```
append_upto  THM

⊢  ∀i,j,k:ℤ. i≤j ⇒ j≤k ⇒ (upto(i;k) ~ (upto(i;j) @ upto(j;k)))
| by Assert
|    (∀d:ℕ, i,j,k:ℤ.
|     j-i = d ⇒ j≤k ⇒ (upto(i;k) ~ (upto(i;j) @ upto(j;k))))
|
|\
| .....assertion .....
| ⊢  ∀d:ℕ, i,j,k:ℤ.
| |  j-i = d ⇒ j≤k ⇒ (upto(i;k) ~ (upto(i;j) @ upto(j;k)))
| | by InductionOnNat THEN UnivCD
| |    THEN
| |    (RW
| |     (AddrC [1] (RecUnfoldC `upto`)
| |      THENC
| |      AddrC [2;1] (RecUnfoldC `upto`))
| |     0)
| |    THEN
| |    SplitOnConclITE
| |    THEN
| |    SplitOnConclITE
| |    THEN
| |    (Reduce 0)
| |    THEN
| |    (Try (Complete Auto))
| |
| |\
| | 1. i : ℤ
| | 2. j : ℤ
| | 3. k : ℤ
| | 4. j-i = 0
| | 5. j≤k
| | 6. i<k
| | 7. j≤i
| | ⊢  [i / upto(i+1;k)] ~ upto(j;k)
| |   by (RW (AddrC [2] (RecUnfoldC `upto`)) 0) THEN SplitOnConclITE
| |      THEN
| |      (Repeat Analyze)
| |
| |\
| | 1. i : ℤ
| | 2. j : ℤ
| | 3. k : ℤ
| | 4. j-i = 0
| | 5. j≤k
| | 6. k≤i
| | 7. j≤i
| | ⊢  nil ~ upto(j;k)  by (RecUnfold `upto` 0) THEN SplitOnConclITE
| |
| .\
| . 1. d : ℤ
| . 2. 0<d
| . 3. ∀i,j,k:ℤ.
| .    j-i = d-1 ⇒ j≤k ⇒ (upto(i;k) ~ (upto(i;j) @ upto(j;k)))
```

```
| . 4. i : ℤ
| . 5. j : ℤ
| . 6. k : ℤ
| . 7. j-i = d
| . 8. j≤k
| . 9. i<k
| . 10. i<j
| . ⊢  [i / upto(i+1;k)] ∼ [i / (upto(i+1;j) @ upto(j;k))]
| .   by Analyze THEN EasyHyp
|
.\
. 1. ∀d:ℕ, i,j,k:ℤ.
.     j-i = d ⇒ j≤k ⇒ (upto(i;k) ∼ (upto(i;j) @ upto(j;k)))
. ⊢  ∀i,j,k:ℤ. i≤j ⇒ j≤k ⇒ (upto(i;k) ∼ (upto(i;j) @ upto(j;k)))
.   by UnivCD THEN (InstHyp [j-i] 1) THEN EasyHyp
END
```