```
paren_order1  THM

⊢  ∀T:Type, s:(T+T) List.
|  paren(T;s)
|  ⇒
|  (∀i:T, s1,s2:(T+T) List. s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2))
| by (Analyze 0) THEN (BackThru Thm: paren_induction)
|
|\
| 1. T : Type
| ⊢  ∀i:T, s1,s2:(T+T) List.
|    nil = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)  by Obvious
|
|\
| 1. T : Type
| ⊢  ∀s1,s2:(T+T) List.
| |  (∀i:T, s1@0,s2:(T+T) List.
| |   s1 = (s1@0 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2))
| |  ⇒
| |  (∀i:T, s1,s2@0:(T+T) List.
| |   s2 = (s1 @ [inl(i)] @ s2@0) ⇒ (inr(i) ∈ s2@0))
| |  ⇒
| |  paren(T;s1)
| |  ⇒
| |  paren(T;s2)
| |  ⇒
| |  (∀i:T, s1@0,s2@0:(T+T) List.
| |   (s1 @ s2) = (s1@0 @ [inl(i)] @ s2@0) ⇒ (inr(i) ∈ s2@0))
| | by Auto THEN (FwdThru Thm: equal_appends [-1]) THEN (Analyze -1)
| |    THEN
| |    (Analyze -1)
| |
| |\
| | 1. T : Type
| | 2. s1 : (T+T) List
| | 3. s2 : (T+T) List
| | 4. ∀i:T, s1@0,s2:(T+T) List.
| | |  s1 = (s1@0 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
| | 5. ∀i:T, s1,s2@0:(T+T) List.
| | |  s2 = (s1 @ [inl(i)] @ s2@0) ⇒ (inr(i) ∈ s2@0)
| | 6. paren(T;s1)
| | 7. paren(T;s2)
| | 8. i : T
| | 9. s1@0 : (T+T) List
| | 10. s2@0 : (T+T) List
| | 11. (s1 @ s2) = (s1@0 @ [inl(i)] @ s2@0)
| | 12. e : (T+T) List
| | 13. s1 = (s1@0 @ e) & ([inl(i)] @ s2@0) = (e @ s2)
| | ⊢  (inr(i) ∈ s2@0)
| | | by (Analyze -1) THEN (FwdThru Thm: equal_appends [-1])
| | |    THEN
| | |    (Analyze -1)
| | |    THEN
| | |    (Analyze -1)
| | |
```

```
| | |\
| | | 1. T : Type
| | | 2. s1 : (T+T) List
| | | 3. s2 : (T+T) List
| | | 4. ∀i:T, s1@0,s2:(T+T) List.
| | | |  s1 = (s1@0 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
| | | 5. ∀i:T, s1,s2@0:(T+T) List.
| | | |  s2 = (s1 @ [inl(i)] @ s2@0) ⇒ (inr(i) ∈ s2@0)
| | | 6. paren(T;s1)
| | | 7. paren(T;s2)
| | | 8. i : T
| | | 9. s1@0 : (T+T) List
| | | 10. s2@0 : (T+T) List
| | | 11. (s1 @ s2) = (s1@0 @ [inl(i)] @ s2@0)
| | | 12. e : (T+T) List
| | | 13. s1 = (s1@0 @ e)
| | | 14. ([inl(i)] @ s2@0) = (e @ s2)
| | | 15. e1 : (T+T) List
| | | 16. [inl(i)] = (e @ e1) & s2 = (e1 @ s2@0)
| | | ⊢  (inr(i) ∈ s2@0)  by (Analyze -5) THEN (All Reduce)
| | | |
| | | |\
| | | | 1. T : Type
| | | | 2. s1 : (T+T) List
| | | | 3. s2 : (T+T) List
| | | | 4. ∀i:T, s1@0,s2:(T+T) List.
| | | | |  s1 = (s1@0 @ [inl(i) / s2]) ⇒ (inr(i) ∈ s2)
| | | | 5. ∀i:T, s1,s2@0:(T+T) List.
| | | | |  s2 = (s1 @ [inl(i) / s2@0]) ⇒ (inr(i) ∈ s2@0)
| | | | 6. paren(T;s1)
| | | | 7. paren(T;s2)
| | | | 8. i : T
| | | | 9. s1@0 : (T+T) List
| | | | 10. s2@0 : (T+T) List
| | | | 11. (s1 @ s2) = (s1@0 @ [inl(i) / s2@0])
| | | | 12. s1 = (s1@0 @ nil)
| | | | 13. [inl(i) / s2@0] = s2
| | | | 14. e1 : (T+T) List
| | | | 15. [inl(i)] = e1 & s2 = (e1 @ s2@0)
| | | | ⊢  (inr(i) ∈ s2@0)  by InstHyp [i;nil;s2@0] 5
| | | | |
| | | | 1. T : Type
| | | | 2. s1 : (T+T) List
| | | | 3. s2 : (T+T) List
| | | | 4. ∀i:T, s1@0,s2:(T+T) List.
| | | | |   s1 = (s1@0 @ [inl(i) / s2]) ⇒ (inr(i) ∈ s2)
| | | | 5. ∀i:T, s1,s2@0:(T+T) List.
| | | | |   s2 = (s1 @ [inl(i) / s2@0]) ⇒ (inr(i) ∈ s2@0)
| | | | 6. paren(T;s1)
| | | | 7. paren(T;s2)
| | | | 8. i : T
| | | | 9. s1@0 : (T+T) List
| | | | 10. s2@0 : (T+T) List
| | | | 11. (s1 @ s2) = (s1@0 @ [inl(i) / s2@0])
| | | | 12. s1 = (s1@0 @ nil)
```

```
| | | | 13. [inl(i) / s2@0] = s2
| | | | 14. e1 : (T+T) List
| | | | 15. [inl(i)] = e1
| | | | 16. s2 = (e1 @ s2@0)
| | | | ⊢  s2 = (nil @ [inl(i) / s2@0])
| | | |   by (Reduce 0) THEN (SubstFor e1 -1) THEN (Reduce -1)
| | | |
| | | .\
| | | . 1. T : Type
| | | . 2. s1 : (T+T) List
| | | . 3. s2 : (T+T) List
| | | . 4. ∀i:T, s1@0,s2:(T+T) List.
| | | . |  s1 = (s1@0 @ [inl(i) / s2]) ⇒ (inr(i) ∈ s2)
| | | . 5. ∀i:T, s1,s2@0:(T+T) List.
| | | . |  s2 = (s1 @ [inl(i) / s2@0]) ⇒ (inr(i) ∈ s2@0)
| | | . 6. paren(T;s1)
| | | . 7. paren(T;s2)
| | | . 8. i : T
| | | . 9. s1@0 : (T+T) List
| | | . 10. s2@0 : (T+T) List
| | | . 11. (s1 @ s2) = (s1@0 @ [inl(i) / s2@0])
| | | . 12. u : T+T
| | | . 13. v : (T+T) List
| | | . 14. s1 = (s1@0 @ [u / v])
| | | . 15. [inl(i) / s2@0] = [u / (v @ s2)]
| | | . 16. e1 : (T+T) List
| | | . 17. [inl(i)] = [u / (v @ e1)] & s2 = (e1 @ s2@0)
| | | . ⊢  (inr(i) ∈ s2@0)  by Auto THEN (SplitCons -2)
| | | . |
| | | . 1. T : Type
| | | . 2. s1 : (T+T) List
| | | . 3. s2 : (T+T) List
| | | . 4. ∀i:T, s1@0,s2:(T+T) List.
| | | . |  s1 = (s1@0 @ [inl(i) / s2]) ⇒ (inr(i) ∈ s2)
| | | . 5. ∀i:T, s1,s2@0:(T+T) List.
| | | . |  s2 = (s1 @ [inl(i) / s2@0]) ⇒ (inr(i) ∈ s2@0)
| | | . 6. paren(T;s1)
| | | . 7. paren(T;s2)
| | | . 8. i : T
| | | . 9. s1@0 : (T+T) List
| | | . 10. s2@0 : (T+T) List
| | | . 11. (s1 @ s2) = (s1@0 @ [inl(i) / s2@0])
| | | . 12. u : T+T
| | | . 13. v : (T+T) List
| | | . 14. s1 = (s1@0 @ [u / v])
| | | . 15. [inl(i) / s2@0] = [u / (v @ s2)]
| | | . 16. e1 : (T+T) List
| | | . 17. s2 = (e1 @ s2@0)
| | | . 18. inl(i) = u
| | | . 19. nil = (v @ e1)
| | | . ⊢  (inr(i) ∈ s2@0)  by InstHyp [i;s1@0;v] 4
| | | . |
| | | . |\
| | | . | .....antecedent .....
| | | . | 1. T : Type
```

```
| | | . | 2. s1 : (T+T) List
| | | . | 3. s2 : (T+T) List
| | | . | 4. ∀i:T, s1@0,s2:(T+T) List.
| | | . |    s1 = (s1@0 @ [inl(i) / s2]) ⇒ (inr(i) ∈ s2)
| | | . | 5. ∀i:T, s1,s2@0:(T+T) List.
| | | . |    s2 = (s1 @ [inl(i) / s2@0]) ⇒ (inr(i) ∈ s2@0)
| | | . | 6. paren(T;s1)
| | | . | 7. paren(T;s2)
| | | . | 8. i : T
| | | . | 9. s1@0 : (T+T) List
| | | . | 10. s2@0 : (T+T) List
| | | . | 11. (s1 @ s2) = (s1@0 @ [inl(i) / s2@0])
| | | . | 12. u : T+T
| | | . | 13. v : (T+T) List
| | | . | 14. s1 = (s1@0 @ [u / v])
| | | . | 15. [inl(i) / s2@0] = [u / (v @ s2)]
| | | . | 16. e1 : (T+T) List
| | | . | 17. s2 = (e1 @ s2@0)
| | | . | 18. inl(i) = u
| | | . | 19. nil = (v @ e1)
| | | . | ⊢  s1 = (s1@0 @ [inl(i) / v])
| | | . |   by (SubstFor u -6) THEN (Reduce -6)
| | | . |
| | | . .\
| | | . . 1. T : Type
| | | . . 2. s1 : (T+T) List
| | | . . 3. s2 : (T+T) List
| | | . . 4. ∀i:T, s1@0,s2:(T+T) List.
| | | . .    s1 = (s1@0 @ [inl(i) / s2]) ⇒ (inr(i) ∈ s2)
| | | . . 5. ∀i:T, s1,s2@0:(T+T) List.
| | | . .    s2 = (s1 @ [inl(i) / s2@0]) ⇒ (inr(i) ∈ s2@0)
| | | . . 6. paren(T;s1)
| | | . . 7. paren(T;s2)
| | | . . 8. i : T
| | | . . 9. s1@0 : (T+T) List
| | | . . 10. s2@0 : (T+T) List
| | | . . 11. (s1 @ s2) = (s1@0 @ [inl(i) / s2@0])
| | | . . 12. u : T+T
| | | . . 13. v : (T+T) List
| | | . . 14. s1 = (s1@0 @ [u / v])
| | | . . 15. [inl(i) / s2@0] = [u / (v @ s2)]
| | | . . 16. e1 : (T+T) List
| | | . . 17. s2 = (e1 @ s2@0)
| | | . . 18. inl(i) = u
| | | . . 19. nil = (v @ e1)
| | | . . 20. (inr(i) ∈ v)
| | | . . ⊢  (inr(i) ∈ s2@0)  by ObviousContradiction [-1;-2]
| | |
| | .\
| | . 1. T : Type
| | . 2. s1 : (T+T) List
| | . 3. s2 : (T+T) List
| | . 4. ∀i:T, s1@0,s2:(T+T) List.
| | . |  s1 = (s1@0 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
| | . 5. ∀i:T, s1,s2@0:(T+T) List.
```

```
| | . |  s2 = (s1 @ [inl(i)] @ s2@0) ⇒ (inr(i) ∈ s2@0)
| | . 6. paren(T;s1)
| | . 7. paren(T;s2)
| | . 8. i : T
| | . 9. s1@0 : (T+T) List
| | . 10. s2@0 : (T+T) List
| | . 11. (s1 @ s2) = (s1@0 @ [inl(i)] @ s2@0)
| | . 12. e : (T+T) List
| | . 13. s1 = (s1@0 @ e)
| | . 14. ([inl(i)] @ s2@0) = (e @ s2)
| | . 15. e1 : (T+T) List
| | . 16. e = ([inl(i)] @ e1) & s2@0 = (e1 @ s2)
| | . ⊢  (inr(i) ∈ s2@0)  by InstHyp [i;s1@0;e1] 4
| | . |
| | . |\
| | . | 1. T : Type
| | . | 2. s1 : (T+T) List
| | . | 3. s2 : (T+T) List
| | . | 4. ∀i:T, s1@0,s2:(T+T) List.
| | . |    s1 = (s1@0 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
| | . | 5. ∀i:T, s1,s2@0:(T+T) List.
| | . |    s2 = (s1 @ [inl(i)] @ s2@0) ⇒ (inr(i) ∈ s2@0)
| | . | 6. paren(T;s1)
| | . | 7. paren(T;s2)
| | . | 8. i : T
| | . | 9. s1@0 : (T+T) List
| | . | 10. s2@0 : (T+T) List
| | . | 11. (s1 @ s2) = (s1@0 @ [inl(i)] @ s2@0)
| | . | 12. e : (T+T) List
| | . | 13. s1 = (s1@0 @ e)
| | . | 14. ([inl(i)] @ s2@0) = (e @ s2)
| | . | 15. e1 : (T+T) List
| | . | 16. e = ([inl(i)] @ e1)
| | . | 17. s2@0 = (e1 @ s2)
| | . | ⊢  s1 = (s1@0 @ [inl(i)] @ e1)
| | . |   by (SubstFor s1 0) THEN (SubstFor e 0)
| | . |
| | . .\
| | . . 1. T : Type
| | . . 2. s1 : (T+T) List
| | . . 3. s2 : (T+T) List
| | . . 4. ∀i:T, s1@0,s2:(T+T) List.
| | . . |  s1 = (s1@0 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
| | . . 5. ∀i:T, s1,s2@0:(T+T) List.
| | . . |  s2 = (s1 @ [inl(i)] @ s2@0) ⇒ (inr(i) ∈ s2@0)
| | . . 6. paren(T;s1)
| | . . 7. paren(T;s2)
| | . . 8. i : T
| | . . 9. s1@0 : (T+T) List
| | . . 10. s2@0 : (T+T) List
| | . . 11. (s1 @ s2) = (s1@0 @ [inl(i)] @ s2@0)
| | . . 12. e : (T+T) List
| | . . 13. s1 = (s1@0 @ e)
| | . . 14. ([inl(i)] @ s2@0) = (e @ s2)
| | . . 15. e1 : (T+T) List
```

```
| | . . 16. e = ([inl(i)] @ e1)
| | . . 17. s2@0 = (e1 @ s2)
| | . . 18. (inr(i) ∈ e1)
| | . . ⊢  (inr(i) ∈ s2@0)  by HypSubst -2 0
| | . . |
| | . . 1. T : Type
| | . . 2. s1 : (T+T) List
| | . . 3. s2 : (T+T) List
| | . . 4. ∀i:T, s1@0,s2:(T+T) List.
| | . .    s1 = (s1@0 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
| | . . 5. ∀i:T, s1,s2@0:(T+T) List.
| | . .    s2 = (s1 @ [inl(i)] @ s2@0) ⇒ (inr(i) ∈ s2@0)
| | . . 6. paren(T;s1)
| | . . 7. paren(T;s2)
| | . . 8. i : T
| | . . 9. s1@0 : (T+T) List
| | . . 10. s2@0 : (T+T) List
| | . . 11. (s1 @ s2) = (s1@0 @ [inl(i)] @ s2@0)
| | . . 12. e : (T+T) List
| | . . 13. s1 = (s1@0 @ e)
| | . . 14. ([inl(i)] @ s2@0) = (e @ s2)
| | . . 15. e1 : (T+T) List
| | . . 16. e = ([inl(i)] @ e1)
| | . . 17. s2@0 = (e1 @ s2)
| | . . 18. (inr(i) ∈ e1)
| | . . ⊢  (inr(i) ∈ e1 @ s2)  by ObviousFrom [-1]
| |

| .\
| . 1. T : Type
| . 2. s1 : (T+T) List
| . 3. s2 : (T+T) List
| . 4. ∀i:T, s1@0,s2:(T+T) List.
| .    s1 = (s1@0 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
| . 5. ∀i:T, s1,s2@0:(T+T) List.
| .    s2 = (s1 @ [inl(i)] @ s2@0) ⇒ (inr(i) ∈ s2@0)
| . 6. paren(T;s1)
| . 7. paren(T;s2)
| . 8. i : T
| . 9. s1@0 : (T+T) List
| . 10. s2@0 : (T+T) List
| . 11. (s1 @ s2) = (s1@0 @ [inl(i)] @ s2@0)
| . 12. e : (T+T) List
| . 13. s1@0 = (s1 @ e) & s2 = (e @ [inl(i)] @ s2@0)
| . ⊢  (inr(i) ∈ s2@0)  by Obvious
|

.\
. 1. T : Type
. ⊢  ∀s:(T+T) List, t:T.
. |  (∀i:T, s1,s2:(T+T) List.
. |   s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2))
. |  ⇒
. |  paren(T;s)
. |  ⇒
. |  (∀i:T, s1,s2:(T+T) List.
. |   ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
```

```
. |   ⇒
. |   (inr(i) ∈ s2))  by Auto THEN (EqualAppendsD -1)
. |
. |\
. | 1. T : Type
. | 2. s : (T+T) List
. | 3. t : T
. | 4. ∀i:T, s1,s2:(T+T) List.
. | |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. | 5. paren(T;s)
. | 6. i : T
. | 7. s1 : (T+T) List
. | 8. s2 : (T+T) List
. | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. | 10. e : (T+T) List
. | 11. [inl(t)] = (s1 @ e)
. | 12. ([inl(i)] @ s2) = (e @ s @ [inr(t)])
. | ⊢  (inr(i) ∈ s2)
. | | by (FwdThru
. | |     Thm* ∀a,b:T List, x:T.
. | |          [x] = (a @ b) ⇒ a = nil & b = [x] ∨ b = nil & a = [x]
. | |     {append_is_singleton}
. | |     [-2])
. | |    THEN
. | |    (Analyze -1)
. | |    THEN
. | |    (Analyze -1)
. | |
. | |\
. | | 1. T : Type
. | | 2. s : (T+T) List
. | | 3. t : T
. | | 4. ∀i:T, s1,s2:(T+T) List.
. | | |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. | | 5. paren(T;s)
. | | 6. i : T
. | | 7. s1 : (T+T) List
. | | 8. s2 : (T+T) List
. | | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. | | 10. e : (T+T) List
. | | 11. [inl(t)] = (s1 @ e)
. | | 12. ([inl(i)] @ s2) = (e @ s @ [inr(t)])
. | | 13. s1 = nil
. | | 14. e = [inl(t)]
. | | ⊢  (inr(i) ∈ s2)
. | | | by (HypSubst -1 -3) THEN (Reduce -3) THEN (SplitCons -3)
. | | |    THEN
. | | |    (HypSubst -1 0)
. | | |
. | | 1. T : Type
. | | 2. s : (T+T) List
. | | 3. t : T
. | | 4. ∀i:T, s1,s2:(T+T) List.
. | |    s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. | | 5. paren(T;s)
```

```
. | | 6. i : T
. | | 7. s1 : (T+T) List
. | | 8. s2 : (T+T) List
. | | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. | | 10. e : (T+T) List
. | | 11. [inl(t)] = (s1 @ e)
. | | 12. s1 = nil
. | | 13. e = [inl(t)]
. | | 14. inl(i) = inl(t)
. | | 15. s2 = (s @ [inr(t)])
. | | ⊢  (inr(i) ∈ s @ [inr(t)])  by ObviousFrom [-2]
. | |
. | .\
. | . 1. T : Type
. | . 2. s : (T+T) List
. | . 3. t : T
. | . 4. ∀i:T, s1,s2:(T+T) List.
. | . |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. | . 5. paren(T;s)
. | . 6. i : T
. | . 7. s1 : (T+T) List
. | . 8. s2 : (T+T) List
. | . 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. | . 10. e : (T+T) List
. | . 11. [inl(t)] = (s1 @ e)
. | . 12. ([inl(i)] @ s2) = (e @ s @ [inr(t)])
. | . 13. e = nil
. | . 14. s1 = [inl(t)]
. | . ⊢  (inr(i) ∈ s2)  by Assert (([inl(i)] @ s2) = (s @ [inr(t)]))
. | . |
. | . |\
. | . | .....assertion .....
. | . | 1. T : Type
. | . | 2. s : (T+T) List
. | . | 3. t : T
. | . | 4. ∀i:T, s1,s2:(T+T) List.
. | . |    s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. | . | 5. paren(T;s)
. | . | 6. i : T
. | . | 7. s1 : (T+T) List
. | . | 8. s2 : (T+T) List
. | . | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. | . | 10. e : (T+T) List
. | . | 11. [inl(t)] = (s1 @ e)
. | . | 12. ([inl(i)] @ s2) = (e @ s @ [inr(t)])
. | . | 13. e = nil
. | . | 14. s1 = [inl(t)]
. | . | ⊢  ([inl(i)] @ s2) = (s @ [inr(t)])
. | . |   by (HypSubst -2 -3) THEN (All Reduce)
. | . |
. | . .\
. | . . 1. T : Type
. | . . 2. s : (T+T) List
. | . . 3. t : T
. | . . 4. ∀i:T, s1,s2:(T+T) List.
```

```
. | . . |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. | . . 5. paren(T;s)
. | . . 6. i : T
. | . . 7. s1 : (T+T) List
. | . . 8. s2 : (T+T) List
. | . . 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. | . . 10. e : (T+T) List
. | . . 11. [inl(t)] = (s1 @ e)
. | . . 12. ([inl(i)] @ s2) = (e @ s @ [inr(t)])
. | . . 13. e = nil
. | . . 14. s1 = [inl(t)]
. | . . 15. ([inl(i)] @ s2) = (s @ [inr(t)])
. | . . ⊢  (inr(i) ∈ s2)  by EqualAppendsD -1
. | . . |
. | . . |\
. | . . | 1. T : Type
. | . . | 2. s : (T+T) List
. | . . | 3. t : T
. | . . | 4. ∀i:T, s1,s2:(T+T) List.
. | . . | |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. | . . | 5. paren(T;s)
. | . . | 6. i : T
. | . . | 7. s1 : (T+T) List
. | . . | 8. s2 : (T+T) List
. | . . | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. | . . | 10. e : (T+T) List
. | . . | 11. [inl(t)] = (s1 @ e)
. | . . | 12. ([inl(i)] @ s2) = (e @ s @ [inr(t)])
. | . . | 13. e = nil
. | . . | 14. s1 = [inl(t)]
. | . . | 15. ([inl(i)] @ s2) = (s @ [inr(t)])
. | . . | 16. e1 : (T+T) List
. | . . | 17. [inl(i)] = (s @ e1)
. | . . | 18. [inr(t)] = (e1 @ s2)
. | . . | ⊢  (inr(i) ∈ s2)
. | . . | | by (SingletonAppendD -2) THEN (SubstFor e1 -3)
. | . . | |    THEN
. | . . | |    (Reduce -3)
. | . . | |
. | . . | |\
. | . . | | 1. T : Type
. | . . | | 2. s : (T+T) List
. | . . | | 3. t : T
. | . . | | 4. ∀i:T, s1,s2:(T+T) List.
. | . . | |    s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. | . . | | 5. paren(T;s)
. | . . | | 6. i : T
. | . . | | 7. s1 : (T+T) List
. | . . | | 8. s2 : (T+T) List
. | . . | | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. | . . | | 10. e : (T+T) List
. | . . | | 11. [inl(t)] = (s1 @ e)
. | . . | | 12. ([inl(i)] @ s2) = (e @ s @ [inr(t)])
. | . . | | 13. e = nil
. | . . | | 14. s1 = [inl(t)]
```

```
. | . . | | 15. ([inl(i)] @ s2) = (s @ [inr(t)])
. | . . | | 16. e1 : (T+T) List
. | . . | | 17. [inl(i)] = (s @ e1)
. | . . | | 18. [inr(t)] = [inl(i) / s2]
. | . . | | 19. s = nil
. | . . | | 20. e1 = [inl(i)]
. | . . | | ⊢  (inr(i) ∈ s2)  by ObviousContradiction [-3]

. | . . | |
. | . . | .\
. | . . | . 1. T : Type
. | . . | . 2. s : (T+T) List
. | . . | . 3. t : T
. | . . | . 4. ∀i:T, s1,s2:(T+T) List.
. | . . | . |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. | . . | . 5. paren(T;s)
. | . . | . 6. i : T
. | . . | . 7. s1 : (T+T) List
. | . . | . 8. s2 : (T+T) List
. | . . | . 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. | . . | . 10. e : (T+T) List
. | . . | . 11. [inl(t)] = (s1 @ e)
. | . . | . 12. ([inl(i)] @ s2) = (e @ s @ [inr(t)])
. | . . | . 13. e = nil
. | . . | . 14. s1 = [inl(t)]
. | . . | . 15. ([inl(i)] @ s2) = (s @ [inr(t)])
. | . . | . 16. e1 : (T+T) List
. | . . | . 17. [inl(i)] = (s @ e1)
. | . . | . 18. [inr(t)] = s2
. | . . | . 19. e1 = nil
. | . . | . 20. s = [inl(i)]
. | . . | . ⊢  (inr(i) ∈ s2)
. | . . | . | by (InstHyp [i;nil;nil] 4) THEN (All Reduce)
. | . . | . |
. | . . | . 1. T : Type
. | . . | . 2. s : (T+T) List
. | . . | . 3. t : T
. | . . | . 4. ∀i:T, s1,s2:(T+T) List.
. | . . | .    s = (s1 @ [inl(i) / s2]) ⇒ (inr(i) ∈ s2)
. | . . | . 5. paren(T;s)
. | . . | . 6. i : T
. | . . | . 7. s1 : (T+T) List
. | . . | . 8. s2 : (T+T) List
. | . . | . 9. [inl(t) / (s @ [inr(t)])] = (s1 @ [inl(i) / s2])
. | . . | . 10. e : (T+T) List
. | . . | . 11. [inl(t)] = (s1 @ e)
. | . . | . 12. [inl(i) / s2] = (e @ s @ [inr(t)])
. | . . | . 13. e = nil
. | . . | . 14. s1 = [inl(t)]
. | . . | . 15. [inl(i) / s2] = (s @ [inr(t)])
. | . . | . 16. e1 : (T+T) List
. | . . | . 17. [inl(i)] = (s @ e1)
. | . . | . 18. [inr(t)] = s2
. | . . | . 19. e1 = nil
. | . . | . 20. s = [inl(i)]
. | . . | . 21. (inr(i) ∈ nil)
```

```
. | . . | . ⊢  (inr(i) ∈ s2)  by ObviousContradiction [-1]
. | . . |
. | . . .\
. | . . . 1. T : Type
. | . . . 2. s : (T+T) List
. | . . . 3. t : T
. | . . . 4. ∀i:T, s1,s2:(T+T) List.
. | . . . |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. | . . . 5. paren(T;s)
. | . . . 6. i : T
. | . . . 7. s1 : (T+T) List
. | . . . 8. s2 : (T+T) List
. | . . . 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. | . . . 10. e : (T+T) List
. | . . . 11. [inl(t)] = (s1 @ e)
. | . . . 12. ([inl(i)] @ s2) = (e @ s @ [inr(t)])
. | . . . 13. e = nil
. | . . . 14. s1 = [inl(t)]
. | . . . 15. ([inl(i)] @ s2) = (s @ [inr(t)])
. | . . . 16. e1 : (T+T) List
. | . . . 17. s = ([inl(i)] @ e1)
. | . . . 18. s2 = (e1 @ [inr(t)])
. | . . . ⊢  (inr(i) ∈ s2)
. | . . . | by (InstHyp [i;nil;e1] 4) THEN (All Reduce)
. | . . . |    THEN
. | . . . |    (SubstFor s2 0)
. | . . . |
. | . . . 1. T : Type
. | . . . 2. s : (T+T) List
. | . . . 3. t : T
. | . . . 4. ∀i:T, s1,s2:(T+T) List.
. | . . .    s = (s1 @ [inl(i) / s2]) ⇒ (inr(i) ∈ s2)
. | . . . 5. paren(T;s)
. | . . . 6. i : T
. | . . . 7. s1 : (T+T) List
. | . . . 8. s2 : (T+T) List
. | . . . 9. [inl(t) / (s @ [inr(t)])] = (s1 @ [inl(i) / s2])
. | . . . 10. e : (T+T) List
. | . . . 11. [inl(t)] = (s1 @ e)
. | . . . 12. [inl(i) / s2] = (e @ s @ [inr(t)])
. | . . . 13. e = nil
. | . . . 14. s1 = [inl(t)]
. | . . . 15. [inl(i) / s2] = (s @ [inr(t)])
. | . . . 16. e1 : (T+T) List
. | . . . 17. s = [inl(i) / e1]
. | . . . 18. s2 = (e1 @ [inr(t)])
. | . . . 19. (inr(i) ∈ e1)
. | . . . ⊢  (inr(i) ∈ e1 @ [inr(t)])  by ObviousFrom [-1]
. |
. .\
. . 1. T : Type
. . 2. s : (T+T) List
. . 3. t : T
. . 4. ∀i:T, s1,s2:(T+T) List.
. . |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
```

```
. . 5. paren(T;s)
. . 6. i : T
. . 7. s1 : (T+T) List
. . 8. s2 : (T+T) List
. . 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . 10. e : (T+T) List
. . 11. s1 = ([inl(t)] @ e)
. . 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . ⊢  (inr(i) ∈ s2)  by (EqualAppendsD -1) THEN (Analyze -1)
. . |
. . |\
. . | 1. T : Type
. . | 2. s : (T+T) List
. . | 3. t : T
. . | 4. ∀i:T, s1,s2:(T+T) List.
. . | |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . | 5. paren(T;s)
. . | 6. i : T
. . | 7. s1 : (T+T) List
. . | 8. s2 : (T+T) List
. . | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . | 10. e : (T+T) List
. . | 11. s1 = ([inl(t)] @ e)
. . | 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . | 13. e1 : (T+T) List
. . | 14. s = (e @ e1)
. . | 15. ([inl(i)] @ s2) = (e1 @ [inr(t)])
. . | ⊢  (inr(i) ∈ s2)  by (EqualAppendsD -1) THEN (Analyze -1)
. . | |
. . | |\
. . | | 1. T : Type
. . | | 2. s : (T+T) List
. . | | 3. t : T
. . | | 4. ∀i:T, s1,s2:(T+T) List.
. . | | |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . | | 5. paren(T;s)
. . | | 6. i : T
. . | | 7. s1 : (T+T) List
. . | | 8. s2 : (T+T) List
. . | | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . | | 10. e : (T+T) List
. . | | 11. s1 = ([inl(t)] @ e)
. . | | 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . | | 13. e1 : (T+T) List
. . | | 14. s = (e @ e1)
. . | | 15. ([inl(i)] @ s2) = (e1 @ [inr(t)])
. . | | 16. e2 : (T+T) List
. . | | 17. [inl(i)] = (e1 @ e2)
. . | | 18. [inr(t)] = (e2 @ s2)
. . | | ⊢  (inr(i) ∈ s2)
. . | | | by (SingletonAppendD -2) THEN (SingletonAppendD -3)
. . | | |
. . | | |\
. . | | | 1. T : Type
. . | | | 2. s : (T+T) List
```

```
. . | | | 3. t : T
. . | | | 4. ∀i:T, s1,s2:(T+T) List.
. . | | |    s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . | | | 5. paren(T;s)
. . | | | 6. i : T
. . | | | 7. s1 : (T+T) List
. . | | | 8. s2 : (T+T) List
. . | | | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . | | | 10. e : (T+T) List
. . | | | 11. s1 = ([inl(t)] @ e)
. . | | | 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . | | | 13. e1 : (T+T) List
. . | | | 14. s = (e @ e1)
. . | | | 15. ([inl(i)] @ s2) = (e1 @ [inr(t)])
. . | | | 16. e2 : (T+T) List
. . | | | 17. [inl(i)] = (e1 @ e2)
. . | | | 18. [inr(t)] = (e2 @ s2)
. . | | | 19. e1 = nil
. . | | | 20. e2 = [inl(i)]
. . | | | 21. e2 = nil
. . | | | 22. s2 = [inr(t)]
. . | | | ⊢  (inr(i) ∈ s2)  by ObviousFrom [-1;-2;-3;-4]
. . | | |
. . | | |\
. . | | | 1. T : Type
. . | | | 2. s : (T+T) List
. . | | | 3. t : T
. . | | | 4. ∀i:T, s1,s2:(T+T) List.
. . | | | |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . | | | 5. paren(T;s)
. . | | | 6. i : T
. . | | | 7. s1 : (T+T) List
. . | | | 8. s2 : (T+T) List
. . | | | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . | | | 10. e : (T+T) List
. . | | | 11. s1 = ([inl(t)] @ e)
. . | | | 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . | | | 13. e1 : (T+T) List
. . | | | 14. s = (e @ e1)
. . | | | 15. ([inl(i)] @ s2) = (e1 @ [inr(t)])
. . | | | 16. e2 : (T+T) List
. . | | | 17. [inl(i)] = (e1 @ e2)
. . | | | 18. [inr(t)] = (e2 @ s2)
. . | | | 19. e1 = nil
. . | | | 20. e2 = [inl(i)]
. . | | | 21. s2 = nil
. . | | | 22. e2 = [inr(t)]
. . | | | ⊢  (inr(i) ∈ s2)  by HypSubst -3 -1
. . | | | |
. . | | | 1. T : Type
. . | | | 2. s : (T+T) List
. . | | | 3. t : T
. . | | | 4. ∀i:T, s1,s2:(T+T) List.
. . | | |    s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . | | | 5. paren(T;s)
```

```
. . | | | 6. i : T
. . | | | 7. s1 : (T+T) List
. . | | | 8. s2 : (T+T) List
. . | | | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . | | | 10. e : (T+T) List
. . | | | 11. s1 = ([inl(t)] @ e)
. . | | | 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . | | | 13. e1 : (T+T) List
. . | | | 14. s = (e @ e1)
. . | | | 15. ([inl(i)] @ s2) = (e1 @ [inr(t)])
. . | | | 16. e2 : (T+T) List
. . | | | 17. [inl(i)] = (e1 @ e2)
. . | | | 18. [inr(t)] = (e2 @ s2)
. . | | | 19. e1 = nil
. . | | | 20. e2 = [inl(i)]
. . | | | 21. s2 = nil
. . | | | 22. [inl(i)] = [inr(t)]
. . | | | ⊢  (inr(i) ∈ s2)  by ObviousFrom [-1]
. . | | |
. . | | |\
. . | | | 1. T : Type
. . | | | 2. s : (T+T) List
. . | | | 3. t : T
. . | | | 4. ∀i:T, s1,s2:(T+T) List.
. . | | | |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . | | | 5. paren(T;s)
. . | | | 6. i : T
. . | | | 7. s1 : (T+T) List
. . | | | 8. s2 : (T+T) List
. . | | | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . | | | 10. e : (T+T) List
. . | | | 11. s1 = ([inl(t)] @ e)
. . | | | 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . | | | 13. e1 : (T+T) List
. . | | | 14. s = (e @ e1)
. . | | | 15. ([inl(i)] @ s2) = (e1 @ [inr(t)])
. . | | | 16. e2 : (T+T) List
. . | | | 17. [inl(i)] = (e1 @ e2)
. . | | | 18. [inr(t)] = (e2 @ s2)
. . | | | 19. e2 = nil
. . | | | 20. e1 = [inl(i)]
. . | | | 21. e2 = nil
. . | | | 22. s2 = [inr(t)]
. . | | | ⊢  (inr(i) ∈ s2)  by InstHyp [i;e;nil] 4
. . | | | |
. . | | | |\
. . | | | | .....antecedent .....
. . | | | | 1. T : Type
. . | | | | 2. s : (T+T) List
. . | | | | 3. t : T
. . | | | | 4. ∀i:T, s1,s2:(T+T) List.
. . | | | | |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . | | | | 5. paren(T;s)
. . | | | | 6. i : T
. . | | | | 7. s1 : (T+T) List
```

```
. . | | | | 8. s2 : (T+T) List
. . | | | | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . | | | | 10. e : (T+T) List
. . | | | | 11. s1 = ([inl(t)] @ e)
. . | | | | 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . | | | | 13. e1 : (T+T) List
. . | | | | 14. s = (e @ e1)
. . | | | | 15. ([inl(i)] @ s2) = (e1 @ [inr(t)])
. . | | | | 16. e2 : (T+T) List
. . | | | | 17. [inl(i)] = (e1 @ e2)
. . | | | | 18. [inr(t)] = (e2 @ s2)
. . | | | | 19. e2 = nil
. . | | | | 20. e1 = [inl(i)]
. . | | | | 21. e2 = nil
. . | | | | 22. s2 = [inr(t)]
. . | | | | ⊢  s = (e @ [inl(i)] @ nil)
. . | | | | | by (SubstFor s 0) THEN Analyze
. . | | | | |
. . | | | | .....subterm .....
. . | | | | 1. T : Type
. . | | | | 2. s : (T+T) List
. . | | | | 3. t : T
. . | | | | 4. ∀i:T, s1,s2:(T+T) List.
. . | | | |    s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . | | | | 5. paren(T;s)
. . | | | | 6. i : T
. . | | | | 7. s1 : (T+T) List
. . | | | | 8. s2 : (T+T) List
. . | | | | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . | | | | 10. e : (T+T) List
. . | | | | 11. s1 = ([inl(t)] @ e)
. . | | | | 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . | | | | 13. e1 : (T+T) List
. . | | | | 14. s = (e @ e1)
. . | | | | 15. ([inl(i)] @ s2) = (e1 @ [inr(t)])
. . | | | | 16. e2 : (T+T) List
. . | | | | 17. [inl(i)] = (e1 @ e2)
. . | | | | 18. [inr(t)] = (e2 @ s2)
. . | | | | 19. e2 = nil
. . | | | | 20. e1 = [inl(i)]
. . | | | | 21. e2 = nil
. . | | | | 22. s2 = [inr(t)]
. . | | | | ⊢  e1 = ([inl(i)] @ nil)  by ObviousFrom [-3]
. . | | | |
. . | | | .\
. . | | | . 1. T : Type
. . | | | . 2. s : (T+T) List
. . | | | . 3. t : T
. . | | | . 4. ∀i:T, s1,s2:(T+T) List.
. . | | | .    s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . | | | . 5. paren(T;s)
. . | | | . 6. i : T
. . | | | . 7. s1 : (T+T) List
. . | | | . 8. s2 : (T+T) List
. . | | | . 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
```

```
. . | | | . 10. e : (T+T) List
. . | | | . 11. s1 = ([inl(t)] @ e)
. . | | | . 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . | | | . 13. e1 : (T+T) List
. . | | | . 14. s = (e @ e1)
. . | | | . 15. ([inl(i)] @ s2) = (e1 @ [inr(t)])
. . | | | . 16. e2 : (T+T) List
. . | | | . 17. [inl(i)] = (e1 @ e2)
. . | | | . 18. [inr(t)] = (e2 @ s2)
. . | | | . 19. e2 = nil
. . | | | . 20. e1 = [inl(i)]
. . | | | . 21. e2 = nil
. . | | | . 22. s2 = [inr(t)]
. . | | | . 23. (inr(i) ∈ nil)
. . | | | . ⊢  (inr(i) ∈ s2)  by ObviousContradiction [-1]
. . | | |
. . | | .\
. . | | . 1. T : Type
. . | | . 2. s : (T+T) List
. . | | . 3. t : T
. . | | . 4. ∀i:T, s1,s2:(T+T) List.
. . | | .    s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . | | . 5. paren(T;s)
. . | | . 6. i : T
. . | | . 7. s1 : (T+T) List
. . | | . 8. s2 : (T+T) List
. . | | . 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . | | . 10. e : (T+T) List
. . | | . 11. s1 = ([inl(t)] @ e)
. . | | . 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . | | . 13. e1 : (T+T) List
. . | | . 14. s = (e @ e1)
. . | | . 15. ([inl(i)] @ s2) = (e1 @ [inr(t)])
. . | | . 16. e2 : (T+T) List
. . | | . 17. [inl(i)] = (e1 @ e2)
. . | | . 18. [inr(t)] = (e2 @ s2)
. . | | . 19. e2 = nil
. . | | . 20. e1 = [inl(i)]
. . | | . 21. s2 = nil
. . | | . 22. e2 = [inr(t)]
. . | | . ⊢  (inr(i) ∈ s2)  by ObviousFrom [-1;-2;-3;-4]
. . | |
. . | .\
. . | . 1. T : Type
. . | . 2. s : (T+T) List
. . | . 3. t : T
. . | . 4. ∀i:T, s1,s2:(T+T) List.
. . | . |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . | . 5. paren(T;s)
. . | . 6. i : T
. . | . 7. s1 : (T+T) List
. . | . 8. s2 : (T+T) List
. . | . 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . | . 10. e : (T+T) List
. . | . 11. s1 = ([inl(t)] @ e)
```

```
. . | . 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . | . 13. e1 : (T+T) List
. . | . 14. s = (e @ e1)
. . | . 15. ([inl(i)] @ s2) = (e1 @ [inr(t)])
. . | . 16. e2 : (T+T) List
. . | . 17. e1 = ([inl(i)] @ e2)
. . | . 18. s2 = (e2 @ [inr(t)])
. . | . ⊢  (inr(i) ∈ s2)  by InstHyp [i;e;e2] 4
. . | . |
. . | . |\
. . | . | .....antecedent .....
. . | . | 1. T : Type
. . | . | 2. s : (T+T) List
. . | . | 3. t : T
. . | . | 4. ∀i:T, s1,s2:(T+T) List.
. . | . |    s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . | . | 5. paren(T;s)
. . | . | 6. i : T
. . | . | 7. s1 : (T+T) List
. . | . | 8. s2 : (T+T) List
. . | . | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . | . | 10. e : (T+T) List
. . | . | 11. s1 = ([inl(t)] @ e)
. . | . | 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . | . | 13. e1 : (T+T) List
. . | . | 14. s = (e @ e1)
. . | . | 15. ([inl(i)] @ s2) = (e1 @ [inr(t)])
. . | . | 16. e2 : (T+T) List
. . | . | 17. e1 = ([inl(i)] @ e2)
. . | . | 18. s2 = (e2 @ [inr(t)])
. . | . | ⊢  s = (e @ [inl(i)] @ e2)  by (SubstFor s 0) THEN Analyze
. . | . |
. . | . .\
. . | . . 1. T : Type
. . | . . 2. s : (T+T) List
. . | . . 3. t : T
. . | . . 4. ∀i:T, s1,s2:(T+T) List.
. . | . . |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . | . . 5. paren(T;s)
. . | . . 6. i : T
. . | . . 7. s1 : (T+T) List
. . | . . 8. s2 : (T+T) List
. . | . . 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . | . . 10. e : (T+T) List
. . | . . 11. s1 = ([inl(t)] @ e)
. . | . . 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . | . . 13. e1 : (T+T) List
. . | . . 14. s = (e @ e1)
. . | . . 15. ([inl(i)] @ s2) = (e1 @ [inr(t)])
. . | . . 16. e2 : (T+T) List
. . | . . 17. e1 = ([inl(i)] @ e2)
. . | . . 18. s2 = (e2 @ [inr(t)])
. . | . . 19. (inr(i) ∈ e2)
. . | . . ⊢  (inr(i) ∈ s2)  by HypSubst -2 0
. . | . . |
```

```
. . | . . 1. T : Type
. . | . . 2. s : (T+T) List
. . | . . 3. t : T
. . | . . 4. ∀i:T, s1,s2:(T+T) List.
. . | . .    s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . | . . 5. paren(T;s)
. . | . . 6. i : T
. . | . . 7. s1 : (T+T) List
. . | . . 8. s2 : (T+T) List
. . | . . 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . | . . 10. e : (T+T) List
. . | . . 11. s1 = ([inl(t)] @ e)
. . | . . 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . | . . 13. e1 : (T+T) List
. . | . . 14. s = (e @ e1)
. . | . . 15. ([inl(i)] @ s2) = (e1 @ [inr(t)])
. . | . . 16. e2 : (T+T) List
. . | . . 17. e1 = ([inl(i)] @ e2)
. . | . . 18. s2 = (e2 @ [inr(t)])
. . | . . 19. (inr(i) ∈ e2)
. . | . . ⊢  (inr(i) ∈ e2 @ [inr(t)])  by ObviousFrom [-1]
. . |
. . .\
. . . 1. T : Type
. . . 2. s : (T+T) List
. . . 3. t : T
. . . 4. ∀i:T, s1,s2:(T+T) List.
. . . |  s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . . 5. paren(T;s)
. . . 6. i : T
. . . 7. s1 : (T+T) List
. . . 8. s2 : (T+T) List
. . . 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . . 10. e : (T+T) List
. . . 11. s1 = ([inl(t)] @ e)
. . . 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . . 13. e1 : (T+T) List
. . . 14. e = (s @ e1)
. . . 15. [inr(t)] = (e1 @ [inl(i)] @ s2)
. . . ⊢  (inr(i) ∈ s2)  by SingletonAppendD -1
. . . |
. . . |\
. . . | 1. T : Type
. . . | 2. s : (T+T) List
. . . | 3. t : T
. . . | 4. ∀i:T, s1,s2:(T+T) List.
. . . |    s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . . | 5. paren(T;s)
. . . | 6. i : T
. . . | 7. s1 : (T+T) List
. . . | 8. s2 : (T+T) List
. . . | 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . . | 10. e : (T+T) List
. . . | 11. s1 = ([inl(t)] @ e)
. . . | 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
```

```
. . . | 13. e1 : (T+T) List
. . . | 14. e = (s @ e1)
. . . | 15. [inr(t)] = (e1 @ [inl(i)] @ s2)
. . . | 16. e1 = nil
. . . | 17. ([inl(i)] @ s2) = [inr(t)]
. . . | ⊢  (inr(i) ∈ s2)  by ObviousContradiction [-1]
. . . |
. . . .\
. . . . 1. T : Type
. . . . 2. s : (T+T) List
. . . . 3. t : T
. . . . 4. ∀i:T, s1,s2:(T+T) List.

. . . .    s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2)
. . . . 5. paren(T;s)
. . . . 6. i : T
. . . . 7. s1 : (T+T) List
. . . . 8. s2 : (T+T) List
. . . . 9. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(i)] @ s2)
. . . . 10. e : (T+T) List
. . . . 11. s1 = ([inl(t)] @ e)
. . . . 12. (s @ [inr(t)]) = (e @ [inl(i)] @ s2)
. . . . 13. e1 : (T+T) List
. . . . 14. e = (s @ e1)
. . . . 15. [inr(t)] = (e1 @ [inl(i)] @ s2)
. . . . 16. ([inl(i)] @ s2) = nil
. . . . 17. e1 = [inr(t)]
. . . . ⊢  (inr(i) ∈ s2)  by ObviousContradiction [-2]
END
```