```
paren_interval  THM

⊢  ∀T:Type, s:(T+T) List.
|  paren(T;s)
|  ⇒
|  no_repeats(T+T;s)
|  ⇒
|  (∀s1,s2,s3:(T+T) List, x:T.
|   s = (s1 @ [inl(x)] @ s2 @ [inr(x)] @ s3) ⇒ paren(T;s2))
| by (Analyze 0) THEN (BackThru Thm: paren_induction)
|
|\
| 1. T : Type
| ⊢  no_repeats(T+T;nil)
|    ⇒
|    (∀s1,s2,s3:(T+T) List, x:T.
|     nil = (s1 @ [inl(x)] @ s2 @ [inr(x)] @ s3) ⇒ paren(T;s2))
|   by Obvious
|
|\
| 1. T : Type
| ⊢  ∀s1,s2:(T+T) List.
| |  (no_repeats(T+T;s1)
| |   ⇒
| |   (∀s1@0,s2,s3:(T+T) List, x:T.
| |    s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3) ⇒ paren(T;s2)))
| |  ⇒
| |  (no_repeats(T+T;s2)
| |   ⇒
| |   (∀s1,s2@0,s3:(T+T) List, x:T.
| |    s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3) ⇒ paren(T;s2@0)))
| |  ⇒
| |  paren(T;s1)
| |  ⇒
| |  paren(T;s2)
| |  ⇒
| |  no_repeats(T+T;s1 @ s2)
| |  ⇒
| |  (∀s1@0,s2@0,s3:(T+T) List, x:T.
| |   (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| |   ⇒
| |   paren(T;s2@0))
| | by Auto
| |    THEN
| |    (RWO
| |     Thm* ∀l1,l2:T List.
| |          no_repeats(T;l1 @ l2)
| |          ⟺
| |          l_disjoint(T;l1;l2) & no_repeats(T;l1) & no_repeats(T;l2)
| |     {no_repeats_append_iff}
| |     8)
| |    THEN
| |    (EqualAppendsD -1)
| |    THEN
| |    (Analyze -1)
```

```
| |
| |\
| | 1. T : Type
| | 2. s1 : (T+T) List
| | 3. s2 : (T+T) List
| | 4. no_repeats(T+T;s1)
| | |  ⇒
| | |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3) ⇒ paren(T;s2))
| | 5. no_repeats(T+T;s2)
| | |  ⇒
| | |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3) ⇒ paren(T;s2@0))
| | 6. paren(T;s1)
| | 7. paren(T;s2)
| | 8. l_disjoint(T+T;s1;s2) & no_repeats(T+T;s1) & no_repeats(T+T;s2)
| | 9. s1@0 : (T+T) List
| | 10. s2@0 : (T+T) List
| | 11. s3 : (T+T) List
| | 12. x : T
| | 13. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | 14. e : (T+T) List
| | 15. s1 = (s1@0 @ e)
| | 16. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | ⊢  paren(T;s2@0)  by (EqualAppendsD -1) THEN (Analyze -1)
| | |
| | |\
| | | 1. T : Type
| | | 2. s1 : (T+T) List
| | | 3. s2 : (T+T) List
| | | 4. no_repeats(T+T;s1)
| | | |  ⇒
| | | |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | | |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3) ⇒ paren(T;s2))
| | | 5. no_repeats(T+T;s2)
| | | |  ⇒
| | | |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | | |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | |   ⇒
| | | |   paren(T;s2@0))
| | | 6. paren(T;s1)
| | | 7. paren(T;s2)
| | | 8. l_disjoint(T+T;s1;s2)
| | | |  & no_repeats(T+T;s1)
| | | |  & no_repeats(T+T;s2)
| | | 9. s1@0 : (T+T) List
| | | 10. s2@0 : (T+T) List
| | | 11. s3 : (T+T) List
| | | 12. x : T
| | | 13. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | 14. e : (T+T) List
| | | 15. s1 = (s1@0 @ e)
| | | 16. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | | 17. e1 : (T+T) List
| | | 18. [inl(x)] = (e @ e1)
```

```
| | | 19. s2 = (e1 @ s2@0 @ [inr(x)] @ s3)
| | | ⊢  paren(T;s2@0)  by SingletonAppendD -2
| | | |
| | | |\
| | | | 1. T : Type
| | | | 2. s1 : (T+T) List
| | | | 3. s2 : (T+T) List
| | | | 4. no_repeats(T+T;s1)
| | | |    ⇒
| | | |    (∀s1@0,s2,s3:(T+T) List, x:T.
| | | |     s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | | |     ⇒
| | | |     paren(T;s2))
| | | | 5. no_repeats(T+T;s2)
| | | |    ⇒
| | | |    (∀s1,s2@0,s3:(T+T) List, x:T.
| | | |     s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | |     ⇒
| | | |     paren(T;s2@0))
| | | | 6. paren(T;s1)
| | | | 7. paren(T;s2)
| | | | 8. l_disjoint(T+T;s1;s2)
| | | |    & no_repeats(T+T;s1)
| | | |    & no_repeats(T+T;s2)
| | | | 9. s1@0 : (T+T) List
| | | | 10. s2@0 : (T+T) List
| | | | 11. s3 : (T+T) List
| | | | 12. x : T
| | | | 13. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | | 14. e : (T+T) List
| | | | 15. s1 = (s1@0 @ e)
| | | | 16. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | | | 17. e1 : (T+T) List
| | | | 18. [inl(x)] = (e @ e1)
| | | | 19. s2 = (e1 @ s2@0 @ [inr(x)] @ s3)
| | | | 20. e = nil
| | | | 21. e1 = [inl(x)]
| | | | ⊢  paren(T;s2@0)
| | | |   by (InstHyp [nil;s2@0;s3;x] 5) THEN (SubstFor s2 0)
| | | |      THEN
| | | |      (SubstFor e1 0)
| | | |      THEN
| | | |      (Reduce 0)
| | | |
| | | .\
| | | . 1. T : Type
| | | . 2. s1 : (T+T) List
| | | . 3. s2 : (T+T) List
| | | . 4. no_repeats(T+T;s1)
| | | . |  ⇒
| | | . |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | | . |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | | . |   ⇒
| | | . |   paren(T;s2))
| | | . 5. no_repeats(T+T;s2)
```

```
| | | . |  ⇒
| | | . |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | | . |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | . |   ⇒
| | | . |   paren(T;s2@0))
| | | . 6. paren(T;s1)
| | | . 7. paren(T;s2)
| | | . 8. l_disjoint(T+T;s1;s2)
| | | . |  & no_repeats(T+T;s1)
| | | . |  & no_repeats(T+T;s2)
| | | . 9. s1@0 : (T+T) List
| | | . 10. s2@0 : (T+T) List
| | | . 11. s3 : (T+T) List
| | | . 12. x : T
| | | . 13. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | . 14. e : (T+T) List
| | | . 15. s1 = (s1@0 @ e)
| | | . 16. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | | . 17. e1 : (T+T) List
| | | . 18. [inl(x)] = (e @ e1)
| | | . 19. s2 = (e1 @ s2@0 @ [inr(x)] @ s3)
| | | . 20. e1 = nil
| | | . 21. e = [inl(x)]
| | | . ⊢  paren(T;s2@0)
| | | . | by Auto
| | | . |    THEN
| | | . |    (AllHyps
| | | . |     (λh.
| | | . |      (Unfold `l_disjoint` h) THEN (InstHyp [inl(x)] h)
| | | . |      THEN
| | | . |      (Analyze -1)))
| | | . |
| | | . 1. T : Type
| | | . 2. s1 : (T+T) List
| | | . 3. s2 : (T+T) List
| | | . 4. no_repeats(T+T;s1)
| | | . |  ⇒
| | | . |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | | . |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | | . |   ⇒
| | | . |   paren(T;s2))
| | | . 5. no_repeats(T+T;s2)
| | | . |  ⇒
| | | . |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | | . |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | . |   ⇒
| | | . |   paren(T;s2@0))
| | | . 6. paren(T;s1)
| | | . 7. paren(T;s2)
| | | . 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | | . 9. no_repeats(T+T;s1)
| | | . 10. no_repeats(T+T;s2)
| | | . 11. s1@0 : (T+T) List
| | | . 12. s2@0 : (T+T) List
| | | . 13. s3 : (T+T) List
```

```
| | | . 14. x : T
| | | . 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | . 16. e : (T+T) List
| | | . 17. s1 = (s1@0 @ e)
| | | . 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | | . 19. e1 : (T+T) List
| | | . 20. [inl(x)] = (e @ e1)
| | | . 21. s2 = (e1 @ s2@0 @ [inr(x)] @ s3)
| | | . 22. e1 = nil
| | | . 23. e = [inl(x)]
| | | . ⊢  (inl(x) ∈ s1) & (inl(x) ∈ s2)  by Analyze 0
| | | . |
| | | . |\
| | | . | 1. T : Type
| | | . | 2. s1 : (T+T) List
| | | . | 3. s2 : (T+T) List
| | | . | 4. no_repeats(T+T;s1)
| | | . | |  ⇒
| | | . | |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | | . | |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | | . | |   ⇒
| | | . | |   paren(T;s2))
| | | . | 5. no_repeats(T+T;s2)
| | | . | |  ⇒
| | | . | |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | | . | |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | . | |   ⇒
| | | . | |   paren(T;s2@0))
| | | . | 6. paren(T;s1)
| | | . | 7. paren(T;s2)
| | | . | 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | | . | 9. no_repeats(T+T;s1)
| | | . | 10. no_repeats(T+T;s2)
| | | . | 11. s1@0 : (T+T) List
| | | . | 12. s2@0 : (T+T) List
| | | . | 13. s3 : (T+T) List
| | | . | 14. x : T
| | | . | 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | . | 16. e : (T+T) List
| | | . | 17. s1 = (s1@0 @ e)
| | | . | 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | | . | 19. e1 : (T+T) List
| | | . | 20. [inl(x)] = (e @ e1)
| | | . | 21. s2 = (e1 @ s2@0 @ [inr(x)] @ s3)
| | | . | 22. e1 = nil
| | | . | 23. e = [inl(x)]
| | | . | ⊢  (inl(x) ∈ s1)  by (SubstFor s1 0) THEN (SubstFor e 0)
| | | . | |
| | | . | 1. T : Type
| | | . | 2. s1 : (T+T) List
| | | . | 3. s2 : (T+T) List
| | | . | 4. no_repeats(T+T;s1)
| | | . |    ⇒
| | | . |    (∀s1@0,s2,s3:(T+T) List, x:T.
| | | . |     s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
```

```
| | | . |      ⇒
| | | . |      paren(T;s2))
| | | . | 5. no_repeats(T+T;s2)
| | | . |     ⇒
| | | . |     (∀s1,s2@0,s3:(T+T) List, x:T.
| | | . |      s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | . |      ⇒
| | | . |      paren(T;s2@0))
| | | . | 6. paren(T;s1)
| | | . | 7. paren(T;s2)
| | | . | 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | | . | 9. no_repeats(T+T;s1)
| | | . | 10. no_repeats(T+T;s2)
| | | . | 11. s1@0 : (T+T) List
| | | . | 12. s2@0 : (T+T) List
| | | . | 13. s3 : (T+T) List
| | | . | 14. x : T
| | | . | 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | . | 16. e : (T+T) List
| | | . | 17. s1 = (s1@0 @ e)
| | | . | 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | | . | 19. e1 : (T+T) List
| | | . | 20. [inl(x)] = (e @ e1)
| | | . | 21. s2 = (e1 @ s2@0 @ [inr(x)] @ s3)
| | | . | 22. e1 = nil
| | | . | 23. e = [inl(x)]
| | | . | ⊢  (inl(x) ∈ s1@0 @ [inl(x)])  by ObviousConcl

| | | . |
| | | . .\
| | | . . 1. T : Type
| | | . . 2. s1 : (T+T) List
| | | . . 3. s2 : (T+T) List
| | | . . 4. no_repeats(T+T;s1)
| | | . . |  ⇒
| | | . . |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | | . . |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | | . . |   ⇒
| | | . . |   paren(T;s2))
| | | . . 5. no_repeats(T+T;s2)
| | | . . |  ⇒
| | | . . |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | | . . |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | . . |   ⇒
| | | . . |   paren(T;s2@0))
| | | . . 6. paren(T;s1)
| | | . . 7. paren(T;s2)
| | | . . 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | | . . 9. no_repeats(T+T;s1)
| | | . . 10. no_repeats(T+T;s2)
| | | . . 11. s1@0 : (T+T) List
| | | . . 12. s2@0 : (T+T) List
| | | . . 13. s3 : (T+T) List
| | | . . 14. x : T
| | | . . 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | . . 16. e : (T+T) List
```

```
| | | . . 17. s1 = (s1@0 @ e)
| | | . . 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | | . . 19. e1 : (T+T) List
| | | . . 20. [inl(x)] = (e @ e1)
| | | . . 21. s2 = (e1 @ s2@0 @ [inr(x)] @ s3)
| | | . . 22. e1 = nil
| | | . . 23. e = [inl(x)]
| | | . . ⊢  (inl(x) ∈ s2)
| | | . . | by (BackThru
| | | . . |     Thm* ∀s':(T+T) List.
| | | . . |          paren(T;s')
| | | . . |          ⇒
| | | . . |          (∀i:T. (inr(i) ∈ s') ⇒ (inl(i) ∈ s'))
| | | . . |     {paren_balance2})
| | | . . |    THEN
| | | . . |    (SubstFor s2 0)
| | | . . |
| | | . . 1. T : Type
| | | . . 2. s1 : (T+T) List
| | | . . 3. s2 : (T+T) List
| | | . . 4. no_repeats(T+T;s1)
| | | . .    ⇒
| | | . .    (∀s1@0,s2,s3:(T+T) List, x:T.
| | | . .     s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | | . .     ⇒
| | | . .     paren(T;s2))
| | | . . 5. no_repeats(T+T;s2)
| | | . .    ⇒
| | | . .    (∀s1,s2@0,s3:(T+T) List, x:T.
| | | . .     s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | . .     ⇒
| | | . .     paren(T;s2@0))
| | | . . 6. paren(T;s1)
| | | . . 7. paren(T;s2)
| | | . . 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | | . . 9. no_repeats(T+T;s1)
| | | . . 10. no_repeats(T+T;s2)
| | | . . 11. s1@0 : (T+T) List
| | | . . 12. s2@0 : (T+T) List
| | | . . 13. s3 : (T+T) List
| | | . . 14. x : T
| | | . . 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | | . . 16. e : (T+T) List
| | | . . 17. s1 = (s1@0 @ e)
| | | . . 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | | . . 19. e1 : (T+T) List
| | | . . 20. [inl(x)] = (e @ e1)
| | | . . 21. s2 = (e1 @ s2@0 @ [inr(x)] @ s3)
| | | . . 22. e1 = nil
| | | . . 23. e = [inl(x)]
| | | . . ⊢  (inr(x) ∈ e1 @ s2@0 @ [inr(x)] @ s3)  by ObviousConcl
| | |
| | .\
| | . 1. T : Type
| | . 2. s1 : (T+T) List
```

```
| | . 3. s2 : (T+T) List
| | . 4. no_repeats(T+T;s1)
| | . |  ⇒
| | . |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | . |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3) ⇒ paren(T;s2))
| | . 5. no_repeats(T+T;s2)
| | . |  ⇒
| | . |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | . |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . |   ⇒
| | . |   paren(T;s2@0))
| | . 6. paren(T;s1)
| | . 7. paren(T;s2)
| | . 8. l_disjoint(T+T;s1;s2)
| | . |  & no_repeats(T+T;s1)
| | . |  & no_repeats(T+T;s2)
| | . 9. s1@0 : (T+T) List
| | . 10. s2@0 : (T+T) List
| | . 11. s3 : (T+T) List
| | . 12. x : T
| | . 13. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . 14. e : (T+T) List
| | . 15. s1 = (s1@0 @ e)
| | . 16. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . 17. e1 : (T+T) List
| | . 18. e = ([inl(x)] @ e1)
| | . 19. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . ⊢  paren(T;s2@0)  by (EqualAppendsD -1) THEN (Analyze -1)
| | . |
| | . |\
| | . | 1. T : Type
| | . | 2. s1 : (T+T) List
| | . | 3. s2 : (T+T) List
| | . | 4. no_repeats(T+T;s1)
| | . | |  ⇒
| | . | |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | . | |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . | |   ⇒
| | . | |   paren(T;s2))
| | . | 5. no_repeats(T+T;s2)
| | . | |  ⇒
| | . | |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | . | |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . | |   ⇒
| | . | |   paren(T;s2@0))
| | . | 6. paren(T;s1)
| | . | 7. paren(T;s2)
| | . | 8. l_disjoint(T+T;s1;s2)
| | . | |  & no_repeats(T+T;s1)
| | . | |  & no_repeats(T+T;s2)
| | . | 9. s1@0 : (T+T) List
| | . | 10. s2@0 : (T+T) List
| | . | 11. s3 : (T+T) List
| | . | 12. x : T
| | . | 13. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
```

```
| | . | 14. e : (T+T) List
| | . | 15. s1 = (s1@0 @ e)
| | . | 16. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . | 17. e1 : (T+T) List
| | . | 18. e = ([inl(x)] @ e1)
| | . | 19. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . | 20. e2 : (T+T) List
| | . | 21. s2@0 = (e1 @ e2)
| | . | 22. s2 = (e2 @ [inr(x)] @ s3)
| | . | ⊢  paren(T;s2@0)
| | . | | by Auto
| | . | |    THEN
| | . | |    (AllHyps
| | . | |     (λh.
| | . | |      (Unfold `l_disjoint` h) THEN (InstHyp [inl(x)] h)
| | . | |      THEN
| | . | |      (Analyze -1)))
| | . | |
| | . | 1. T : Type
| | . | 2. s1 : (T+T) List
| | . | 3. s2 : (T+T) List
| | . | 4. no_repeats(T+T;s1)
| | . | |  ⇒
| | . | |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | . | |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . | |   ⇒
| | . | |   paren(T;s2))
| | . | 5. no_repeats(T+T;s2)
| | . | |  ⇒
| | . | |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | . | |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . | |   ⇒
| | . | |   paren(T;s2@0))
| | . | 6. paren(T;s1)
| | . | 7. paren(T;s2)
| | . | 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | . | 9. no_repeats(T+T;s1)
| | . | 10. no_repeats(T+T;s2)
| | . | 11. s1@0 : (T+T) List
| | . | 12. s2@0 : (T+T) List
| | . | 13. s3 : (T+T) List
| | . | 14. x : T
| | . | 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . | 16. e : (T+T) List
| | . | 17. s1 = (s1@0 @ e)
| | . | 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . | 19. e1 : (T+T) List
| | . | 20. e = ([inl(x)] @ e1)
| | . | 21. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . | 22. e2 : (T+T) List
| | . | 23. s2@0 = (e1 @ e2)
| | . | 24. s2 = (e2 @ [inr(x)] @ s3)
| | . | ⊢  (inl(x) ∈ s1) & (inl(x) ∈ s2)  by Analyze 0
| | . | |
| | . | |\
```

```
| | . | | 1. T : Type
| | . | | 2. s1 : (T+T) List
| | . | | 3. s2 : (T+T) List
| | . | | 4. no_repeats(T+T;s1)
| | . | | |  ⇒
| | . | | |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | . | | |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . | | |   ⇒
| | . | | |   paren(T;s2))
| | . | | 5. no_repeats(T+T;s2)
| | . | | |  ⇒
| | . | | |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | . | | |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . | | |   ⇒
| | . | | |   paren(T;s2@0))
| | . | | 6. paren(T;s1)
| | . | | 7. paren(T;s2)
| | . | | 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | . | | 9. no_repeats(T+T;s1)
| | . | | 10. no_repeats(T+T;s2)
| | . | | 11. s1@0 : (T+T) List
| | . | | 12. s2@0 : (T+T) List
| | . | | 13. s3 : (T+T) List
| | . | | 14. x : T
| | . | | 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . | | 16. e : (T+T) List
| | . | | 17. s1 = (s1@0 @ e)
| | . | | 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . | | 19. e1 : (T+T) List
| | . | | 20. e = ([inl(x)] @ e1)
| | . | | 21. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . | | 22. e2 : (T+T) List
| | . | | 23. s2@0 = (e1 @ e2)
| | . | | 24. s2 = (e2 @ [inr(x)] @ s3)
| | . | | ⊢  (inl(x) ∈ s1)  by (SubstFor s1 0) THEN (SubstFor e 0)
| | . | | |
| | . | | 1. T : Type
| | . | | 2. s1 : (T+T) List
| | . | | 3. s2 : (T+T) List
| | . | | 4. no_repeats(T+T;s1)
| | . | | |  ⇒
| | . | | |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | . | | |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . | | |   ⇒
| | . | | |   paren(T;s2))
| | . | | 5. no_repeats(T+T;s2)
| | . | | |  ⇒
| | . | | |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | . | | |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . | | |   ⇒
| | . | | |   paren(T;s2@0))
| | . | | 6. paren(T;s1)
| | . | | 7. paren(T;s2)
| | . | | 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | . | | 9. no_repeats(T+T;s1)
```

```
| | . | | 10. no_repeats(T+T;s2)
| | . | | 11. s1@0 : (T+T) List
| | . | | 12. s2@0 : (T+T) List
| | . | | 13. s3 : (T+T) List
| | . | | 14. x : T
| | . | | 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . | | 16. e : (T+T) List
| | . | | 17. s1 = (s1@0 @ e)
| | . | | 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . | | 19. e1 : (T+T) List
| | . | | 20. e = ([inl(x)] @ e1)
| | . | | 21. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . | | 22. e2 : (T+T) List
| | . | | 23. s2@0 = (e1 @ e2)
| | . | | 24. s2 = (e2 @ [inr(x)] @ s3)
| | . | | ⊢  (inl(x) ∈ s1@0 @ [inl(x)] @ e1)  by Lemmaize []
| | . | | |
| | . | | 1. T : Type
| | . | | 2. s1@0 : (T+T) List
| | . | | 3. x : T
| | . | | 4. e1 : (T+T) List
| | . | | ⊢  (inl(x) ∈ s1@0 @ [inl(x)] @ e1)  by Obvious
| | . | |
| | . | .\
| | . | . 1. T : Type
| | . | . 2. s1 : (T+T) List
| | . | . 3. s2 : (T+T) List
| | . | . 4. no_repeats(T+T;s1)
| | . | . |  ⇒
| | . | . |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | . | . |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . | . |   ⇒
| | . | . |   paren(T;s2))
| | . | . 5. no_repeats(T+T;s2)
| | . | . |  ⇒
| | . | . |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | . | . |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . | . |   ⇒
| | . | . |   paren(T;s2@0))
| | . | . 6. paren(T;s1)
| | . | . 7. paren(T;s2)
| | . | . 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | . | . 9. no_repeats(T+T;s1)
| | . | . 10. no_repeats(T+T;s2)
| | . | . 11. s1@0 : (T+T) List
| | . | . 12. s2@0 : (T+T) List
| | . | . 13. s3 : (T+T) List
| | . | . 14. x : T
| | . | . 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . | . 16. e : (T+T) List
| | . | . 17. s1 = (s1@0 @ e)
| | . | . 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)

| | . | . 19. e1 : (T+T) List
| | . | . 20. e = ([inl(x)] @ e1)
| | . | . 21. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
```

```
| | . | . 22. e2 : (T+T) List
| | . | . 23. s2@0 = (e1 @ e2)
| | . | . 24. s2 = (e2 @ [inr(x)] @ s3)
| | . | . ⊢  (inl(x) ∈ s2)
| | . | . | by (BackThru
| | . | . |     Thm* ∀s':(T+T) List.
| | . | . |          paren(T;s')
| | . | . |          ⇒
| | . | . |          (∀i:T. (inr(i) ∈ s') ⇒ (inl(i) ∈ s'))
| | . | . |     {paren_balance2})
| | . | . |    THEN
| | . | . |    (SubstFor s2 0)
| | . | . |
| | . | . 1. T : Type
| | . | . 2. s1 : (T+T) List
| | . | . 3. s2 : (T+T) List
| | . | . 4. no_repeats(T+T;s1)
| | . | .    ⇒
| | . | .    (∀s1@0,s2,s3:(T+T) List, x:T.
| | . | .     s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . | .     ⇒
| | . | .     paren(T;s2))
| | . | . 5. no_repeats(T+T;s2)
| | . | .    ⇒
| | . | .    (∀s1,s2@0,s3:(T+T) List, x:T.
| | . | .     s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . | .     ⇒
| | . | .     paren(T;s2@0))
| | . | . 6. paren(T;s1)
| | . | . 7. paren(T;s2)
| | . | . 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | . | . 9. no_repeats(T+T;s1)
| | . | . 10. no_repeats(T+T;s2)
| | . | . 11. s1@0 : (T+T) List
| | . | . 12. s2@0 : (T+T) List
| | . | . 13. s3 : (T+T) List
| | . | . 14. x : T
| | . | . 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . | . 16. e : (T+T) List
| | . | . 17. s1 = (s1@0 @ e)
| | . | . 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . | . 19. e1 : (T+T) List
| | . | . 20. e = ([inl(x)] @ e1)
| | . | . 21. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . | . 22. e2 : (T+T) List
| | . | . 23. s2@0 = (e1 @ e2)
| | . | . 24. s2 = (e2 @ [inr(x)] @ s3)
| | . | . ⊢  (inr(x) ∈ e2 @ [inr(x)] @ s3)  by ObviousConcl
| | . |
| | . .\
| | . . 1. T : Type
| | . . 2. s1 : (T+T) List
| | . . 3. s2 : (T+T) List
| | . . 4. no_repeats(T+T;s1)
| | . . |  ⇒
```

```
| | . . |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | . . |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . . |   ⇒
| | . . |   paren(T;s2))
| | . . 5. no_repeats(T+T;s2)
| | . . |  ⇒
| | . . |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | . . |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . |   ⇒
| | . . |   paren(T;s2@0))
| | . . 6. paren(T;s1)
| | . . 7. paren(T;s2)
| | . . 8. l_disjoint(T+T;s1;s2)
| | . . |  & no_repeats(T+T;s1)
| | . . |  & no_repeats(T+T;s2)
| | . . 9. s1@0 : (T+T) List
| | . . 10. s2@0 : (T+T) List
| | . . 11. s3 : (T+T) List
| | . . 12. x : T
| | . . 13. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . 14. e : (T+T) List
| | . . 15. s1 = (s1@0 @ e)
| | . . 16. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . . 17. e1 : (T+T) List
| | . . 18. e = ([inl(x)] @ e1)
| | . . 19. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . . 20. e2 : (T+T) List
| | . . 21. e1 = (s2@0 @ e2)
| | . . 22. ([inr(x)] @ s3) = (e2 @ s2)
| | . . ⊢  paren(T;s2@0)  by (EqualAppendsD -1) THEN (Analyze -1)
| | . . |
| | . . |\
| | . . | 1. T : Type
| | . . | 2. s1 : (T+T) List
| | . . | 3. s2 : (T+T) List
| | . . | 4. no_repeats(T+T;s1)
| | . . | |  ⇒
| | . . | |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | . . | |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . . | |   ⇒
| | . . | |   paren(T;s2))
| | . . | 5. no_repeats(T+T;s2)
| | . . | |  ⇒
| | . . | |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | . . | |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | |   ⇒
| | . . | |   paren(T;s2@0))
| | . . | 6. paren(T;s1)
| | . . | 7. paren(T;s2)
| | . . | 8. l_disjoint(T+T;s1;s2)
| | . . | |  & no_repeats(T+T;s1)
| | . . | |  & no_repeats(T+T;s2)
| | . . | 9. s1@0 : (T+T) List
| | . . | 10. s2@0 : (T+T) List
| | . . | 11. s3 : (T+T) List
```

```
| | . . | 12. x : T
| | . . | 13. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | 14. e : (T+T) List
| | . . | 15. s1 = (s1@0 @ e)
| | . . | 16. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . . | 17. e1 : (T+T) List
| | . . | 18. e = ([inl(x)] @ e1)
| | . . | 19. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . . | 20. e2 : (T+T) List
| | . . | 21. e1 = (s2@0 @ e2)
| | . . | 22. ([inr(x)] @ s3) = (e2 @ s2)
| | . . | 23. e3 : (T+T) List
| | . . | 24. [inr(x)] = (e2 @ e3)
| | . . | 25. s2 = (e3 @ s3)
| | . . | ⊢  paren(T;s2@0)  by SingletonAppendD -2
| | . . | |
| | . . | |\
| | . . | | 1. T : Type
| | . . | | 2. s1 : (T+T) List
| | . . | | 3. s2 : (T+T) List
| | . . | | 4. no_repeats(T+T;s1)
| | . . | | |  ⇒
| | . . | | |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | . . | | |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . . | | |   ⇒
| | . . | | |   paren(T;s2))
| | . . | | 5. no_repeats(T+T;s2)
| | . . | | |  ⇒
| | . . | | |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | . . | | |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | | |   ⇒
| | . . | | |   paren(T;s2@0))
| | . . | | 6. paren(T;s1)
| | . . | | 7. paren(T;s2)
| | . . | | 8. l_disjoint(T+T;s1;s2)
| | . . | | |  & no_repeats(T+T;s1)
| | . . | | |  & no_repeats(T+T;s2)
| | . . | | 9. s1@0 : (T+T) List
| | . . | | 10. s2@0 : (T+T) List
| | . . | | 11. s3 : (T+T) List
| | . . | | 12. x : T
| | . . | | 13. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | | 14. e : (T+T) List
| | . . | | 15. s1 = (s1@0 @ e)
| | . . | | 16. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . . | | 17. e1 : (T+T) List
| | . . | | 18. e = ([inl(x)] @ e1)
| | . . | | 19. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . . | | 20. e2 : (T+T) List
| | . . | | 21. e1 = (s2@0 @ e2)
| | . . | | 22. ([inr(x)] @ s3) = (e2 @ s2)
| | . . | | 23. e3 : (T+T) List
| | . . | | 24. [inr(x)] = (e2 @ e3)
| | . . | | 25. s2 = (e3 @ s3)
| | . . | | 26. e2 = nil
```

```
| | . . | | 27. e3 = [inr(x)]
| | . . | | ⊢  paren(T;s2@0)
| | . . | | | by Auto
| | . . | | |    THEN
| | . . | | |    (AllHyps
| | . . | | |     (λh.
| | . . | | |      (Unfold `l_disjoint` h) THEN (InstHyp [inl(x)] h)
| | . . | | |      THEN
| | . . | | |      (Analyze -1)))
| | . . | | |
| | . . | | 1. T : Type
| | . . | | 2. s1 : (T+T) List
| | . . | | 3. s2 : (T+T) List
| | . . | | 4. no_repeats(T+T;s1)
| | . . | | |  ⇒
| | . . | | |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | . . | | |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . . | | |   ⇒
| | . . | | |   paren(T;s2))
| | . . | | 5. no_repeats(T+T;s2)
| | . . | | |  ⇒
| | . . | | |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | . . | | |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | | |   ⇒
| | . . | | |   paren(T;s2@0))
| | . . | | 6. paren(T;s1)
| | . . | | 7. paren(T;s2)
| | . . | | 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | . . | | 9. no_repeats(T+T;s1)
| | . . | | 10. no_repeats(T+T;s2)
| | . . | | 11. s1@0 : (T+T) List
| | . . | | 12. s2@0 : (T+T) List
| | . . | | 13. s3 : (T+T) List
| | . . | | 14. x : T
| | . . | | 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | | 16. e : (T+T) List
| | . . | | 17. s1 = (s1@0 @ e)
| | . . | | 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . . | | 19. e1 : (T+T) List
| | . . | | 20. e = ([inl(x)] @ e1)
| | . . | | 21. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . . | | 22. e2 : (T+T) List
| | . . | | 23. e1 = (s2@0 @ e2)
| | . . | | 24. ([inr(x)] @ s3) = (e2 @ s2)
| | . . | | 25. e3 : (T+T) List
| | . . | | 26. [inr(x)] = (e2 @ e3)
| | . . | | 27. s2 = (e3 @ s3)
| | . . | | 28. e2 = nil
| | . . | | 29. e3 = [inr(x)]
| | . . | | ⊢  (inl(x) ∈ s1) & (inl(x) ∈ s2)  by Analyze 0
| | . . | | |
| | . . | | |\
| | . . | | | 1. T : Type
| | . . | | | 2. s1 : (T+T) List
| | . . | | | 3. s2 : (T+T) List
```

```
| | . . | | | | 4. no_repeats(T+T;s1)
| | . . | | | | |  ⇒
| | . . | | | | |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | . . | | | | |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . . | | | | |   ⇒
| | . . | | | | |   paren(T;s2))
| | . . | | | | 5. no_repeats(T+T;s2)
| | . . | | | | |  ⇒
| | . . | | | | |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | . . | | | | |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | | | | |   ⇒
| | . . | | | | |   paren(T;s2@0))
| | . . | | | | 6. paren(T;s1)
| | . . | | | | 7. paren(T;s2)
| | . . | | | | 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | . . | | | | 9. no_repeats(T+T;s1)
| | . . | | | | 10. no_repeats(T+T;s2)
| | . . | | | | 11. s1@0 : (T+T) List
| | . . | | | | 12. s2@0 : (T+T) List
| | . . | | | | 13. s3 : (T+T) List
| | . . | | | | 14. x : T
| | . . | | | | 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | | | | 16. e : (T+T) List
| | . . | | | | 17. s1 = (s1@0 @ e)
| | . . | | | | 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . . | | | | 19. e1 : (T+T) List
| | . . | | | | 20. e = ([inl(x)] @ e1)
| | . . | | | | 21. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . . | | | | 22. e2 : (T+T) List
| | . . | | | | 23. e1 = (s2@0 @ e2)
| | . . | | | | 24. ([inr(x)] @ s3) = (e2 @ s2)
| | . . | | | | 25. e3 : (T+T) List
| | . . | | | | 26. [inr(x)] = (e2 @ e3)
| | . . | | | | 27. s2 = (e3 @ s3)
| | . . | | | | 28. e2 = nil
| | . . | | | | 29. e3 = [inr(x)]
| | . . | | | | ⊢  (inl(x) ∈ s1)
| | . . | | | | | by (SubstFor s1 0) THEN (SubstFor e 0)
| | . . | | | | |
| | . . | | | | 1. T : Type
| | . . | | | | 2. s1 : (T+T) List
| | . . | | | | 3. s2 : (T+T) List
| | . . | | | | 4. no_repeats(T+T;s1)
| | . . | | | |    ⇒
| | . . | | | |    (∀s1@0,s2,s3:(T+T) List, x:T.
| | . . | | | |     s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . . | | | |     ⇒
| | . . | | | |     paren(T;s2))
| | . . | | | | 5. no_repeats(T+T;s2)
| | . . | | | |    ⇒
| | . . | | | |    (∀s1,s2@0,s3:(T+T) List, x:T.
| | . . | | | |     s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | | | |     ⇒
| | . . | | | |     paren(T;s2@0))
| | . . | | | | 6. paren(T;s1)
```

```
| | . . | | | 7. paren(T;s2)
| | . . | | | 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | . . | | | 9. no_repeats(T+T;s1)
| | . . | | | 10. no_repeats(T+T;s2)
| | . . | | | 11. s1@0 : (T+T) List
| | . . | | | 12. s2@0 : (T+T) List
| | . . | | | 13. s3 : (T+T) List
| | . . | | | 14. x : T
| | . . | | | 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | | | 16. e : (T+T) List
| | . . | | | 17. s1 = (s1@0 @ e)
| | . . | | | 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . . | | | 19. e1 : (T+T) List
| | . . | | | 20. e = ([inl(x)] @ e1)
| | . . | | | 21. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . . | | | 22. e2 : (T+T) List
| | . . | | | 23. e1 = (s2@0 @ e2)
| | . . | | | 24. ([inr(x)] @ s3) = (e2 @ s2)
| | . . | | | 25. e3 : (T+T) List
| | . . | | | 26. [inr(x)] = (e2 @ e3)
| | . . | | | 27. s2 = (e3 @ s3)
| | . . | | | 28. e2 = nil
| | . . | | | 29. e3 = [inr(x)]
| | . . | | | ⊢  (inl(x) ∈ s1@0 @ [inl(x)] @ e1)  by ObviousConcl
| | . . | | |
| | . . | | .\
| | . . | | . 1. T : Type
| | . . | | . 2. s1 : (T+T) List
| | . . | | . 3. s2 : (T+T) List
| | . . | | . 4. no_repeats(T+T;s1)
| | . . | | . |  ⇒
| | . . | | . |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | . . | | . |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . . | | . |   ⇒
| | . . | | . |   paren(T;s2))
| | . . | | . 5. no_repeats(T+T;s2)
| | . . | | . |  ⇒
| | . . | | . |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | . . | | . |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | | . |   ⇒
| | . . | | . |   paren(T;s2@0))
| | . . | | . 6. paren(T;s1)
| | . . | | . 7. paren(T;s2)
| | . . | | . 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | . . | | . 9. no_repeats(T+T;s1)
| | . . | | . 10. no_repeats(T+T;s2)
| | . . | | . 11. s1@0 : (T+T) List
| | . . | | . 12. s2@0 : (T+T) List
| | . . | | . 13. s3 : (T+T) List
| | . . | | . 14. x : T
| | . . | | . 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | | . 16. e : (T+T) List
| | . . | | . 17. s1 = (s1@0 @ e)
| | . . | | . 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . . | | . 19. e1 : (T+T) List
```

```
| | . . | | . 20. e = ([inl(x)] @ e1)
| | . . | | . 21. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . . | | . 22. e2 : (T+T) List
| | . . | | . 23. e1 = (s2@0 @ e2)
| | . . | | . 24. ([inr(x)] @ s3) = (e2 @ s2)
| | . . | | . 25. e3 : (T+T) List
| | . . | | . 26. [inr(x)] = (e2 @ e3)
| | . . | | . 27. s2 = (e3 @ s3)
| | . . | | . 28. e2 = nil
| | . . | | . 29. e3 = [inr(x)]
| | . . | | . ⊢  (inl(x) ∈ s2)
| | . . | | . | by (BackThru
| | . . | | . |     Thm* ∀s':(T+T) List.
| | . . | | . |          paren(T;s')
| | . . | | . |          ⇒
| | . . | | . |          (∀i:T. (inr(i) ∈ s') ⇒ (inl(i) ∈ s'))
| | . . | | . |     {paren_balance2})
| | . . | | . |    THEN
| | . . | | . |    (SubstFor s2 0)
| | . . | | . |    THEN
| | . . | | . |    (SubstFor e3 0)
| | . . | | . |
| | . . | | . 1. T : Type
| | . . | | . 2. s1 : (T+T) List
| | . . | | . 3. s2 : (T+T) List
| | . . | | . 4. no_repeats(T+T;s1)
| | . . | | .    ⇒
| | . . | | .    (∀s1@0,s2,s3:(T+T) List, x:T.
| | . . | | .     s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . . | | .     ⇒
| | . . | | .     paren(T;s2))
| | . . | | . 5. no_repeats(T+T;s2)
| | . . | | .    ⇒
| | . . | | .    (∀s1,s2@0,s3:(T+T) List, x:T.
| | . . | | .     s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | | .     ⇒
| | . . | | .     paren(T;s2@0))
| | . . | | . 6. paren(T;s1)
| | . . | | . 7. paren(T;s2)
| | . . | | . 8. ∀x:T+T. ¬((x ∈ s1) & (x ∈ s2))
| | . . | | . 9. no_repeats(T+T;s1)
| | . . | | . 10. no_repeats(T+T;s2)
| | . . | | . 11. s1@0 : (T+T) List
| | . . | | . 12. s2@0 : (T+T) List
| | . . | | . 13. s3 : (T+T) List
| | . . | | . 14. x : T
| | . . | | . 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | | . 16. e : (T+T) List
| | . . | | . 17. s1 = (s1@0 @ e)
| | . . | | . 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . . | | . 19. e1 : (T+T) List
| | . . | | . 20. e = ([inl(x)] @ e1)
| | . . | | . 21. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . . | | . 22. e2 : (T+T) List
| | . . | | . 23. e1 = (s2@0 @ e2)
```

```
| | . . | | . 24. ([inr(x)] @ s3) = (e2 @ s2)
| | . . | | . 25. e3 : (T+T) List
| | . . | | . 26. [inr(x)] = (e2 @ e3)
| | . . | | . 27. s2 = (e3 @ s3)
| | . . | | . 28. e2 = nil
| | . . | | . 29. e3 = [inr(x)]
| | . . | | . ⊢  (inr(x) ∈ [inr(x)] @ s3)  by ObviousConcl
| | . . | |
| | . . | .\
| | . . | . 1. T : Type
| | . . | . 2. s1 : (T+T) List
| | . . | . 3. s2 : (T+T) List
| | . . | . 4. no_repeats(T+T;s1)
| | . . | . |  ⇒
| | . . | . |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | . . | . |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . . | . |   ⇒
| | . . | . |   paren(T;s2))
| | . . | . 5. no_repeats(T+T;s2)
| | . . | . |  ⇒
| | . . | . |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | . . | . |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | . |   ⇒
| | . . | . |   paren(T;s2@0))
| | . . | . 6. paren(T;s1)
| | . . | . 7. paren(T;s2)
| | . . | . 8. l_disjoint(T+T;s1;s2)
| | . . | . |  & no_repeats(T+T;s1)
| | . . | . |  & no_repeats(T+T;s2)
| | . . | . 9. s1@0 : (T+T) List
| | . . | . 10. s2@0 : (T+T) List
| | . . | . 11. s3 : (T+T) List
| | . . | . 12. x : T
| | . . | . 13. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | . 14. e : (T+T) List
| | . . | . 15. s1 = (s1@0 @ e)
| | . . | . 16. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . . | . 17. e1 : (T+T) List
| | . . | . 18. e = ([inl(x)] @ e1)
| | . . | . 19. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . . | . 20. e2 : (T+T) List
| | . . | . 21. e1 = (s2@0 @ e2)
| | . . | . 22. ([inr(x)] @ s3) = (e2 @ s2)
| | . . | . 23. e3 : (T+T) List
| | . . | . 24. [inr(x)] = (e2 @ e3)
| | . . | . 25. s2 = (e3 @ s3)
| | . . | . 26. e3 = nil
| | . . | . 27. e2 = [inr(x)]
| | . . | . ⊢  paren(T;s2@0)
| | . . | . | by (InstHyp [s1@0;s2@0;nil;x] 4) THEN (SubstFor s1 0)
| | . . | . |    THEN
| | . . | . |    Analyze
| | . . | . |    THEN
| | . . | . |    (SubstFor e 0)
| | . . | . |    THEN
```

```
| | . . | . |    Analyze
| | . . | . |    THEN
| | . . | . |    (SubstFor e1 0)
| | . . | . |    THEN
| | . . | . |    Analyze
| | . . | . |    THEN
| | . . | . |    (SubstFor e2 0)
| | . . | . |
| | . . | . 1. T : Type
| | . . | . 2. s1 : (T+T) List
| | . . | . 3. s2 : (T+T) List
| | . . | . 4. no_repeats(T+T;s1)
| | . . | .    ⇒
| | . . | .    (∀s1@0,s2,s3:(T+T) List, x:T.
| | . . | .     s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . . | .     ⇒
| | . . | .     paren(T;s2))
| | . . | . 5. no_repeats(T+T;s2)
| | . . | .    ⇒
| | . . | .    (∀s1,s2@0,s3:(T+T) List, x:T.
| | . . | .     s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | .     ⇒
| | . . | .     paren(T;s2@0))
| | . . | . 6. paren(T;s1)
| | . . | . 7. paren(T;s2)
| | . . | . 8. l_disjoint(T+T;s1;s2)
| | . . | . 9. no_repeats(T+T;s1)
| | . . | . 10. no_repeats(T+T;s2)
| | . . | . 11. s1@0 : (T+T) List
| | . . | . 12. s2@0 : (T+T) List
| | . . | . 13. s3 : (T+T) List
| | . . | . 14. x : T
| | . . | . 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . | . 16. e : (T+T) List
| | . . | . 17. s1 = (s1@0 @ e)
| | . . | . 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . . | . 19. e1 : (T+T) List
| | . . | . 20. e = ([inl(x)] @ e1)
| | . . | . 21. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . . | . 22. e2 : (T+T) List
| | . . | . 23. e1 = (s2@0 @ e2)
| | . . | . 24. ([inr(x)] @ s3) = (e2 @ s2)
| | . . | . 25. e3 : (T+T) List
| | . . | . 26. [inr(x)] = (e2 @ e3)
| | . . | . 27. s2 = (e3 @ s3)
| | . . | . 28. e3 = nil
| | . . | . 29. e2 = [inr(x)]
| | . . | . ⊢  [inr(x)] = ([inr(x)] @ nil)  by Obvious
| | . . |
| | . . .\
| | . . . 1. T : Type
| | . . . 2. s1 : (T+T) List
| | . . . 3. s2 : (T+T) List
| | . . . 4. no_repeats(T+T;s1)
| | . . . |  ⇒
```

```
| | . . . |  (∀s1@0,s2,s3:(T+T) List, x:T.
| | . . . |   s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . . . |   ⇒
| | . . . |   paren(T;s2))
| | . . . 5. no_repeats(T+T;s2)
| | . . . |  ⇒
| | . . . |  (∀s1,s2@0,s3:(T+T) List, x:T.
| | . . . |   s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . . |   ⇒
| | . . . |   paren(T;s2@0))
| | . . . 6. paren(T;s1)
| | . . . 7. paren(T;s2)
| | . . . 8. l_disjoint(T+T;s1;s2)
| | . . . |  & no_repeats(T+T;s1)
| | . . . |  & no_repeats(T+T;s2)
| | . . . 9. s1@0 : (T+T) List
| | . . . 10. s2@0 : (T+T) List
| | . . . 11. s3 : (T+T) List
| | . . . 12. x : T
| | . . . 13. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . . 14. e : (T+T) List
| | . . . 15. s1 = (s1@0 @ e)
| | . . . 16. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . . . 17. e1 : (T+T) List
| | . . . 18. e = ([inl(x)] @ e1)
| | . . . 19. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . . . 20. e2 : (T+T) List
| | . . . 21. e1 = (s2@0 @ e2)
| | . . . 22. ([inr(x)] @ s3) = (e2 @ s2)
| | . . . 23. e3 : (T+T) List
| | . . . 24. e2 = ([inr(x)] @ e3)
| | . . . 25. s3 = (e3 @ s2)
| | . . . ⊢  paren(T;s2@0)  by InstHyp [s1@0;s2@0;e3;x] 4
| | . . . |
| | . . . 1. T : Type
| | . . . 2. s1 : (T+T) List
| | . . . 3. s2 : (T+T) List
| | . . . 4. no_repeats(T+T;s1)
| | . . .    ⇒
| | . . .    (∀s1@0,s2,s3:(T+T) List, x:T.
| | . . .     s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
| | . . .     ⇒
| | . . .     paren(T;s2))
| | . . . 5. no_repeats(T+T;s2)
| | . . .    ⇒
| | . . .    (∀s1,s2@0,s3:(T+T) List, x:T.
| | . . .     s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . .     ⇒
| | . . .     paren(T;s2@0))
| | . . . 6. paren(T;s1)
| | . . . 7. paren(T;s2)
| | . . . 8. l_disjoint(T+T;s1;s2)
| | . . . 9. no_repeats(T+T;s1)
| | . . . 10. no_repeats(T+T;s2)
| | . . . 11. s1@0 : (T+T) List
```

```
| | . . . 12. s2@0 : (T+T) List
| | . . . 13. s3 : (T+T) List
| | . . . 14. x : T
| | . . . 15. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| | . . . 16. e : (T+T) List
| | . . . 17. s1 = (s1@0 @ e)
| | . . . 18. ([inl(x)] @ s2@0 @ [inr(x)] @ s3) = (e @ s2)
| | . . . 19. e1 : (T+T) List
| | . . . 20. e = ([inl(x)] @ e1)
| | . . . 21. (s2@0 @ [inr(x)] @ s3) = (e1 @ s2)
| | . . . 22. e2 : (T+T) List
| | . . . 23. e1 = (s2@0 @ e2)
| | . . . 24. ([inr(x)] @ s3) = (e2 @ s2)
| | . . . 25. e3 : (T+T) List
| | . . . 26. e2 = ([inr(x)] @ e3)
| | . . . 27. s3 = (e3 @ s2)
| | . . . ⊢  s1 = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ e3)
| | . . .   by (SubstFor s1 0) THEN Analyze THEN (SubstFor e 0)
| | . . .      THEN
| | . . .      Analyze
| | . . .      THEN
| | . . .      (SubstFor e1 0)
| | . . .      THEN
| | . . .      Analyze
| | . . .      THEN
| | . . .      (SubstFor e2 0)
| | . . .      THEN
| | . . .      Analyze
| |
| .\
| . 1. T : Type
| . 2. s1 : (T+T) List
| . 3. s2 : (T+T) List
| . 4. no_repeats(T+T;s1)
| .    ⇒
| .    (∀s1@0,s2,s3:(T+T) List, x:T.
| .     s1 = (s1@0 @ [inl(x)] @ s2 @ [inr(x)] @ s3) ⇒ paren(T;s2))
| . 5. no_repeats(T+T;s2)
| .    ⇒
| .    (∀s1,s2@0,s3:(T+T) List, x:T.
| .     s2 = (s1 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3) ⇒ paren(T;s2@0))
| . 6. paren(T;s1)
| . 7. paren(T;s2)
| . 8. l_disjoint(T+T;s1;s2) & no_repeats(T+T;s1) & no_repeats(T+T;s2)
| . 9. s1@0 : (T+T) List
| . 10. s2@0 : (T+T) List
| . 11. s3 : (T+T) List
| . 12. x : T
| . 13. (s1 @ s2) = (s1@0 @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| . 14. e : (T+T) List
| . 15. s1@0 = (s1 @ e)
| . 16. s2 = (e @ [inl(x)] @ s2@0 @ [inr(x)] @ s3)
| . ⊢  paren(T;s2@0)  by InstHyp [e;s2@0;s3;x] 5
|
.\
```

```
. 1. T : Type
. ⊢  ∀s:(T+T) List, t:T.
. |  (no_repeats(T+T;s)
. |   ⇒
. |   (∀s1,s2,s3:(T+T) List, x:T.
. |    s = (s1 @ [inl(x)] @ s2 @ [inr(x)] @ s3) ⇒ paren(T;s2)))
. |  ⇒
. |  paren(T;s)
. |  ⇒
. |  no_repeats(T+T;[inl(t)] @ s @ [inr(t)])
. |  ⇒
. |  (∀s1,s2,s3:(T+T) List, x:T.
. |   ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
. |   ⇒
. |   paren(T;s2))
. | by Auto
. |    THEN
. |    (RWO
. |     Thm* ∀l1,l2:T List.
. |          no_repeats(T;l1 @ l2)
. |          ⟺
. |          l_disjoint(T;l1;l2) & no_repeats(T;l1) & no_repeats(T;l2)
. |     {no_repeats_append_iff}
. |     6)
. |    THEN
. |    (RWO
. |     Thm* ∀l1,l2:T List.
. |          no_repeats(T;l1 @ l2)
. |          ⟺
. |          l_disjoint(T;l1;l2) & no_repeats(T;l1) & no_repeats(T;l2)
. |     {no_repeats_append_iff}
. |     8)
. |
. 1. T : Type
. 2. s : (T+T) List
. 3. t : T
. 4. no_repeats(T+T;s)
. |  ⇒
. |  (∀s1,s2,s3:(T+T) List, x:T.
. |   s = (s1 @ [inl(x)] @ s2 @ [inr(x)] @ s3) ⇒ paren(T;s2))
. 5. paren(T;s)
. 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. 7. no_repeats(T+T;[inl(t)])
. 8. l_disjoint(T+T;s;[inr(t)])
. 9. no_repeats(T+T;s)
. 10. no_repeats(T+T;[inr(t)])
. 11. s1 : (T+T) List
. 12. s2 : (T+T) List
. 13. s3 : (T+T) List
. 14. x : T
. 15. ([inl(t)] @ s @ [inr(t)]) = (s1 @ [inl(x)] @ s2 @ [inr(x)] @ s3)
. ⊢  paren(T;s2)
. | by (Analyze -5) THEN (All Reduce) THEN (SplitCons -1)
. |    THEN
. |    (Try ((Reduce 0) THEN (Reduce 0) THEN (Unfold ‘label‘ 0)))
```

```
. |
. |\
. | 1. T : Type
. | 2. s : (T+T) List
. | 3. t : T
. | 4. no_repeats(T+T;s)
. | |  ⇒
. | |  (∀s1,s2,s3:(T+T) List, x:T.
. | |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])]) ⇒ paren(T;s2))
. | 5. paren(T;s)
. | 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. | 7. no_repeats(T+T;[inl(t)])
. | 8. l_disjoint(T+T;s;[inr(t)])
. | 9. no_repeats(T+T;s)
. | 10. no_repeats(T+T;[inr(t)])
. | 11. s2 : (T+T) List
. | 12. s3 : (T+T) List
. | 13. x : T
. | 14. inl(t) = inl(x)
. | 15. (s @ [inr(t)]) = (s2 @ [inr(x) / s3])
. | ⊢  paren(T;s2)  by (EqualAppendsD -1) THEN (Analyze -1)
. | |
. | |\
. | | 1. T : Type
. | | 2. s : (T+T) List
. | | 3. t : T
. | | 4. no_repeats(T+T;s)
. | | |  ⇒
. | | |  (∀s1,s2,s3:(T+T) List, x:T.
. | | |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])]) ⇒ paren(T;s2))
. | | 5. paren(T;s)
. | | 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. | | 7. no_repeats(T+T;[inl(t)])
. | | 8. l_disjoint(T+T;s;[inr(t)])
. | | 9. no_repeats(T+T;s)
. | | 10. no_repeats(T+T;[inr(t)])
. | | 11. s2 : (T+T) List
. | | 12. s3 : (T+T) List
. | | 13. x : T
. | | 14. inl(t) = inl(x)
. | | 15. (s @ [inr(t)]) = (s2 @ [inr(x) / s3])
. | | 16. e : (T+T) List
. | | 17. s = (s2 @ e)
. | | 18. [inr(x) / s3] = (e @ [inr(t)])
. | | ⊢  paren(T;s2)  by (Analyze -3) THEN (All Reduce)
. | | |
. | | |\
. | | | 1. T : Type
. | | | 2. s : (T+T) List
. | | | 3. t : T
. | | | 4. no_repeats(T+T;s)
. | | |    ⇒
. | | |    (∀s1,s2,s3:(T+T) List, x:T.
. | | |     s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])]) ⇒ paren(T;s2))
. | | | 5. paren(T;s)
```

```
. | | | 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. | | | 7. no_repeats(T+T;[inl(t)])
. | | | 8. l_disjoint(T+T;s;[inr(t)])
. | | | 9. no_repeats(T+T;s)
. | | | 10. no_repeats(T+T;[inr(t)])
. | | | 11. s2 : (T+T) List
. | | | 12. s3 : (T+T) List
. | | | 13. x : T
. | | | 14. inl(t) = inl(x)
. | | | 15. (s @ [inr(t)]) = (s2 @ [inr(x) / s3])
. | | | 16. s = (s2 @ nil)
. | | | 17. [inr(x) / s3] = [inr(t)]
. | | | ⊢  paren(T;s2)
. | | |   by (RWO Thm* ∀l:T List. (l @ nil) ~ l    {append_nil_sq} -2)
. | | |      THEN
. | | |      (RevHypSubst -2 0)
. | | |
. | | .\
. | | . 1. T : Type
. | | . 2. s : (T+T) List
. | | . 3. t : T
. | | . 4. no_repeats(T+T;s)
. | | . |  ⇒
. | | . |  (∀s1,s2,s3:(T+T) List, x:T.
. | | . |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])]) ⇒ paren(T;s2))
. | | . 5. paren(T;s)
. | | . 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. | | . 7. no_repeats(T+T;[inl(t)])
. | | . 8. l_disjoint(T+T;s;[inr(t)])
. | | . 9. no_repeats(T+T;s)
. | | . 10. no_repeats(T+T;[inr(t)])
. | | . 11. s2 : (T+T) List
. | | . 12. s3 : (T+T) List
. | | . 13. x : T
. | | . 14. inl(t) = inl(x)
. | | . 15. (s @ [inr(t)]) = (s2 @ [inr(x) / s3])
. | | . 16. u : T+T
. | | . 17. v : (T+T) List
. | | . 18. s = (s2 @ [u / v])
. | | . 19. [inr(x) / s3] = [u / (v @ [inr(t)])]
. | | . ⊢  paren(T;s2)  by SplitCons -1
. | | . |
. | | . 1. T : Type
. | | . 2. s : (T+T) List
. | | . 3. t : T
. | | . 4. no_repeats(T+T;s)
. | | . |  ⇒
. | | . |  (∀s1,s2,s3:(T+T) List, x:T.
. | | . |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])]) ⇒ paren(T;s2))
. | | . 5. paren(T;s)
. | | . 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. | | . 7. no_repeats(T+T;[inl(t)])
. | | . 8. l_disjoint(T+T;s;[inr(t)])
. | | . 9. no_repeats(T+T;s)
. | | . 10. no_repeats(T+T;[inr(t)])
```

```
. | | . 11. s2 : (T+T) List
. | | . 12. s3 : (T+T) List
. | | . 13. x : T
. | | . 14. inl(t) = inl(x)
. | | . 15. (s @ [inr(t)]) = (s2 @ [inr(x) / s3])
. | | . 16. u : T+T
. | | . 17. v : (T+T) List
. | | . 18. s = (s2 @ [u / v])
. | | . 19. inr(x) = u
. | | . 20. s3 = (v @ [inr(t)])
. | | . ⊢  paren(T;s2)
. | | . | by (Unfold `l_disjoint` 8) THEN (InstHyp [inr(x)] 8)
. | | . |    THEN
. | | . |    (Analyze -1)
. | | . |    THEN
. | | . |    (SubstFor s 0)
. | | . |
. | | . 1. T : Type
. | | . 2. s : (T+T) List
. | | . 3. t : T
. | | . 4. no_repeats(T+T;s)
. | | .    ⇒
. | | .    (∀s1,s2,s3:(T+T) List, x:T.
. | | .     s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])]) ⇒ paren(T;s2))
. | | . 5. paren(T;s)
. | | . 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. | | . 7. no_repeats(T+T;[inl(t)])
. | | . 8. ∀x:T+T. ¬((x ∈ s) & (x ∈ [inr(t)]))
. | | . 9. no_repeats(T+T;s)
. | | . 10. no_repeats(T+T;[inr(t)])
. | | . 11. s2 : (T+T) List
. | | . 12. s3 : (T+T) List
. | | . 13. x : T
. | | . 14. inl(t) = inl(x)
. | | . 15. (s @ [inr(t)]) = (s2 @ [inr(x) / s3])
. | | . 16. u : T+T
. | | . 17. v : (T+T) List
. | | . 18. s = (s2 @ [u / v])
. | | . 19. inr(x) = u
. | | . 20. s3 = (v @ [inr(t)])
. | | . ⊢  (inr(x) ∈ s2 @ [u / v]) & (inr(x) ∈ [inr(t)])
. | | .   by ObviousFrom [14;19]
. | |
. | .\
. | . 1. T : Type
. | . 2. s : (T+T) List
. | . 3. t : T
. | . 4. no_repeats(T+T;s)
. | . |  ⇒
. | . |  (∀s1,s2,s3:(T+T) List, x:T.
. | . |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])]) ⇒ paren(T;s2))
. | . 5. paren(T;s)
. | . 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. | . 7. no_repeats(T+T;[inl(t)])
. | . 8. l_disjoint(T+T;s;[inr(t)])
```

```
. | . 9. no_repeats(T+T;s)
. | . 10. no_repeats(T+T;[inr(t)])
. | . 11. s2 : (T+T) List
. | . 12. s3 : (T+T) List
. | . 13. x : T
. | . 14. inl(t) = inl(x)
. | . 15. (s @ [inr(t)]) = (s2 @ [inr(x) / s3])
. | . 16. e : (T+T) List
. | . 17. s2 = (s @ e)
. | . 18. [inr(t)] = (e @ [inr(x) / s3])
. | . ⊢  paren(T;s2)
. | . | by (Analyze -3) THEN (All Reduce) THEN (SplitCons -1)
. | . |
. | . |\
. | . | 1. T : Type
. | . | 2. s : (T+T) List
. | . | 3. t : T
. | . | 4. no_repeats(T+T;s)
. | . |    ⇒
. | . |    (∀s1,s2,s3:(T+T) List, x:T.
. | . |     s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])]) ⇒ paren(T;s2))
. | . | 5. paren(T;s)
. | . | 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. | . | 7. no_repeats(T+T;[inl(t)])
. | . | 8. l_disjoint(T+T;s;[inr(t)])
. | . | 9. no_repeats(T+T;s)
. | . | 10. no_repeats(T+T;[inr(t)])
. | . | 11. s2 : (T+T) List
. | . | 12. s3 : (T+T) List
. | . | 13. x : T
. | . | 14. inl(t) = inl(x)
. | . | 15. (s @ [inr(t)]) = (s2 @ [inr(x) / s3])
. | . | 16. s2 = (s @ nil)
. | . | 17. inr(t) = inr(x)
. | . | 18. nil = s3
. | . | ⊢  paren(T;s2)
. | . |   by (SubstFor s2 0)
. | . |      THEN
. | . |      (RWO Thm* ∀l:T List. (l @ nil) ~ l    {append_nil_sq} 0)
. | . |
. | . .\
. | . . 1. T : Type
. | . . 2. s : (T+T) List
. | . . 3. t : T
. | . . 4. no_repeats(T+T;s)
. | . .    ⇒
. | . .    (∀s1,s2,s3:(T+T) List, x:T.
. | . .     s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])]) ⇒ paren(T;s2))
. | . . 5. paren(T;s)
. | . . 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. | . . 7. no_repeats(T+T;[inl(t)])
. | . . 8. l_disjoint(T+T;s;[inr(t)])
. | . . 9. no_repeats(T+T;s)
. | . . 10. no_repeats(T+T;[inr(t)])
. | . . 11. s2 : (T+T) List
```

```
. | . . 12. s3 : (T+T) List
. | . . 13. x : T
. | . . 14. inl(t) = inl(x)
. | . . 15. (s @ [inr(t)]) = (s2 @ [inr(x) / s3])
. | . . 16. u : T+T
. | . . 17. v : (T+T) List
. | . . 18. s2 = (s @ [u / v])
. | . . 19. inr(t) = u
. | . . 20. nil = (v @ [inr(x) / s3])
. | . . ⊢  paren(T;s2)  by ObviousContradiction [-1]
. |
. .\
. . 1. T : Type
. . 2. s : (T+T) List
. . 3. t : T
. . 4. no_repeats(T+T;s)
. . |  ⇒
. . |  (∀s1,s2,s3:(T+T) List, x:T.
. . |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])]) ⇒ paren(T;s2))
. . 5. paren(T;s)
. . 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. . 7. no_repeats(T+T;[inl(t)])
. . 8. l_disjoint(T+T;s;[inr(t)])
. . 9. no_repeats(T+T;s)
. . 10. no_repeats(T+T;[inr(t)])
. . 11. u : T+T
. . 12. v : (T+T) List
. . 13. s2 : (T+T) List
. . 14. s3 : (T+T) List
. . 15. x : T
. . 16. inl(t) = u
. . 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . ⊢  paren(T;s2)  by (EqualAppendsD -1) THEN (Analyze -1)
. . |
. . |\
. . | 1. T : Type
. . | 2. s : (T+T) List
. . | 3. t : T
. . | 4. no_repeats(T+T;s)
. . | |  ⇒
. . | |  (∀s1,s2,s3:(T+T) List, x:T.
. . | |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])]) ⇒ paren(T;s2))
. . | 5. paren(T;s)
. . | 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. . | 7. no_repeats(T+T;[inl(t)])
. . | 8. l_disjoint(T+T;s;[inr(t)])
. . | 9. no_repeats(T+T;s)
. . | 10. no_repeats(T+T;[inr(t)])
. . | 11. u : T+T
. . | 12. v : (T+T) List
. . | 13. s2 : (T+T) List
. . | 14. s3 : (T+T) List
. . | 15. x : T
. . | 16. inl(t) = u
. . | 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
```

```
. . | 18. e : (T+T) List
. . | 19. s = (v @ e)
. . | 20. [inl(x) / (s2 @ [inr(x) / s3])] = (e @ [inr(t)])
. . | ⊢  paren(T;s2)
. . | | by (Analyze -3) THEN (All Reduce) THEN (SplitCons -1)
. . | |
. . | |\
. . | | 1. T : Type
. . | | 2. s : (T+T) List
. . | | 3. t : T
. . | | 4. no_repeats(T+T;s)
. . | |    ⇒
. . | |    (∀s1,s2,s3:(T+T) List, x:T.
. . | |     s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])]) ⇒ paren(T;s2))
. . | | 5. paren(T;s)
. . | | 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. . | | 7. no_repeats(T+T;[inl(t)])
. . | | 8. l_disjoint(T+T;s;[inr(t)])
. . | | 9. no_repeats(T+T;s)
. . | | 10. no_repeats(T+T;[inr(t)])
. . | | 11. u : T+T
. . | | 12. v : (T+T) List
. . | | 13. s2 : (T+T) List
. . | | 14. s3 : (T+T) List
. . | | 15. x : T
. . | | 16. inl(t) = u
. . | | 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | | 18. s = (v @ nil)
. . | | 19. inl(x) = inr(t)
. . | | 20. (s2 @ [inr(x) / s3]) = nil
. . | | ⊢  paren(T;s2)  by ObviousContradiction [-1]
. . | |
. . | .\
. . | . 1. T : Type
. . | . 2. s : (T+T) List
. . | . 3. t : T
. . | . 4. no_repeats(T+T;s)
. . | . |  ⇒
. . | . |  (∀s1,s2,s3:(T+T) List, x:T.
. . | . |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])]) ⇒ paren(T;s2))
. . | . 5. paren(T;s)
. . | . 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. . | . 7. no_repeats(T+T;[inl(t)])
. . | . 8. l_disjoint(T+T;s;[inr(t)])
. . | . 9. no_repeats(T+T;s)
. . | . 10. no_repeats(T+T;[inr(t)])
. . | . 11. u : T+T
. . | . 12. v : (T+T) List
. . | . 13. s2 : (T+T) List
. . | . 14. s3 : (T+T) List
. . | . 15. x : T
. . | . 16. inl(t) = u
. . | . 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . 18. u1 : T+T
. . | . 19. v1 : (T+T) List
```

```
. . | . 20. s = (v @ [u1 / v1])
. . | . 21. inl(x) = u1
. . | . 22. (s2 @ [inr(x) / s3]) = (v1 @ [inr(t)])
. . | . ⊢  paren(T;s2)
. . | . | by (RevHypSubst -2 -3) THEN (Thin -2)
. . | . |    THEN
. . | . |    (EqualAppendsD -1)
. . | . |    THEN
. . | . |    (Analyze -1)
. . | . |
. . | . |\
. . | . | 1. T : Type
. . | . | 2. s : (T+T) List
. . | . | 3. t : T
. . | . | 4. no_repeats(T+T;s)
. . | . | |  ⇒
. . | . | |  (∀s1,s2,s3:(T+T) List, x:T.
. . | . | |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . | |   ⇒
. . | . | |   paren(T;s2))
. . | . | 5. paren(T;s)
. . | . | 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. . | . | 7. no_repeats(T+T;[inl(t)])
. . | . | 8. l_disjoint(T+T;s;[inr(t)])
. . | . | 9. no_repeats(T+T;s)
. . | . | 10. no_repeats(T+T;[inr(t)])
. . | . | 11. u : T+T
. . | . | 12. v : (T+T) List
. . | . | 13. s2 : (T+T) List
. . | . | 14. s3 : (T+T) List
. . | . | 15. x : T
. . | . | 16. inl(t) = u
. . | . | 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . | 18. T+T
. . | . | 19. v1 : (T+T) List
. . | . | 20. s = (v @ [inl(x) / v1])
. . | . | 21. (s2 @ [inr(x) / s3]) = (v1 @ [inr(t)])
. . | . | 22. e : (T+T) List
. . | . | 23. s2 = (v1 @ e)
. . | . | 24. [inr(t)] = (e @ [inr(x) / s3])
. . | . | ⊢  paren(T;s2)  by SingletonAppendD -1
. . | . | |
. . | . | |\
. . | . | | 1. T : Type
. . | . | | 2. s : (T+T) List
. . | . | | 3. t : T
. . | . | | 4. no_repeats(T+T;s)
. . | . | | |  ⇒
. . | . | | |  (∀s1,s2,s3:(T+T) List, x:T.
. . | . | | |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . | | |   ⇒
. . | . | | |   paren(T;s2))
. . | . | | 5. paren(T;s)
. . | . | | 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. . | . | | 7. no_repeats(T+T;[inl(t)])
```

```
. . | . | | 8. l_disjoint(T+T;s;[inr(t)])
. . | . | | 9. no_repeats(T+T;s)
. . | . | | 10. no_repeats(T+T;[inr(t)])
. . | . | | 11. u : T+T
. . | . | | 12. v : (T+T) List
. . | . | | 13. s2 : (T+T) List
. . | . | | 14. s3 : (T+T) List
. . | . | | 15. x : T
. . | . | | 16. inl(t) = u
. . | . | | 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . | | 18. T+T
. . | . | | 19. v1 : (T+T) List
. . | . | | 20. s = (v @ [inl(x) / v1])
. . | . | | 21. (s2 @ [inr(x) / s3]) = (v1 @ [inr(t)])
. . | . | | 22. e : (T+T) List
. . | . | | 23. s2 = (v1 @ e)
. . | . | | 24. [inr(t)] = (e @ [inr(x) / s3])
. . | . | | 25. e = nil
. . | . | | 26. [inr(x) / s3] = [inr(t)]
. . | . | | ⊢  paren(T;s2)  by SplitCons -1
. . | . | | |
. . | . | | 1. T : Type
. . | . | | 2. s : (T+T) List
. . | . | | 3. t : T
. . | . | | 4. no_repeats(T+T;s)
. . | . | | |  ⇒
. . | . | | |  (∀s1,s2,s3:(T+T) List, x:T.
. . | . | | |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . | | |   ⇒
. . | . | | |   paren(T;s2))
. . | . | | 5. paren(T;s)
. . | . | | 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. . | . | | 7. no_repeats(T+T;[inl(t)])
. . | . | | 8. l_disjoint(T+T;s;[inr(t)])
. . | . | | 9. no_repeats(T+T;s)
. . | . | | 10. no_repeats(T+T;[inr(t)])
. . | . | | 11. u : T+T
. . | . | | 12. v : (T+T) List
. . | . | | 13. s2 : (T+T) List
. . | . | | 14. s3 : (T+T) List
. . | . | | 15. x : T
. . | . | | 16. inl(t) = u
. . | . | | 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . | | 18. T+T
. . | . | | 19. v1 : (T+T) List
. . | . | | 20. s = (v @ [inl(x) / v1])
. . | . | | 21. (s2 @ [inr(x) / s3]) = (v1 @ [inr(t)])
. . | . | | 22. e : (T+T) List
. . | . | | 23. s2 = (v1 @ e)
. . | . | | 24. [inr(t)] = (e @ [inr(x) / s3])
. . | . | | 25. e = nil
. . | . | | 26. inr(x) = inr(t)

. . | . | | 27. s3 = nil
. . | . | | ⊢  paren(T;s2)
. . | . | | | by (Unfold `l_disjoint` 6) THEN (InstHyp [inl(x)] 6)
```

```
. . | . | | |    THEN
. . | . | | |    (Analyze -1)
. . | . | | |    THEN
. . | . | | |    (SubstFor s 0)
. . | . | | |
. . | . | | | 1. T : Type
. . | . | | | 2. s : (T+T) List
. . | . | | | 3. t : T
. . | . | | | 4. no_repeats(T+T;s)
. . | . | | |    ⇒
. . | . | | |    (∀s1,s2,s3:(T+T) List, x:T.
. . | . | | |     s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . | | |     ⇒
. . | . | | |     paren(T;s2))
. . | . | | | 5. paren(T;s)
. . | . | | | 6. ∀x:T+T. ¬((x ∈ [inl(t)]) & (x ∈ s @ [inr(t)]))
. . | . | | | 7. no_repeats(T+T;[inl(t)])
. . | . | | | 8. l_disjoint(T+T;s;[inr(t)])
. . | . | | | 9. no_repeats(T+T;s)
. . | . | | | 10. no_repeats(T+T;[inr(t)])
. . | . | | | 11. u : T+T
. . | . | | | 12. v : (T+T) List
. . | . | | | 13. s2 : (T+T) List
. . | . | | | 14. s3 : (T+T) List
. . | . | | | 15. x : T
. . | . | | | 16. inl(t) = u
. . | . | | | 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . | | | 18. T+T
. . | . | | | 19. v1 : (T+T) List
. . | . | | | 20. s = (v @ [inl(x) / v1])
. . | . | | | 21. (s2 @ [inr(x) / s3]) = (v1 @ [inr(t)])
. . | . | | | 22. e : (T+T) List
. . | . | | | 23. s2 = (v1 @ e)
. . | . | | | 24. [inr(t)] = (e @ [inr(x) / s3])
. . | . | | | 25. e = nil
. . | . | | | 26. inr(x) = inr(t)
. . | . | | | 27. s3 = nil
. . | . | | | ⊢  (inl(x) ∈ [inl(t)])
. . | . | | |    & (inl(x) ∈ (v @ [inl(x) / v1]) @ [inr(t)])
. . | . | | |   by ObviousFrom [-2]
. . | . | |
. . | . | .\
. . | . | . 1. T : Type
. . | . | . 2. s : (T+T) List
. . | . | . 3. t : T
. . | . | . 4. no_repeats(T+T;s)
. . | . | .    ⇒
. . | . | .    (∀s1,s2,s3:(T+T) List, x:T.
. . | . | .     s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . | .     ⇒
. . | . | .     paren(T;s2))
. . | . | . 5. paren(T;s)
. . | . | . 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. . | . | . 7. no_repeats(T+T;[inl(t)])
. . | . | . 8. l_disjoint(T+T;s;[inr(t)])
```

```
. . | . | . 9. no_repeats(T+T;s)
. . | . | . 10. no_repeats(T+T;[inr(t)])
. . | . | . 11. u : T+T
. . | . | . 12. v : (T+T) List
. . | . | . 13. s2 : (T+T) List
. . | . | . 14. s3 : (T+T) List
. . | . | . 15. x : T
. . | . | . 16. inl(t) = u
. . | . | . 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . | . 18. T+T
. . | . | . 19. v1 : (T+T) List
. . | . | . 20. s = (v @ [inl(x) / v1])
. . | . | . 21. (s2 @ [inr(x) / s3]) = (v1 @ [inr(t)])
. . | . | . 22. e : (T+T) List
. . | . | . 23. s2 = (v1 @ e)
. . | . | . 24. [inr(t)] = (e @ [inr(x) / s3])
. . | . | . 25. [inr(x) / s3] = nil
. . | . | . 26. e = [inr(t)]
. . | . | . ⊢  paren(T;s2)  by ObviousContradiction [-2]
. . | . |
. . | . .\
. . | . . 1. T : Type
. . | . . 2. s : (T+T) List
. . | . . 3. t : T
. . | . . 4. no_repeats(T+T;s)
. . | . . |  ⇒
. . | . . |  (∀s1,s2,s3:(T+T) List, x:T.
. . | . . |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . . |   ⇒
. . | . . |   paren(T;s2))
. . | . . 5. paren(T;s)
. . | . . 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. . | . . 7. no_repeats(T+T;[inl(t)])
. . | . . 8. l_disjoint(T+T;s;[inr(t)])
. . | . . 9. no_repeats(T+T;s)
. . | . . 10. no_repeats(T+T;[inr(t)])
. . | . . 11. u : T+T
. . | . . 12. v : (T+T) List
. . | . . 13. s2 : (T+T) List
. . | . . 14. s3 : (T+T) List
. . | . . 15. x : T
. . | . . 16. inl(t) = u
. . | . . 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . . 18. T+T
. . | . . 19. v1 : (T+T) List
. . | . . 20. s = (v @ [inl(x) / v1])
. . | . . 21. (s2 @ [inr(x) / s3]) = (v1 @ [inr(t)])
. . | . . 22. e : (T+T) List
. . | . . 23. v1 = (s2 @ e)
. . | . . 24. [inr(x) / s3] = (e @ [inr(t)])
. . | . . ⊢  paren(T;s2)
. . | . . | by (Analyze -3) THEN (All Reduce) THEN (SplitCons -1)
. . | . . |
. . | . . |\
. . | . . | 1. T : Type
```

```
. . | . . | 2. s : (T+T) List
. . | . . | 3. t : T
. . | . . | 4. no_repeats(T+T;s)
. . | . . | |  ⇒
. . | . . | |  (∀s1,s2,s3:(T+T) List, x:T.
. . | . . | |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . . | |   ⇒
. . | . . | |   paren(T;s2))
. . | . . | 5. paren(T;s)
. . | . . | 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. . | . . | 7. no_repeats(T+T;[inl(t)])
. . | . . | 8. l_disjoint(T+T;s;[inr(t)])
. . | . . | 9. no_repeats(T+T;s)
. . | . . | 10. no_repeats(T+T;[inr(t)])
. . | . . | 11. u : T+T
. . | . . | 12. v : (T+T) List
. . | . . | 13. s2 : (T+T) List
. . | . . | 14. s3 : (T+T) List
. . | . . | 15. x : T
. . | . . | 16. inl(t) = u
. . | . . | 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . . | 18. T+T
. . | . . | 19. v1 : (T+T) List
. . | . . | 20. s = (v @ [inl(x) / v1])
. . | . . | 21. (s2 @ [inr(x) / s3]) = (v1 @ [inr(t)])
. . | . . | 22. v1 = (s2 @ nil)
. . | . . | 23. inr(x) = inr(t)
. . | . . | 24. s3 = nil
. . | . . | ⊢  paren(T;s2)
. . | . . | | by (Unfold ‘l_disjoint‘ 6) THEN (InstHyp [inl(x)] 6)
. . | . . | |    THEN
. . | . . | |    (Analyze -1)
. . | . . | |    THEN
. . | . . | |    (SubstFor s 0)
. . | . . | |
. . | . . | 1. T : Type
. . | . . | 2. s : (T+T) List
. . | . . | 3. t : T
. . | . . | 4. no_repeats(T+T;s)
. . | . . |    ⇒
. . | . . |    (∀s1,s2,s3:(T+T) List, x:T.
. . | . . |     s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . . |     ⇒
. . | . . |     paren(T;s2))
. . | . . | 5. paren(T;s)
. . | . . | 6. ∀x:T+T. ¬((x ∈ [inl(t)]) & (x ∈ s @ [inr(t)]))
. . | . . | 7. no_repeats(T+T;[inl(t)])
. . | . . | 8. l_disjoint(T+T;s;[inr(t)])
. . | . . | 9. no_repeats(T+T;s)
. . | . . | 10. no_repeats(T+T;[inr(t)])
. . | . . | 11. u : T+T
. . | . . | 12. v : (T+T) List
. . | . . | 13. s2 : (T+T) List
. . | . . | 14. s3 : (T+T) List
. . | . . | 15. x : T
```

```
. . | . . | 16. inl(t) = u
. . | . . | 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . . | 18. T+T
. . | . . | 19. v1 : (T+T) List
. . | . . | 20. s = (v @ [inl(x) / v1])
. . | . . | 21. (s2 @ [inr(x) / s3]) = (v1 @ [inr(t)])
. . | . . | 22. v1 = (s2 @ nil)
. . | . . | 23. inr(x) = inr(t)
. . | . . | 24. s3 = nil
. . | . . | ⊢  (inl(x) ∈ [inl(t)])
. . | . . |    & (inl(x) ∈ (v @ [inl(x) / v1]) @ [inr(t)])
. . | . . |   by ObviousFrom [-2]
. . | . . |
. . | . . .\
. . | . . . 1. T : Type
. . | . . . 2. s : (T+T) List
. . | . . . 3. t : T
. . | . . . 4. no_repeats(T+T;s)
. . | . . . |  ⇒
. . | . . . |  (∀s1,s2,s3:(T+T) List, x:T.
. . | . . . |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . . . |   ⇒
. . | . . . |   paren(T;s2))
. . | . . . 5. paren(T;s)
. . | . . . 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. . | . . . 7. no_repeats(T+T;[inl(t)])
. . | . . . 8. l_disjoint(T+T;s;[inr(t)])
. . | . . . 9. no_repeats(T+T;s)
. . | . . . 10. no_repeats(T+T;[inr(t)])
. . | . . . 11. u : T+T
. . | . . . 12. v : (T+T) List
. . | . . . 13. s2 : (T+T) List
. . | . . . 14. s3 : (T+T) List
. . | . . . 15. x : T
. . | . . . 16. inl(t) = u
. . | . . . 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . . . 18. T+T
. . | . . . 19. v1 : (T+T) List
. . | . . . 20. s = (v @ [inl(x) / v1])
. . | . . . 21. (s2 @ [inr(x) / s3]) = (v1 @ [inr(t)])
. . | . . . 22. u2 : T+T
. . | . . . 23. v2 : (T+T) List
. . | . . . 24. v1 = (s2 @ [u2 / v2])
. . | . . . 25. inr(x) = u2
. . | . . . 26. s3 = (v2 @ [inr(t)])
. . | . . . ⊢  paren(T;s2)  by (RevHypSubst -2 -3) THEN (Thin -2)
. . | . . . |
. . | . . . 1. T : Type
. . | . . . 2. s : (T+T) List
. . | . . . 3. t : T
. . | . . . 4. no_repeats(T+T;s)
. . | . . . |  ⇒
. . | . . . |  (∀s1,s2,s3:(T+T) List, x:T.
. . | . . . |   s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . . . |   ⇒
```

```
. . | . . . |   paren(T;s2))
. . | . . . 5. paren(T;s)
. . | . . . 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. . | . . . 7. no_repeats(T+T;[inl(t)])
. . | . . . 8. l_disjoint(T+T;s;[inr(t)])
. . | . . . 9. no_repeats(T+T;s)
. . | . . . 10. no_repeats(T+T;[inr(t)])
. . | . . . 11. u : T+T
. . | . . . 12. v : (T+T) List
. . | . . . 13. s2 : (T+T) List
. . | . . . 14. s3 : (T+T) List
. . | . . . 15. x : T
. . | . . . 16. inl(t) = u
. . | . . . 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . . . 18. T+T
. . | . . . 19. v1 : (T+T) List
. . | . . . 20. s = (v @ [inl(x) / v1])
. . | . . . 21. (s2 @ [inr(x) / s3]) = (v1 @ [inr(t)])
. . | . . . 22. T+T
. . | . . . 23. v2 : (T+T) List
. . | . . . 24. v1 = (s2 @ [inr(x) / v2])
. . | . . . 25. s3 = (v2 @ [inr(t)])
. . | . . . ⊢  paren(T;s2)  by InstHyp [v;s2;v2;x] 4
. . | . . . |
. . | . . . .....antecedent .....
. . | . . . 1. T : Type
. . | . . . 2. s : (T+T) List
. . | . . . 3. t : T
. . | . . . 4. no_repeats(T+T;s)
. . | . . .    ⇒
. . | . . .    (∀s1,s2,s3:(T+T) List, x:T.
. . | . . .     s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . . .     ⇒
. . | . . .     paren(T;s2))
. . | . . . 5. paren(T;s)
. . | . . . 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. . | . . . 7. no_repeats(T+T;[inl(t)])
. . | . . . 8. l_disjoint(T+T;s;[inr(t)])
. . | . . . 9. no_repeats(T+T;s)
. . | . . . 10. no_repeats(T+T;[inr(t)])
. . | . . . 11. u : T+T
. . | . . . 12. v : (T+T) List
. . | . . . 13. s2 : (T+T) List
. . | . . . 14. s3 : (T+T) List
. . | . . . 15. x : T
. . | . . . 16. inl(t) = u
. . | . . . 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . | . . . 18. T+T
. . | . . . 19. v1 : (T+T) List
. . | . . . 20. s = (v @ [inl(x) / v1])
. . | . . . 21. (s2 @ [inr(x) / s3]) = (v1 @ [inr(t)])
. . | . . . 22. T+T
. . | . . . 23. v2 : (T+T) List
. . | . . . 24. v1 = (s2 @ [inr(x) / v2])
. . | . . . 25. s3 = (v2 @ [inr(t)])
```

```
. . | . . . ⊢  s = (v @ [inl(x) / (s2 @ [inr(x) / v2])])
. . | . . .   by (SubstFor s 0) THEN Analyze THEN Analyze
. . |
. . .\
. . . 1. T : Type
. . . 2. s : (T+T) List
. . . 3. t : T
. . . 4. no_repeats(T+T;s)
. . .    ⇒
. . .    (∀s1,s2,s3:(T+T) List, x:T.
. . .     s = (s1 @ [inl(x) / (s2 @ [inr(x) / s3])]) ⇒ paren(T;s2))
. . . 5. paren(T;s)
. . . 6. l_disjoint(T+T;[inl(t)];s @ [inr(t)])
. . . 7. no_repeats(T+T;[inl(t)])
. . . 8. l_disjoint(T+T;s;[inr(t)])
. . . 9. no_repeats(T+T;s)
. . . 10. no_repeats(T+T;[inr(t)])
. . . 11. u : T+T
. . . 12. v : (T+T) List
. . . 13. s2 : (T+T) List
. . . 14. s3 : (T+T) List
. . . 15. x : T

. . . 16. inl(t) = u
. . . 17. (s @ [inr(t)]) = (v @ [inl(x) / (s2 @ [inr(x) / s3])])
. . . 18. e : (T+T) List
. . . 19. v = (s @ e)
. . . 20. [inr(t)] = (e @ [inl(x) / (s2 @ [inr(x) / s3])])
. . . ⊢  paren(T;s2)
. . .   by (ApFunToHypEquands ‘l’ ||l|| ℤ -1)
. . .      THEN
. . .      (RepeatFor 2
. . .       ((RWW
. . .         Thm* ∀as,bs:T List. ||as @ bs|| = ||as||+||bs|| ∈ ℤ
. . .         {length_append}
. . .         -1)
. . .        THEN
. . .        (Reduce -1)))
END
```