```
upto_decomp  THM

⊢  ∀m:ℕ, n:ℕ(m+1). upto(m) ~ (upto(n) @ map(λx.x+n;upto(m-n)))
| by InductionOnNat THEN Reduce 0 THEN Analyze 0
|
|\
| 1. n : ℕ1
| ⊢  nil ~ (upto(n) @ map(λx.x+n;upto(0-n)))
|   by Subst (n = 0) 0 THEN Reduce 0
|
.\
. 1. m : ℤ
. 2. 0<m
. 3. ∀n:ℕ(m-1+1). upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. 4. n : ℕ(m+1)
. ⊢  upto(m) ~ (upto(n) @ map(λx.x+n;upto(m-n)))  by Decide (n = m)
. |
. |\
. | 1. m : ℤ
. | 2. 0<m
. | 3. ∀n:ℕ(m-1+1). upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. | 4. n : ℕ(m+1)
. | 5. n = m
. | ⊢  upto(m) ~ (upto(n) @ map(λx.x+n;upto(m-n)))
. |   by Subst' (m-n = 0) 0 THEN Reduce 0 THEN Try (Complete Auto)
. |      THEN
. |      RWO Thm* ∀l:T List. (l @ nil) ~ l    {append_nil_sq} 0
. |      THEN
. |      Analyze
. |
. .\
. . 1. m : ℤ
. . 2. 0<m
. . 3. ∀n:ℕ(m-1+1). upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. . 4. n : ℕ(m+1)
. . 5. ¬n = m
. . ⊢  upto(m) ~ (upto(n) @ map(λx.x+n;upto(m-n)))
. . | by RW (AddrC [1] (RecUnfoldC `upto`)) 0 THEN SplitOnConclITE
. . |
. . |\
. . | .....truecase .....
. . | 1. m : ℤ
. . | 2. 0<m
. . | 3. ∀n:ℕ(m-1+1). upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. . | 4. n : ℕ(m+1)
. . | 5. ¬n = m
. . | 6. m = 0
. . | ⊢  nil ~ (upto(n) @ map(λx.x+n;upto(m-n)))
. . |   by Subst (n = 0) 0 THEN Reduce 0
. . |
. . .\
. . . .....falsecase .....
. . . 1. m : ℤ
. . . 2. 0<m
. . . 3. ∀n:ℕ(m-1+1). upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
```

```
. . . 4. n : ℕ(m+1)
. . . 5. ¬n = m
. . . 6. ¬m = 0
. . . ⊢  (upto(m-1) @ [(m-1)]) ~ (upto(n) @ map(λx.x+n;upto(m-n)))
. . . | by InstHyp [n] 3
. . . |    THEN
. . . |    Subst (upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))) 0
. . . |    THEN
. . . |    Try Trivial
. . . |
. . . 1. m : ℤ
. . . 2. 0<m
. . . 3. ∀n:ℕ(m-1+1). upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. . . 4. n : ℕ(m+1)
. . . 5. ¬n = m
. . . 6. ¬m = 0
. . . 7. upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. . . ⊢  ((upto(n) @ map(λx.x+n;upto(m-1-n))) @ [(m-1)]) ~ (upto(n)
. . . |  @ map(λx.x+n;upto(m-n)))
. . . | by RWO
. . . |    Thm* ∀as,bs,cs:Top List. ((as @ bs) @ cs) ~ (as @ bs @ cs)
. . . |    {append_assoc_sq}
. . . |    0
. . . |    THEN
. . . |    Analyze
. . . |    THEN
. . . |    Try Trivial
. . . |
. . . 1. m : ℤ
. . . 2. 0<m
. . . 3. ∀n:ℕ(m-1+1). upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. . . 4. n : ℕ(m+1)
. . . 5. ¬n = m
. . . 6. ¬m = 0
. . . 7. upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. . . ⊢  (map(λx.x+n;upto(m-1-n)) @ [(m-1)]) ~ map(λx.x+n;upto(m-n))
. . . | by RW (AddrC [2] (RecUnfoldC ‘upto‘)) 0 THEN SplitOnConclITE
. . . |
. . . |\
. . . | .....truecase .....
. . . | 1. m : ℤ
. . . | 2. 0<m
. . . | 3. ∀n:ℕ(m-1+1).
. . . |    upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. . . | 4. n : ℕ(m+1)
. . . | 5. ¬n = m
. . . | 6. ¬m = 0
. . . | 7. upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. . . | 8. m-n = 0
. . . | ⊢  (map(λx.x+n;upto(m-1-n)) @ [(m-1)]) ~ map(λx.x+n;nil)
. . . |   by Auto
. . . |
. . . .\
. . . . .....falsecase .....
. . . . 1. m : ℤ
```

```
. . . . . 2. 0<m
. . . . . 3. ∀n:ℕ(m-1+1).
. . . . . |  upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. . . . . 4. n : ℕ(m+1)
. . . . . 5. ¬n = m
. . . . . 6. ¬m = 0
. . . . . 7. upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. . . . . 8. ¬m-n = 0
. . . . . ⊢  (map(λx.x+n;upto(m-1-n)) @ [(m-1)]) ~ map(λx.x+n;upto(m-n
. . . . . |  -1)
. . . . . |  @ [(m-n-1)])
. . . . . | by RWO
. . . . . |    Thm* ∀f,as':Top, A:Type, as:A List.
. . . . . |         map(f;as @ as') ~ (map(f;as) @ map(f;as'))
. . . . . |    {map_append_sq}
. . . . . |    0
. . . . . |    THEN
. . . . . |    Analyze
. . . . . |
. . . . . |\
. . . . . | 1. m : ℤ
. . . . . | 2. 0<m
. . . . . | 3. ∀n:ℕ(m-1+1).
. . . . . |    upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. . . . . | 4. n : ℕ(m+1)
. . . . . | 5. ¬n = m
. . . . . | 6. ¬m = 0
. . . . . | 7. upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. . . . . | 8. ¬m-n = 0
. . . . . | ⊢  map(λx.x+n;upto(m-1-n)) ~ map(λx.x+n;upto(m-n-1))
. . . . . |   by RepeatFor 2 Analyze
. . . . . |
. . . . . .\
. . . . . . 1. m : ℤ
. . . . . . 2. 0<m
. . . . . . 3. ∀n:ℕ(m-1+1).
. . . . . .    upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. . . . . . 4. n : ℕ(m+1)
. . . . . . 5. ¬n = m
. . . . . . 6. ¬m = 0
. . . . . . 7. upto(m-1) ~ (upto(n) @ map(λx.x+n;upto(m-1-n)))
. . . . . . 8. ¬m-n = 0
. . . . . . ⊢  [(m-1)] ~ map(λx.x+n;[(m-n-1)])
. . . . . .   by Reduce 0 THEN Analyze
END
```