```
concat_map_upto  THM

⊢  ∀T:Type, f:(ℕ→(T List)), t,t':ℕ.
|  t<t' ⇒ concat(map(f;upto(t))) @ (f(t)) ≤ concat(map(f;upto(t')))
| by Auto
|    THEN
|    Subst
|    ((concat(map(f;upto(t))) @ (f(t))) ~ concat(map(f;upto(t+1))))
|    0
|
|\
| .....equality .....
| 1. T : Type
| 2. f : ℕ→(T List)
| 3. t : ℕ
| 4. t' : ℕ
| 5. t<t'
| ⊢  (concat(map(f;upto(t))) @ (f(t))) ~ concat(map(f;upto(t+1)))
| | by RW (AddrC [2] (RecUnfoldC `upto`)) 0 THEN SplitOnConclITE
| |
| |\
| | .....truecase .....
| | 1. T : Type
| | 2. f : ℕ→(T List)
| | 3. t : ℕ
| | 4. t' : ℕ
| | 5. t<t'
| | 6. t+1 = 0
| | ⊢  (concat(map(f;upto(t))) @ (f(t))) ~ concat(map(f;nil))
| |   by Auto
| |
| .\
| . .....falsecase .....
| . 1. T : Type
| . 2. f : ℕ→(T List)
| . 3. t : ℕ
| . 4. t' : ℕ
| . 5. t<t'
| . 6. ¬t+1 = 0
| . ⊢  (concat(map(f;upto(t))) @ (f(t))) ~ concat(map(f;upto(t+1-1)
| . |  @ [(t+1-1)]))  by Subst ((t+1-1) ~ t) 0
| . |
| . 1. T : Type
| . 2. f : ℕ→(T List)
| . 3. t : ℕ
| . 4. t' : ℕ
| . 5. t<t'
| . 6. ¬t+1 = 0
| . ⊢  (concat(map(f;upto(t))) @ (f(t))) ~ concat(map(f;upto(t)
| . |  @ [t]))
| . | 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
```

```
| . |
| . 1. T : Type
| . 2. f : ℕ→(T List)
| . 3. t : ℕ
| . 4. t' : ℕ
| . 5. t<t'
| . 6. ¬t+1 = 0
| . ⊢  (concat(map(f;upto(t))) @ (f(t))) ~ concat(map(f;upto(t))
| . |  @ map(f;[t]))
| . | by Reduce 0
| . |    THEN
| . |    RWO
| . |    Thm* ∀ll1,ll2:(Top List) List.
| . |         concat(ll1 @ ll2) ~ (concat(ll1) @ concat(ll2))
| . |    {concat_append}
| . |    0
| . |    THEN
| . |    Analyze
| . |
| . |\
| . | 1. T : Type
| . | 2. f : ℕ→(T List)
| . | 3. t : ℕ
| . | 4. t' : ℕ
| . | 5. t<t'
| . | 6. ¬t+1 = 0
| . | ⊢  concat(map(f;upto(t))) ~ concat(map(f;upto(t)))  by Auto
| . |
| . .\
| . . 1. T : Type
| . . 2. f : ℕ→(T List)
| . . 3. t : ℕ
| . . 4. t' : ℕ
| . . 5. t<t'
| . . 6. ¬t+1 = 0
| . . ⊢  (f(t)) ~ concat([(f(t))])
| . .   by RWO
| . .      Thm* ∀l:Top List, ll:(Top List) List.
| . .           concat([l / ll]) ~ (l @ concat(ll))
| . .      {concat_DASH_cons}
| . .      0
| . .      THEN
| . .      Reduce 0
| . .      THEN
| . .      RWO Thm* ∀l:T List. (l @ nil) ~ l    {append_nil_sq} 0
|
.\
. 1. T : Type
. 2. f : ℕ→(T List)
. 3. t : ℕ
. 4. t' : ℕ
. 5. t<t'
. ⊢  concat(map(f;upto(t+1))) ≤ concat(map(f;upto(t')))
. | by BackThru
. |    Thm* ∀ll1,ll2:(T List) List.
```

```
. |         ll1 ≤ ll2 ⇒ concat(ll1) ≤ concat(ll2)
. |    {concat_iseg}
. |
. 1. T : Type
. 2. f : ℕ→(T List)
. 3. t : ℕ
. 4. t' : ℕ
. 5. t<t'
. ⊢  map(f;upto(t+1)) ≤ map(f;upto(t'))
. | by BackThru
. |    Thm* ∀f:(A→B), L1,L2:A List. L1 ≤ L2 ⇒ map(f;L1) ≤ map(f;L2)
. |    {iseg_map}
. |
. 1. T : Type
. 2. ℕ→(T List)
. 3. t : ℕ
. 4. t' : ℕ
. 5. t<t'
. ⊢  upto(t+1) ≤ upto(t')
.   by BackThru Thm* ∀i,j:ℕ. i≤j ⇒ upto(i) ≤ upto(j)    {upto_iseg}
END
```