```
paren-df-traversal  THM

⊢  ∀G:Graph.
|  (∀x,y:Vertices(G). Dec(x = y))
|  ⇒
|  (∀s:traversal(G).
|   paren(Vertices(G);s)
|   ⇒
|   no_repeats(Vertices(G)+Vertices(G);s)
|   ⇒
|   df-traversal(G;s) ⇒ depthfirst-traversal(G;s))
| by Auto THEN (Unfold ‘depthfirst-traversal‘ 0)
|    THEN
|    (AllHyps
|     (λh.
|      (Unfold ‘df-traversal‘ h) THEN (Analyze h)
|      THEN
|      (InstHyp [i;s1;s2;j] h+1)))
|    THEN
|    (Inst
|     Thm* (∀x,y:T. Dec(x = y))
|          ⇒
|          (∀s:(T+T) List, i:T. Dec((inl(i) ∈ s)))
|     {decidable__l_member_paren}
|     [Vertices(G);s2;j])
|    THEN
|    (Analyze -1)
|
1. G : Graph
2. ∀x,y:Vertices(G). Dec(x = y)
3. s : traversal(G)
4. paren(Vertices(G);s)
5. no_repeats(Vertices(G)+Vertices(G);s)
6. ∀i:Vertices(G), s1,s2:traversal(G).
|  s = (s1 @ [inr(i)] @ s2)
|  ⇒
|  (∀j:Vertices(G). (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
7. ∀i:Vertices(G), s1,s2:traversal(G).
|  (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
|  ⇒
|  s = (s1 @ [inl(i)] @ s2)
|  ⇒
|  (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
8. i : Vertices(G)
9. s1 : traversal(G)
10. s2 : traversal(G)
11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
12. s = (s1 @ [inl(i)] @ s2)
13. j : Vertices(G)
14. ¬j = i
15. i-G->*j
16. (inr(j) ∈ s2)
17. ¬(inl(j) ∈ s2)
⊢  (inl(j) ∈ s2)
| by Inst
```

```
|    Thm* ∀s:T List, z:T.
|         (z ∈ s) ⇒ (∃s1,s2:T List. s = (s1 @ [z / s2]))
|    {l_member_decomp}
|    [Vertices(G)+Vertices(G);s2;inr(i)]
|
|\
| .....antecedent .....
| 1. G : Graph
| 2. ∀x,y:Vertices(G). Dec(x = y)
| 3. s : traversal(G)
| 4. paren(Vertices(G);s)
| 5. no_repeats(Vertices(G)+Vertices(G);s)
| 6. ∀i:Vertices(G), s1,s2:traversal(G).
|    s = (s1 @ [inr(i)] @ s2)
|    ⇒
|    (∀j:Vertices(G). (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
| 7. ∀i:Vertices(G), s1,s2:traversal(G).
|    (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
|    ⇒
|    s = (s1 @ [inl(i)] @ s2)
|    ⇒
|    (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
| 8. i : Vertices(G)
| 9. s1 : traversal(G)
| 10. s2 : traversal(G)
| 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
| 12. s = (s1 @ [inl(i)] @ s2)
| 13. j : Vertices(G)
| 14. ¬j = i
| 15. i-G->*j
| 16. (inr(j) ∈ s2)
| 17. ¬(inl(j) ∈ s2)
| ⊢  (inr(i) ∈ s2)
|   by FwdThru
|      Thm* ∀s:(T+T) List.
|           paren(T;s)
|           ⇒
|           (∀i:T, s1,s2:(T+T) List.
|            s = (s1 @ [inl(i)] @ s2) ⇒ (inr(i) ∈ s2))
|      {paren_order1}
|      [4;12]
|
.\
. 1. G : Graph
. 2. ∀x,y:Vertices(G). Dec(x = y)
. 3. s : traversal(G)
. 4. paren(Vertices(G);s)
. 5. no_repeats(Vertices(G)+Vertices(G);s)
. 6. ∀i:Vertices(G), s1,s2:traversal(G).
. |  s = (s1 @ [inr(i)] @ s2)
. |  ⇒
. |  (∀j:Vertices(G). (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. 7. ∀i:Vertices(G), s1,s2:traversal(G).
. |  (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. |  ⇒
```

```
. |  s = (s1 @ [inl(i)] @ s2)
. |  ⇒
. |  (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. 8. i : Vertices(G)
. 9. s1 : traversal(G)
. 10. s2 : traversal(G)
. 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. 12. s = (s1 @ [inl(i)] @ s2)
. 13. j : Vertices(G)
. 14. ¬j = i
. 15. i-G->*j
. 16. (inr(j) ∈ s2)
. 17. ¬(inl(j) ∈ s2)
. 18. ∃s1,s2@0:(Vertices(G)+Vertices(G)) List.
. |   s2 = (s1 @ [inr(i) / s2@0])
. ⊢  (inl(j) ∈ s2)
. | by ExRepD THEN (Assert ((inr(j) ∈ s3) ∨ (inr(j) ∈ s2@0)))
. |
. |\
. | .....assertion .....
. | 1. G : Graph
. | 2. ∀x,y:Vertices(G). Dec(x = y)
. | 3. s : traversal(G)
. | 4. paren(Vertices(G);s)
. | 5. no_repeats(Vertices(G)+Vertices(G);s)
. | 6. ∀i:Vertices(G), s1,s2:traversal(G).
. | |  s = (s1 @ [inr(i)] @ s2)
. | |  ⇒
. | |  (∀j:Vertices(G). (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. | 7. ∀i:Vertices(G), s1,s2:traversal(G).
. | |  (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. | |  ⇒
. | |  s = (s1 @ [inl(i)] @ s2)
. | |  ⇒
. | |  (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. | 8. i : Vertices(G)
. | 9. s1 : traversal(G)
. | 10. s2 : traversal(G)
. | 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. | 12. s = (s1 @ [inl(i)] @ s2)
. | 13. j : Vertices(G)
. | 14. ¬j = i
. | 15. i-G->*j
. | 16. (inr(j) ∈ s2)
. | 17. ¬(inl(j) ∈ s2)
. | 18. s3 : (Vertices(G)+Vertices(G)) List
. | 19. s2@0 : (Vertices(G)+Vertices(G)) List
. | 20. s2 = (s3 @ [inr(i) / s2@0])
. | ⊢  (inr(j) ∈ s3) ∨ (inr(j) ∈ s2@0)
. | | by (MoveToHyp 0 -5) THEN (HypSubst -2 -1)
. | |
. | 1. G : Graph
. | 2. ∀x,y:Vertices(G). Dec(x = y)
. | 3. s : traversal(G)
. | 4. paren(Vertices(G);s)
```

```
. | 5. no_repeats(Vertices(G)+Vertices(G);s)
. | 6. ∀i:Vertices(G), s1,s2:traversal(G).
. |    s = (s1 @ [inr(i)] @ s2)
. |    ⇒
. |    (∀j:Vertices(G). (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. | 7. ∀i:Vertices(G), s1,s2:traversal(G).
. |    (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. |    ⇒
. |    s = (s1 @ [inl(i)] @ s2)
. |    ⇒
. |    (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. | 8. i : Vertices(G)
. | 9. s1 : traversal(G)
. | 10. s2 : traversal(G)
. | 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. | 12. s = (s1 @ [inl(i)] @ s2)
. | 13. j : Vertices(G)
. | 14. ¬j = i
. | 15. i-G->*j
. | 16. ¬(inl(j) ∈ s2)
. | 17. s3 : (Vertices(G)+Vertices(G)) List
. | 18. s2@0 : (Vertices(G)+Vertices(G)) List
. | 19. s2 = (s3 @ [inr(i) / s2@0])
. | 20. (inr(j) ∈ s3 @ [inr(i) / s2@0])
. | ⊢  (inr(j) ∈ s3) ∨ (inr(j) ∈ s2@0)  by ObviousFrom [14;-1]
. |
. .\
. . 1. G : Graph
. . 2. ∀x,y:Vertices(G). Dec(x = y)
. . 3. s : traversal(G)
. . 4. paren(Vertices(G);s)
. . 5. no_repeats(Vertices(G)+Vertices(G);s)
. . 6. ∀i:Vertices(G), s1,s2:traversal(G).
. . |  s = (s1 @ [inr(i)] @ s2)
. . |  ⇒
. . |  (∀j:Vertices(G). (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. . 7. ∀i:Vertices(G), s1,s2:traversal(G).
. . |  (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. . |  ⇒
. . |  s = (s1 @ [inl(i)] @ s2)
. . |  ⇒
. . |  (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. . 8. i : Vertices(G)
. . 9. s1 : traversal(G)
. . 10. s2 : traversal(G)
. . 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. . 12. s = (s1 @ [inl(i)] @ s2)
. . 13. j : Vertices(G)
. . 14. ¬j = i
. . 15. i-G->*j
. . 16. (inr(j) ∈ s2)
. . 17. ¬(inl(j) ∈ s2)
. . 18. s3 : (Vertices(G)+Vertices(G)) List
. . 19. s2@0 : (Vertices(G)+Vertices(G)) List
. . 20. s2 = (s3 @ [inr(i) / s2@0])
```

```
. . 21. (inr(j) ∈ s3) ∨ (inr(j) ∈ s2@0)
. . ⊢  (inl(j) ∈ s2)  by Analyze -1
. . |
. . |\
. . | 1. G : Graph
. . | 2. ∀x,y:Vertices(G). Dec(x = y)
. . | 3. s : traversal(G)
. . | 4. paren(Vertices(G);s)
. . | 5. no_repeats(Vertices(G)+Vertices(G);s)
. . | 6. ∀i:Vertices(G), s1,s2:traversal(G).
. . | |  s = (s1 @ [inr(i)] @ s2)
. . | |  ⇒
. . | |  (∀j:Vertices(G). (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. . | 7. ∀i:Vertices(G), s1,s2:traversal(G).
. . | |  (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. . | |  ⇒
. . | |  s = (s1 @ [inl(i)] @ s2)
. . | |  ⇒
. . | |  (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. . | 8. i : Vertices(G)
. . | 9. s1 : traversal(G)
. . | 10. s2 : traversal(G)
. . | 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. . | 12. s = (s1 @ [inl(i)] @ s2)
. . | 13. j : Vertices(G)
. . | 14. ¬j = i
. . | 15. i-G->*j
. . | 16. (inr(j) ∈ s2)
. . | 17. ¬(inl(j) ∈ s2)
. . | 18. s3 : (Vertices(G)+Vertices(G)) List
. . | 19. s2@0 : (Vertices(G)+Vertices(G)) List
. . | 20. s2 = (s3 @ [inr(i) / s2@0])
. . | 21. (inr(j) ∈ s3)
. . | ⊢  (inl(j) ∈ s2)
. . | | by Assert (s = (s1 @ [inl(i)] @ s3 @ [inr(i)] @ s2@0))
. . | |
. . | |\
. . | | .....assertion .....
. . | | 1. G : Graph
. . | | 2. ∀x,y:Vertices(G). Dec(x = y)
. . | | 3. s : traversal(G)
. . | | 4. paren(Vertices(G);s)
. . | | 5. no_repeats(Vertices(G)+Vertices(G);s)
. . | | 6. ∀i:Vertices(G), s1,s2:traversal(G).
. . | |    s = (s1 @ [inr(i)] @ s2)
. . | |    ⇒
. . | |    (∀j:Vertices(G).
. . | |     (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. . | | 7. ∀i:Vertices(G), s1,s2:traversal(G).
. . | |    (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. . | |    ⇒
. . | |    s = (s1 @ [inl(i)] @ s2)
. . | |    ⇒
. . | |    (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. . | | 8. i : Vertices(G)
```

```
. . | | 9. s1 : traversal(G)
. . | | 10. s2 : traversal(G)
. . | | 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. . | | 12. s = (s1 @ [inl(i)] @ s2)
. . | | 13. j : Vertices(G)
. . | | 14. ¬j = i
. . | | 15. i-G->*j
. . | | 16. (inr(j) ∈ s2)
. . | | 17. ¬(inl(j) ∈ s2)
. . | | 18. s3 : (Vertices(G)+Vertices(G)) List
. . | | 19. s2@0 : (Vertices(G)+Vertices(G)) List
. . | | 20. s2 = (s3 @ [inr(i) / s2@0])
. . | | 21. (inr(j) ∈ s3)
. . | | ⊢  s = (s1 @ [inl(i)] @ s3 @ [inr(i)] @ s2@0)
. . | |   by (SubstFor s 0) THEN (SubstFor s2 0) THEN (Reduce 0)
. . | |
. . | .\
. . | . 1. G : Graph
. . | . 2. ∀x,y:Vertices(G). Dec(x = y)
. . | . 3. s : traversal(G)
. . | . 4. paren(Vertices(G);s)
. . | . 5. no_repeats(Vertices(G)+Vertices(G);s)
. . | . 6. ∀i:Vertices(G), s1,s2:traversal(G).
. . | . |  s = (s1 @ [inr(i)] @ s2)
. . | . |  ⇒
. . | . |  (∀j:Vertices(G).
. . | . |   (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. . | . 7. ∀i:Vertices(G), s1,s2:traversal(G).
. . | . |  (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. . | . |  ⇒
. . | . |  s = (s1 @ [inl(i)] @ s2)
. . | . |  ⇒
. . | . |  (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. . | . 8. i : Vertices(G)
. . | . 9. s1 : traversal(G)
. . | . 10. s2 : traversal(G)
. . | . 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. . | . 12. s = (s1 @ [inl(i)] @ s2)
. . | . 13. j : Vertices(G)
. . | . 14. ¬j = i
. . | . 15. i-G->*j
. . | . 16. (inr(j) ∈ s2)
. . | . 17. ¬(inl(j) ∈ s2)
. . | . 18. s3 : (Vertices(G)+Vertices(G)) List
. . | . 19. s2@0 : (Vertices(G)+Vertices(G)) List
. . | . 20. s2 = (s3 @ [inr(i) / s2@0])
. . | . 21. (inr(j) ∈ s3)
. . | . 22. s = (s1 @ [inl(i)] @ s3 @ [inr(i)] @ s2@0)
. . | . ⊢  (inl(j) ∈ s2)  by Assert paren(Vertices(G);s3)
. . | . |
. . | . |\
. . | . | .....assertion .....
. . | . | 1. G : Graph
. . | . | 2. ∀x,y:Vertices(G). Dec(x = y)
. . | . | 3. s : traversal(G)
```

```
. . | . | 4. paren(Vertices(G);s)
. . | . | 5. no_repeats(Vertices(G)+Vertices(G);s)
. . | . | 6. ∀i:Vertices(G), s1,s2:traversal(G).
. . | . |    s = (s1 @ [inr(i)] @ s2)
. . | . |    ⇒
. . | . |    (∀j:Vertices(G).
. . | . |     (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. . | . | 7. ∀i:Vertices(G), s1,s2:traversal(G).
. . | . |    (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. . | . |    ⇒
. . | . |    s = (s1 @ [inl(i)] @ s2)
. . | . |    ⇒
. . | . |    (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. . | . | 8. i : Vertices(G)
. . | . | 9. s1 : traversal(G)
. . | . | 10. s2 : traversal(G)
. . | . | 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. . | . | 12. s = (s1 @ [inl(i)] @ s2)
. . | . | 13. j : Vertices(G)
. . | . | 14. ¬j = i
. . | . | 15. i-G->*j
. . | . | 16. (inr(j) ∈ s2)
. . | . | 17. ¬(inl(j) ∈ s2)
. . | . | 18. s3 : (Vertices(G)+Vertices(G)) List
. . | . | 19. s2@0 : (Vertices(G)+Vertices(G)) List
. . | . | 20. s2 = (s3 @ [inr(i) / s2@0])
. . | . | 21. (inr(j) ∈ s3)
. . | . | 22. s = (s1 @ [inl(i)] @ s3 @ [inr(i)] @ s2@0)
. . | . | ⊢  paren(Vertices(G);s3)
. . | . |   by FwdThru Thm: paren_interval [4;-1]
. . | . |
. . | . .\
. . | . . 1. G : Graph
. . | . . 2. ∀x,y:Vertices(G). Dec(x = y)
. . | . . 3. s : traversal(G)
. . | . . 4. paren(Vertices(G);s)
. . | . . 5. no_repeats(Vertices(G)+Vertices(G);s)
. . | . . 6. ∀i:Vertices(G), s1,s2:traversal(G).
. . | . . |  s = (s1 @ [inr(i)] @ s2)
. . | . . |  ⇒
. . | . . |  (∀j:Vertices(G).
. . | . . |   (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. . | . . 7. ∀i:Vertices(G), s1,s2:traversal(G).
. . | . . |  (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. . | . . |  ⇒
. . | . . |  s = (s1 @ [inl(i)] @ s2)
. . | . . |  ⇒
. . | . . |  (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. . | . . 8. i : Vertices(G)
. . | . . 9. s1 : traversal(G)
. . | . . 10. s2 : traversal(G)
. . | . . 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. . | . . 12. s = (s1 @ [inl(i)] @ s2)
. . | . . 13. j : Vertices(G)
. . | . . 14. ¬j = i
```

```
. . | . . 15. i-G->*j
. . | . . 16. (inr(j) ∈ s2)
. . | . . 17. ¬(inl(j) ∈ s2)
. . | . . 18. s3 : (Vertices(G)+Vertices(G)) List
. . | . . 19. s2@0 : (Vertices(G)+Vertices(G)) List
. . | . . 20. s2 = (s3 @ [inr(i) / s2@0])
. . | . . 21. (inr(j) ∈ s3)
. . | . . 22. s = (s1 @ [inl(i)] @ s3 @ [inr(i)] @ s2@0)
. . | . . 23. paren(Vertices(G);s3)
. . | . . ⊢  (inl(j) ∈ s2)
. . | . . | by FwdThru
. . | . . |    Thm* ∀s':(T+T) List.
. . | . . |         paren(T;s')
. . | . . |         ⇒
. . | . . |         (∀i:T. (inr(i) ∈ s') ⇒ (inl(i) ∈ s'))
. . | . . |    {paren_balance2}
. . | . . |    [-1;-3]
. . | . . |
. . | . . 1. G : Graph
. . | . . 2. ∀x,y:Vertices(G). Dec(x = y)
. . | . . 3. s : traversal(G)
. . | . . 4. paren(Vertices(G);s)
. . | . . 5. no_repeats(Vertices(G)+Vertices(G);s)
. . | . . 6. ∀i:Vertices(G), s1,s2:traversal(G).
. . | . .    s = (s1 @ [inr(i)] @ s2)
. . | . .    ⇒
. . | . .    (∀j:Vertices(G).
. . | . .     (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. . | . . 7. ∀i:Vertices(G), s1,s2:traversal(G).
. . | . .    (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. . | . .    ⇒
. . | . .    s = (s1 @ [inl(i)] @ s2)
. . | . .    ⇒
. . | . .    (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. . | . . 8. i : Vertices(G)
. . | . . 9. s1 : traversal(G)
. . | . . 10. s2 : traversal(G)
. . | . . 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. . | . . 12. s = (s1 @ [inl(i)] @ s2)
. . | . . 13. j : Vertices(G)
. . | . . 14. ¬j = i
. . | . . 15. i-G->*j
. . | . . 16. (inr(j) ∈ s2)
. . | . . 17. ¬(inl(j) ∈ s2)
. . | . . 18. s3 : (Vertices(G)+Vertices(G)) List
. . | . . 19. s2@0 : (Vertices(G)+Vertices(G)) List
. . | . . 20. s2 = (s3 @ [inr(i) / s2@0])
. . | . . 21. (inr(j) ∈ s3)
. . | . . 22. s = (s1 @ [inl(i)] @ s3 @ [inr(i)] @ s2@0)
. . | . . 23. paren(Vertices(G);s3)
. . | . . 24. (inl(j) ∈ s3)
. . | . . ⊢  (inl(j) ∈ s2)
. . | . .   by (SubstFor s2 0) THEN (ObviousFrom [-1])
. . |
. . .\
```

```
. . . 1. G : Graph
. . . 2. ∀x,y:Vertices(G). Dec(x = y)
. . . 3. s : traversal(G)
. . . 4. paren(Vertices(G);s)
. . . 5. no_repeats(Vertices(G)+Vertices(G);s)
. . . 6. ∀i:Vertices(G), s1,s2:traversal(G).
. . . |  s = (s1 @ [inr(i)] @ s2)
. . . |  ⇒
. . . |  (∀j:Vertices(G). (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. . . 7. ∀i:Vertices(G), s1,s2:traversal(G).
. . . |  (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. . . |  ⇒
. . . |  s = (s1 @ [inl(i)] @ s2)
. . . |  ⇒
. . . |  (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. . . 8. i : Vertices(G)
. . . 9. s1 : traversal(G)
. . . 10. s2 : traversal(G)
. . . 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. . . 12. s = (s1 @ [inl(i)] @ s2)
. . . 13. j : Vertices(G)
. . . 14. ¬j = i
. . . 15. i-G->*j
. . . 16. (inr(j) ∈ s2)
. . . 17. ¬(inl(j) ∈ s2)
. . . 18. s3 : (Vertices(G)+Vertices(G)) List
. . . 19. s2@0 : (Vertices(G)+Vertices(G)) List
. . . 20. s2 = (s3 @ [inr(i) / s2@0])
. . . 21. (inr(j) ∈ s2@0)
. . . ⊢  (inl(j) ∈ s2)  by InstHyp [i;s1 @ [inl(i)] @ s3;s2@0;j] 6
. . . |
. . . |\
. . . | .....antecedent .....
. . . | 1. G : Graph
. . . | 2. ∀x,y:Vertices(G). Dec(x = y)
. . . | 3. s : traversal(G)
. . . | 4. paren(Vertices(G);s)
. . . | 5. no_repeats(Vertices(G)+Vertices(G);s)
. . . | 6. ∀i:Vertices(G), s1,s2:traversal(G).
. . . |    s = (s1 @ [inr(i)] @ s2)
. . . |    ⇒
. . . |    (∀j:Vertices(G).
. . . |     (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. . . | 7. ∀i:Vertices(G), s1,s2:traversal(G).
. . . |    (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. . . |    ⇒
. . . |    s = (s1 @ [inl(i)] @ s2)
. . . |    ⇒
. . . |    (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. . . | 8. i : Vertices(G)
. . . | 9. s1 : traversal(G)
. . . | 10. s2 : traversal(G)
. . . | 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. . . | 12. s = (s1 @ [inl(i)] @ s2)
. . . | 13. j : Vertices(G)
```

```
. . . | 14. ¬j = i
. . . | 15. i-G->*j
. . . | 16. (inr(j) ∈ s2)
. . . | 17. ¬(inl(j) ∈ s2)
. . . | 18. s3 : (Vertices(G)+Vertices(G)) List
. . . | 19. s2@0 : (Vertices(G)+Vertices(G)) List
. . . | 20. s2 = (s3 @ [inr(i) / s2@0])
. . . | 21. (inr(j) ∈ s2@0)
. . . | ⊢  s = ((s1 @ [inl(i)] @ s3) @ [inr(i)] @ s2@0)
. . . |   by (SubstFor s 0) THEN (SubstFor s2 0)
. . . |      THEN
. . . |      (RWW
. . . |       Thm* ∀as,bs,cs:T List.
. . . |            ((as @ bs) @ cs) = (as @ (bs @ cs))
. . . |       {append_assoc}
. . . |       0)
. . . |      THEN
. . . |      (Reduce 0)
. . . |
. . . |\
. . . | .....antecedent .....
. . . | 1. G : Graph
. . . | 2. ∀x,y:Vertices(G). Dec(x = y)
. . . | 3. s : traversal(G)
. . . | 4. paren(Vertices(G);s)
. . . | 5. no_repeats(Vertices(G)+Vertices(G);s)
. . . | 6. ∀i:Vertices(G), s1,s2:traversal(G).
. . . |    s = (s1 @ [inr(i)] @ s2)
. . . |    ⇒
. . . |    (∀j:Vertices(G).
. . . |     (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. . . | 7. ∀i:Vertices(G), s1,s2:traversal(G).
. . . |    (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. . . |    ⇒
. . . |    s = (s1 @ [inl(i)] @ s2)
. . . |    ⇒
. . . |    (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. . . | 8. i : Vertices(G)
. . . | 9. s1 : traversal(G)
. . . | 10. s2 : traversal(G)
. . . | 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. . . | 12. s = (s1 @ [inl(i)] @ s2)
. . . | 13. j : Vertices(G)
. . . | 14. ¬j = i
. . . | 15. i-G->*j
. . . | 16. (inr(j) ∈ s2)
. . . | 17. ¬(inl(j) ∈ s2)
. . . | 18. s3 : (Vertices(G)+Vertices(G)) List
. . . | 19. s2@0 : (Vertices(G)+Vertices(G)) List
. . . | 20. s2 = (s3 @ [inr(i) / s2@0])
. . . | 21. (inr(j) ∈ s2@0)
. . . | ⊢  ¬(inl(j) ∈ s2@0)
. . . |   by (ParallelOp -5) THEN (SubstFor s2 0)
. . . |      THEN
. . . |      (ObviousFrom [-1])
```

```
. . . |
. . . .\
. . . . 1. G : Graph
. . . . 2. ∀x,y:Vertices(G). Dec(x = y)
. . . . 3. s : traversal(G)
. . . . 4. paren(Vertices(G);s)
. . . . 5. no_repeats(Vertices(G)+Vertices(G);s)
. . . . 6. ∀i:Vertices(G), s1,s2:traversal(G).
. . . . |  s = (s1 @ [inr(i)] @ s2)
. . . . |  ⇒
. . . . |  (∀j:Vertices(G).
. . . . |   (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. . . . 7. ∀i:Vertices(G), s1,s2:traversal(G).
. . . . |  (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. . . . |  ⇒
. . . . |  s = (s1 @ [inl(i)] @ s2)
. . . . |  ⇒
. . . . |  (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. . . . 8. i : Vertices(G)
. . . . 9. s1 : traversal(G)
. . . . 10. s2 : traversal(G)
. . . . 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. . . . 12. s = (s1 @ [inl(i)] @ s2)
. . . . 13. j : Vertices(G)
. . . . 14. ¬j = i
. . . . 15. i-G->*j
. . . . 16. (inr(j) ∈ s2)
. . . . 17. ¬(inl(j) ∈ s2)
. . . . 18. s3 : (Vertices(G)+Vertices(G)) List
. . . . 19. s2@0 : (Vertices(G)+Vertices(G)) List
. . . . 20. s2 = (s3 @ [inr(i) / s2@0])
. . . . 21. (inr(j) ∈ s2@0)
. . . . 22. j-G->*i
. . . . ⊢  (inl(j) ∈ s2)
. . . . | by ((InstHyp [i] 11) THENA (Auto THEN Connect))
. . . . |    THEN
. . . . |    (Analyze -1)
. . . . |
. . . . 1. G : Graph
. . . . 2. ∀x,y:Vertices(G). Dec(x = y)
. . . . 3. s : traversal(G)
. . . . 4. paren(Vertices(G);s)
. . . . 5. no_repeats(Vertices(G)+Vertices(G);s)
. . . . 6. ∀i:Vertices(G), s1,s2:traversal(G).
. . . .    s = (s1 @ [inr(i)] @ s2)
. . . .    ⇒
. . . .    (∀j:Vertices(G).
. . . .     (inr(j) ∈ s2) ⇒ ¬(inl(j) ∈ s2) ⇒ j-G->*i)
. . . . 7. ∀i:Vertices(G), s1,s2:traversal(G).
. . . .    (∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j))
. . . .    ⇒
. . . .    s = (s1 @ [inl(i)] @ s2)

. . . .    ⇒
. . . .    (∀j:Vertices(G). i-G->*j ⇒ (inr(j) ∈ s2))
. . . . 8. i : Vertices(G)
```

```
. . . . 9. s1 : traversal(G)
. . . . 10. s2 : traversal(G)
. . . . 11. ∀j:Vertices(G). i-G->*j ⇒ ¬non-trivial-loop(G;j)
. . . . 12. s = (s1 @ [inl(i)] @ s2)
. . . . 13. j : Vertices(G)
. . . . 14. ¬j = i
. . . . 15. i-G->*j
. . . . 16. (inr(j) ∈ s2)
. . . . 17. ¬(inl(j) ∈ s2)
. . . . 18. s3 : (Vertices(G)+Vertices(G)) List
. . . . 19. s2@0 : (Vertices(G)+Vertices(G)) List
. . . . 20. s2 = (s3 @ [inr(i) / s2@0])
. . . . 21. (inr(j) ∈ s2@0)
. . . . 22. j-G->*i
. . . . ⊢  non-trivial-loop(G;i)
. . . .   by (Unfold `non-trivial-loop` 0) THEN (InstConcl [j])
END
```