @ L2) & mapoutl(L1) = [u / v] & mapoutl(L2) = l2)
| . . ⊢  mapoutl([u1 / v1]) = [u / (v @ l2)]
| . . |  ⇒
| . . |  (∃L1,L2:(A+B) List.
| . . |   [u1 / v1] = (L1 @ L2)
| . . |   & mapoutl(L1) = [u / v]
| . . |   & mapoutl(L2) = l2)  by (Analyze -3) THEN (Reduce 0)
| . . |
| . . |\
| . . | 1. A : Type
| . . | 2. B : Type
| . . | 3. A List
| . . | 4. l2 : A List
| . . | 5. u : A
| . . | 6. v : A List
| . . | 7. ∀L:(A+B) List.
| . . |    mapoutl(L) = (v @ l2)
| . . |    ⇒
| . . |    (∃L1,L2:(A+B) List.
| . . |     L = (L1 @ L2) & mapoutl(L1) = v & mapoutl(L2) = l2)
| . . | 8. (A+B) List
| . . | 9. x : A
```

```
| . . | 10. v1 : (A+B) List
| . . | 11. mapoutl(v1) = [u / (v @ l2)]
| . . |     ⇒
| . . |     (∃L1,L2:(A+B) List.
| . . |      v1 = (L1 @ L2)
| . . |      & mapoutl(L1) = [u / v]
| . . |      & mapoutl(L2) = l2)
| . . | 12. [x / mapoutl(v1)] = [u / (v @ l2)]
| . . | ⊢  ∃L1,L2:(A+B) List.
| . . |    [inl(x) / v1] = (L1 @ L2)
| . . |    & mapoutl(L1) = [u / v]
| . . |    & mapoutl(L2) = l2
| . . |   by (SplitCons -1) THEN (InstHyp [v1] 7) THEN ExRepD
| . . |      THEN
| . . |      (InstConcl [[inl(x) / L1];L2])
| . . |      THEN
| . . |      (Reduce 0)
| . . |      THEN
| . . |      Analyze
| . . |
| . . .\
| . . . 1. A : Type
| . . . 2. B : Type
| . . . 3. A List
| . . . 4. l2 : A List
| . . . 5. u : A
| . . . 6. v : A List
| . . . 7. ∀L:(A+B) List.
| . . .    mapoutl(L) = (v @ l2)
| . . .    ⇒
| . . .    (∃L1,L2:(A+B) List.
| . . .     L = (L1 @ L2) & mapoutl(L1) = v & mapoutl(L2) = l2)
| . . . 8. (A+B) List
| . . . 9. y : B
| . . . 10. v1 : (A+B) List
| . . . 11. mapoutl(v1) = [u / (v @ l2)]
| . . .     ⇒
| . . .     (∃L1,L2:(A+B) List.
| . . .      v1 = (L1 @ L2)
| . . .      & mapoutl(L1) = [u / v]
| . . .      & mapoutl(L2) = l2)
| . . . 12. mapoutl(v1) = [u / (v @ l2)]
| . . . ⊢  ∃L1,L2:(A+B) List.
| . . .    [inr(y) / v1] = (L1 @ L2)
| . . .    & mapoutl(L1) = [u / v]
| . . .    & mapoutl(L2) = l2
| . . .   by ThinTrivial THEN ExRepD
| . . .      THEN
| . . .      (InstConcl [[inr(y) / L1];L2])
| . . .      THEN
| . . .      (Reduce 0)
| . . .      THEN
| . . .      Analyze
|
.\
```

```
. 1. A : Type
. 2. B : Type
. 3. L : (A+B) List
. 4. l1 : A List
. 5. l2 : A List
. 6. L1 : (A+B) List
. 7. L2 : (A+B) List
. 8. L = (L1 @ L2)
. 9. mapoutl(L1) = l1
. 10. mapoutl(L2) = l2
. ⊢  mapoutl(L) = (l1 @ l2)
.   by (SubstFor L 0)
.      THEN
.      (RWO
.       Thm* ∀L1,L2:(A+B) List.
.            mapoutl(L1 @ L2) ∼ (mapoutl(L1) @ mapoutl(L2))
.       {mapoutl_append}
.       0)
.      THEN
.      Analyze
END
```