```
mapoutl_is_singleton  THM

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

```
| . | 3. (A+B) List
| . | 4. a : A
| . | 5. x : A
| . | 6. v : (A+B) List
| . | 7. mapoutl(v) = [a]
| . |    ⇒
| . |    (∃L1,L2:(A+B) List.
| . |     v = (L1 @ [inl(a)] @ L2)
| . |     & mapoutl(L1) = nil
| . |     & mapoutl(L2) = nil)
| . | 8. [x / mapoutl(v)] = [a]
| . | ⊢  ∃L1,L2:(A+B) List.
| . |    [inl(x) / v] = (L1 @ [inl(a) / L2])
| . |    & mapoutl(L1) = nil
| . |    & mapoutl(L2) = nil
| . |   by (SplitCons -1) THEN (InstConcl [nil;v]) THEN (Reduce 0)
| . |      THEN
| . |      Analyze
| . |      THEN
| . |      (ObviousFrom [-2])
| . |
| . .\
| . . 1. A : Type
| . . 2. B : Type
| . . 3. (A+B) List
| . . 4. a : A
| . . 5. y : B
| . . 6. v : (A+B) List
| . . 7. mapoutl(v) = [a]
| . . 8. ∃L1,L2:(A+B) List.
| . .    v = (L1 @ [inl(a)] @ L2)
| . .    & mapoutl(L1) = nil
| . .    & mapoutl(L2) = nil
| . . ⊢  ∃L1,L2:(A+B) List.
| . .    [inr(y) / v] = (L1 @ [inl(a) / L2])
| . .    & mapoutl(L1) = nil
| . .    & mapoutl(L2) = nil
| . .   by ExRepD THEN (InstConcl [[inr(y) / L1];L2])
| . .      THEN
| . .      (All Reduce)
| . .      THEN
| . .      Analyze
|
.\
. 1. A : Type
. 2. B : Type
. 3. L : (A+B) List
. 4. a : A
. 5. L1 : (A+B) List
. 6. L2 : (A+B) List
. 7. L = (L1 @ [inl(a)] @ L2)
. 8. mapoutl(L1) = nil
. 9. mapoutl(L2) = nil
. ⊢  mapoutl(L) = [a]
.   by (((SubstFor L 0)
```

```
.         THEN
.         (RWW
.          Thm* ∀L1,L2:(A+B) List.
.               mapoutl(L1 @ L2) ~ (mapoutl(L1) @ mapoutl(L2))
.          {mapoutl_append}
.          0))
.        THEN
.        (HypSubst -1 0)
.        THEN
.        (HypSubst -2 0))
.       THEN
.       (Reduce 0)
END
```