```
dfsl-traversal-append  THM

⊢  For any graph
|  ∀L1,L2:V List, s:Traversal.
|  L1-->*L2
|  ⇒
|  dfsl-traversal(the_graph;L1;s)
|  ⇒
|  dfsl-traversal(the_graph;L1 @ L2;s)
| by Auto THEN (ParallelOp -1) THEN (Analyze -1) THEN (Analyze 0)
|    THEN
|    (Try Trivial)
|    THEN
|    (Try
|     (((Thin -1) THEN (RepeatFor 3 (ParallelOp -1))
|       THEN
|       (RWO
|        Thm* For any graph
|             ∀A,B:V List, i:V. A @ B-->*i ⟺ A-->*i ∨ B-->*i
|        {list_DASH_connect_DASH_append}
|        0))
|      THEN
|      OrLeft))
|    THEN
|    (ParallelOp -1)
|    THEN
|    (Try
|     ((RepeatFor 2 (ParallelOp -1))
|      THEN
|      (RWO
|       Thm* For any graph
|            ∀A,B:V List, i:V. A @ B-->*i ⟺ A-->*i ∨ B-->*i
|       {list_DASH_connect_DASH_append}
|       0)
|      THEN
|      OrLeft))
|
1. the_graph : Graph
2. L1 : Vertices(the_graph) List
3. L2 : Vertices(the_graph) List
4. s : traversal(the_graph)
5. L1-the_graph->*L2
6. df-traversal(the_graph;s)
|  & (∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i)
7. ∀i:Vertices(the_graph).
|  L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
8. ∀L1@0,L2:Vertices(the_graph) List.
|  L1 = (L1@0 @ L2)
|  ⇒
|  (∃s1,s2:traversal(the_graph).
|   s = (s2 @ s1)
|   & paren(Vertices(the_graph);s1) & paren(Vertices(the_graph);s2)
|   & (∀j:Vertices(the_graph).
|      ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
|      & ((inl(j) ∈ s2) ⟺ L2-the_graph->*j & ¬L1@0-the_graph->*j)))
```

```
⊢  ∀L1@0,L2@0:Vertices(the_graph) List.
|  (L1 @ L2) = (L1@0 @ L2@0)
|  ⇒
|  (∃s1,s2:traversal(the_graph).
|   s = (s2 @ s1)
|   & paren(Vertices(the_graph);s1) & paren(Vertices(the_graph);s2)
|   & (∀j:Vertices(the_graph).
|      ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
|      & ((inl(j) ∈ s2) ⟺ L2@0-the_graph->*j & ¬L1@0-the_graph->*j)))
| by Auto THEN (EqualAppendsD -1) THEN (Analyze -1)
|
|\
| 1. the_graph : Graph
| 2. L1 : Vertices(the_graph) List
| 3. L2 : Vertices(the_graph) List
| 4. s : traversal(the_graph)
| 5. L1-the_graph->*L2
| 6. df-traversal(the_graph;s)
| 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
| 8. ∀i:Vertices(the_graph).
| |  L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
| 9. ∀L1@0,L2:Vertices(the_graph) List.
| |  L1 = (L1@0 @ L2)
| |  ⇒
| |  (∃s1,s2:traversal(the_graph).
| |   s = (s2 @ s1)
| |   & paren(Vertices(the_graph);s1) & paren(Vertices(the_graph);s2)
| |   & (∀j:Vertices(the_graph).
| |      ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
| |      & ((inl(j) ∈ s2) ⟺ L2-the_graph->*j & ¬L1@0-the_graph->*j)))
| 10. L1@0 : Vertices(the_graph) List
| 11. L2@0 : Vertices(the_graph) List
| 12. (L1 @ L2) = (L1@0 @ L2@0)
| 13. e : Vertices(the_graph) List
| 14. L1 = (L1@0 @ e)
| 15. L2@0 = (e @ L2)
| ⊢  ∃s1,s2:traversal(the_graph).
| |  s = (s2 @ s1)
| |  & paren(Vertices(the_graph);s1) & paren(Vertices(the_graph);s2)
| |  & (∀j:Vertices(the_graph).
| |     ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
| |     & ((inl(j) ∈ s2) ⟺ L2@0-the_graph->*j & ¬L1@0-the_graph->*j))
| | by (HypSubst -1 0)
| |    THEN
| |    (RWO
| |     Thm* For any graph
| |          ∀A,B:V List, i:V. A @ B-->*i ⟺ A-->*i ∨ B-->*i
| |     {list_DASH_connect_DASH_append}
| |     0)
| |    THEN
| |    (InstHyp [L1@0;e] 9)
| |    THEN
| |    (RepeatFor 7 (ParallelOp -1))
| |
| |\
```

```
| | 1. the_graph : Graph
| | 2. L1 : Vertices(the_graph) List
| | 3. L2 : Vertices(the_graph) List
| | 4. s : traversal(the_graph)
| | 5. L1-the_graph->*L2
| | 6. df-traversal(the_graph;s)
| | 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
| | 8. ∀i:Vertices(the_graph).
| |    L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
| | 9. ∀L1@0,L2:Vertices(the_graph) List.
| |    L1 = (L1@0 @ L2)
| |    ⇒
| |    (∃s1,s2:traversal(the_graph).
| |     s = (s2 @ s1)
| |     & paren(Vertices(the_graph);s1)
| |     & paren(Vertices(the_graph);s2)
| |     & (∀j:Vertices(the_graph).
| |        ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
| |        & ((inl(j) ∈ s2)
| |           ⟺
| |           L2-the_graph->*j & ¬L1@0-the_graph->*j)))
| | 10. L1@0 : Vertices(the_graph) List
| | 11. L2@0 : Vertices(the_graph) List
| | 12. (L1 @ L2) = (L1@0 @ L2@0)
| | 13. e : Vertices(the_graph) List
| | 14. L1 = (L1@0 @ e)
| | 15. L2@0 = (e @ L2)
| | 16. s1 : traversal(the_graph)
| | 17. s2 : traversal(the_graph)
| | 18. ∀j:Vertices(the_graph).
| |     ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
| |     & ((inl(j) ∈ s2) ⟺ e-the_graph->*j & ¬L1@0-the_graph->*j)
| | 19. j : Vertices(the_graph)
| | 20. (inl(j) ∈ s2)
| | 21. e-the_graph->*j
| | 22. ¬L1@0-the_graph->*j
| | ⊢  e-the_graph->*j ∨ L2-the_graph->*j  by OrLeft
| |
| .\
| . 1. the_graph : Graph
| . 2. L1 : Vertices(the_graph) List
| . 3. L2 : Vertices(the_graph) List
| . 4. s : traversal(the_graph)
| . 5. L1-the_graph->*L2
| . 6. df-traversal(the_graph;s)
| . 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
| . 8. ∀i:Vertices(the_graph).
| . |  L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
| . 9. ∀L1@0,L2:Vertices(the_graph) List.
| . |  L1 = (L1@0 @ L2)
| . |  ⇒
| . |  (∃s1,s2:traversal(the_graph).
| . |   s = (s2 @ s1)
| . |   & paren(Vertices(the_graph);s1)
| . |   & paren(Vertices(the_graph);s2)
```

```
| . |   & (∀j:Vertices(the_graph).
| . |      ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
| . |      & ((inl(j) ∈ s2)
| . |         ⟺
| . |         L2-the_graph->*j & ¬L1@0-the_graph->*j)))
| . 10. L1@0 : Vertices(the_graph) List
| . 11. L2@0 : Vertices(the_graph) List
| . 12. (L1 @ L2) = (L1@0 @ L2@0)
| . 13. e : Vertices(the_graph) List
| . 14. L1 = (L1@0 @ e)
| . 15. L2@0 = (e @ L2)
| . 16. s1 : traversal(the_graph)
| . 17. s2 : traversal(the_graph)
| . 18. ∀j:Vertices(the_graph).
| . |   ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
| . |   & ((inl(j) ∈ s2) ⟺ e-the_graph->*j & ¬L1@0-the_graph->*j)
| . 19. j : Vertices(the_graph)
| . 20. e-the_graph->*j ∨ L2-the_graph->*j
| . 21. ¬L1@0-the_graph->*j
| . ⊢  e-the_graph->*j  by Analyze -2
| . |
| . 1. the_graph : Graph
| . 2. L1 : Vertices(the_graph) List
| . 3. L2 : Vertices(the_graph) List
| . 4. s : traversal(the_graph)
| . 5. L1-the_graph->*L2
| . 6. df-traversal(the_graph;s)
| . 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
| . 8. ∀i:Vertices(the_graph).
| . |  L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
| . 9. ∀L1@0,L2:Vertices(the_graph) List.
| . |  L1 = (L1@0 @ L2)
| . |  ⇒
| . |  (∃s1,s2:traversal(the_graph).
| . |   s = (s2 @ s1)
| . |   & paren(Vertices(the_graph);s1)
| . |   & paren(Vertices(the_graph);s2)
| . |   & (∀j:Vertices(the_graph).
| . |      ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
| . |      & ((inl(j) ∈ s2)
| . |         ⟺
| . |         L2-the_graph->*j & ¬L1@0-the_graph->*j)))
| . 10. L1@0 : Vertices(the_graph) List
| . 11. L2@0 : Vertices(the_graph) List
| . 12. (L1 @ L2) = (L1@0 @ L2@0)
| . 13. e : Vertices(the_graph) List
| . 14. L1 = (L1@0 @ e)
| . 15. L2@0 = (e @ L2)
| . 16. s1 : traversal(the_graph)
| . 17. s2 : traversal(the_graph)
| . 18. ∀j:Vertices(the_graph).
| . |   ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
| . |   & ((inl(j) ∈ s2) ⟺ e-the_graph->*j & ¬L1@0-the_graph->*j)
| . 19. j : Vertices(the_graph)
| . 20. L2-the_graph->*j
```

```
| . 21. ¬L1@0-the_graph->*j
| . ⊢  e-the_graph->*j  by Assert L1-the_graph->*j
| . |
| . |\
| . | .....assertion .....
| . | 1. the_graph : Graph
| . | 2. L1 : Vertices(the_graph) List
| . | 3. L2 : Vertices(the_graph) List
| . | 4. s : traversal(the_graph)
| . | 5. L1-the_graph->*L2
| . | 6. df-traversal(the_graph;s)
| . | 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
| . | 8. ∀i:Vertices(the_graph).
| . |    L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
| . | 9. ∀L1@0,L2:Vertices(the_graph) List.
| . |    L1 = (L1@0 @ L2)
| . |    ⇒
| . |    (∃s1,s2:traversal(the_graph).
| . |     s = (s2 @ s1)
| . |     & paren(Vertices(the_graph);s1)
| . |     & paren(Vertices(the_graph);s2)
| . |     & (∀j:Vertices(the_graph).
| . |        ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
| . |        & ((inl(j) ∈ s2)
| . |           ⟺
| . |           L2-the_graph->*j & ¬L1@0-the_graph->*j)))
| . | 10. L1@0 : Vertices(the_graph) List
| . | 11. L2@0 : Vertices(the_graph) List
| . | 12. (L1 @ L2) = (L1@0 @ L2@0)
| . | 13. e : Vertices(the_graph) List
| . | 14. L1 = (L1@0 @ e)
| . | 15. L2@0 = (e @ L2)
| . | 16. s1 : traversal(the_graph)
| . | 17. s2 : traversal(the_graph)
| . | 18. ∀j:Vertices(the_graph).
| . |     ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
| . |     & ((inl(j) ∈ s2) ⟺ e-the_graph->*j & ¬L1@0-the_graph->*j)
| . | 19. j : Vertices(the_graph)
| . | 20. L2-the_graph->*j
| . | 21. ¬L1@0-the_graph->*j
| . | ⊢  L1-the_graph->*j
| . |   by ((RWO "list-list-connect-singleton<" 0)
| . |       THEN
| . |       (RWO "list-list-connect-singleton<" -2))
| . |      THEN
| . |      (Using [‘B’,L2]
| . |       (BackThru
| . |        Thm* For any graph
| . |             ∀A,B,C:V List. A-->*B ⇒ B-->*C ⇒ A-->*C
| . |        {list_DASH_list_DASH_connect_transitivity}))
| . |
| . .\
| . . 1. the_graph : Graph
| . . 2. L1 : Vertices(the_graph) List
| . . 3. L2 : Vertices(the_graph) List
```

```
| . . 4. s : traversal(the_graph)
| . . 5. L1-the_graph->*L2
| . . 6. df-traversal(the_graph;s)
| . . 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
| . . 8. ∀i:Vertices(the_graph).
| . .    L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
| . . 9. ∀L1@0,L2:Vertices(the_graph) List.
| . .    L1 = (L1@0 @ L2)
| . .    ⇒
| . .    (∃s1,s2:traversal(the_graph).
| . .     s = (s2 @ s1)
| . .     & paren(Vertices(the_graph);s1)
| . .     & paren(Vertices(the_graph);s2)
| . .     & (∀j:Vertices(the_graph).
| . .        ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
| . .        & ((inl(j) ∈ s2)
| . .           ⟺
| . .           L2-the_graph->*j & ¬L1@0-the_graph->*j)))
| . . 10. L1@0 : Vertices(the_graph) List
| . . 11. L2@0 : Vertices(the_graph) List
| . . 12. (L1 @ L2) = (L1@0 @ L2@0)
| . . 13. e : Vertices(the_graph) List
| . . 14. L1 = (L1@0 @ e)
| . . 15. L2@0 = (e @ L2)
| . . 16. s1 : traversal(the_graph)
| . . 17. s2 : traversal(the_graph)
| . . 18. ∀j:Vertices(the_graph).
| . .     ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
| . .     & ((inl(j) ∈ s2) ⟺ e-the_graph->*j & ¬L1@0-the_graph->*j)
| . . 19. j : Vertices(the_graph)
| . . 20. L2-the_graph->*j
| . . 21. ¬L1@0-the_graph->*j
| . . 22. L1-the_graph->*j
| . . ⊢  e-the_graph->*j
| . .   by (SubstFor L1 -1)
| . .      THEN
| . .      (RWO
| . .       Thm* For any graph
| . .            ∀A,B:V List, i:V. A @ B-->*i ⟺ A-->*i ∨ B-->*i
| . .       {list_DASH_connect_DASH_append}
| . .       -1)
| . .      THEN
| . .      (Analyze -1)
|
.\
. 1. the_graph : Graph
. 2. L1 : Vertices(the_graph) List
. 3. L2 : Vertices(the_graph) List
. 4. s : traversal(the_graph)
. 5. L1-the_graph->*L2
. 6. df-traversal(the_graph;s)
. 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
. 8. ∀i:Vertices(the_graph).
. |  L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
. 9. ∀L1@0,L2:Vertices(the_graph) List.
```

```
. |  L1 = (L1@0 @ L2)
. |  ⇒
. |  (∃s1,s2:traversal(the_graph).
. |   s = (s2 @ s1)
. |   & paren(Vertices(the_graph);s1) & paren(Vertices(the_graph);s2)
. |   & (∀j:Vertices(the_graph).
. |      ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
. |      & ((inl(j) ∈ s2) ⟺ L2-the_graph->*j & ¬L1@0-the_graph->*j)))
. 10. L1@0 : Vertices(the_graph) List
. 11. L2@0 : Vertices(the_graph) List
. 12. (L1 @ L2) = (L1@0 @ L2@0)
. 13. e : Vertices(the_graph) List
. 14. L1@0 = (L1 @ e)
. 15. L2 = (e @ L2@0)
. ⊢  ∃s1,s2:traversal(the_graph).
. |  s = (s2 @ s1)
. |  & paren(Vertices(the_graph);s1) & paren(Vertices(the_graph);s2)
. |  & (∀j:Vertices(the_graph).
. |     ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
. |     & ((inl(j) ∈ s2) ⟺ L2@0-the_graph->*j & ¬L1@0-the_graph->*j))
. | by (((HypSubst -2 0)
. |      THEN
. |      (RWO
. |       Thm* For any graph
. |            ∀A,B:V List, i:V. A @ B-->*i ⟺ A-->*i ∨ B-->*i
. |       {list_DASH_connect_DASH_append}
. |       0)
. |      THEN
. |      (InstHyp [L1;nil] 9))
. |     THENA
. |     (Auto
. |      THEN
. |      (RWO Thm* ∀l:T List. (l @ nil) ~ l    {append_nil_sq} 0)))
. |    THEN
. |    (RepeatFor 7 (ParallelOp -1))
. |
. |\
. | 1. the_graph : Graph
. | 2. L1 : Vertices(the_graph) List
. | 3. L2 : Vertices(the_graph) List
. | 4. s : traversal(the_graph)
. | 5. L1-the_graph->*L2
. | 6. df-traversal(the_graph;s)
. | 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
. | 8. ∀i:Vertices(the_graph).
. |    L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
. | 9. ∀L1@0,L2:Vertices(the_graph) List.
. |    L1 = (L1@0 @ L2)
. |    ⇒
. |    (∃s1,s2:traversal(the_graph).
. |     s = (s2 @ s1)
. |     & paren(Vertices(the_graph);s1)
. |     & paren(Vertices(the_graph);s2)
. |     & (∀j:Vertices(the_graph).
. |        ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
```

```
. |        & ((inl(j) ∈ s2)
. |            ⟺
. |            L2-the_graph->*j & ¬L1@0-the_graph->*j)))
. | 10. L1@0 : Vertices(the_graph) List
. | 11. L2@0 : Vertices(the_graph) List
. | 12. (L1 @ L2) = (L1@0 @ L2@0)
. | 13. e : Vertices(the_graph) List
. | 14. L1@0 = (L1 @ e)
. | 15. L2 = (e @ L2@0)
. | 16. s1 : traversal(the_graph)
. | 17. s2 : traversal(the_graph)
. | 18. ∀j:Vertices(the_graph).
. |     ((inl(j) ∈ s1) ⟺ L1-the_graph->*j)
. |     & ((inl(j) ∈ s2) ⟺ nil-the_graph->*j & ¬L1-the_graph->*j)
. | 19. j : Vertices(the_graph)
. | 20. (inl(j) ∈ s1)
. | 21. L1-the_graph->*j
. | ⊢  L1-the_graph->*j ∨ e-the_graph->*j  by OrLeft
. |
. |\
. | .....antecedent .....
. | 1. the_graph : Graph
. | 2. L1 : Vertices(the_graph) List
. | 3. L2 : Vertices(the_graph) List
. | 4. s : traversal(the_graph)
. | 5. L1-the_graph->*L2
. | 6. df-traversal(the_graph;s)
. | 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
. | 8. ∀i:Vertices(the_graph).
. | |  L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
. | 9. ∀L1@0,L2:Vertices(the_graph) List.
. | |  L1 = (L1@0 @ L2)
. | |  ⇒
. | |  (∃s1,s2:traversal(the_graph).
. | |   s = (s2 @ s1)
. | |   & paren(Vertices(the_graph);s1)
. | |   & paren(Vertices(the_graph);s2)
. | |   & (∀j:Vertices(the_graph).
. | |      ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
. | |      & ((inl(j) ∈ s2)
. | |         ⟺
. | |         L2-the_graph->*j & ¬L1@0-the_graph->*j)))
. | 10. L1@0 : Vertices(the_graph) List
. | 11. L2@0 : Vertices(the_graph) List
. | 12. (L1 @ L2) = (L1@0 @ L2@0)
. | 13. e : Vertices(the_graph) List
. | 14. L1@0 = (L1 @ e)
. | 15. L2 = (e @ L2@0)
. | 16. s1 : traversal(the_graph)
. | 17. s2 : traversal(the_graph)
. | 18. ∀j:Vertices(the_graph).
. | |   ((inl(j) ∈ s1) ⟺ L1-the_graph->*j)
. | |   & ((inl(j) ∈ s2) ⟺ nil-the_graph->*j & ¬L1-the_graph->*j)
. | 19. j : Vertices(the_graph)
. | 20. L1-the_graph->*j ∨ e-the_graph->*j
```

```
. | ⊢  L1-the_graph->*j  by Analyze -1
. | |
. | 1. the_graph : Graph
. | 2. L1 : Vertices(the_graph) List
. | 3. L2 : Vertices(the_graph) List
. | 4. s : traversal(the_graph)
. | 5. L1-the_graph->*L2
. | 6. df-traversal(the_graph;s)
. | 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
. | 8. ∀i:Vertices(the_graph).
. | |  L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
. | 9. ∀L1@0,L2:Vertices(the_graph) List.
. | |  L1 = (L1@0 @ L2)
. | |  ⇒
. | |  (∃s1,s2:traversal(the_graph).
. | |   s = (s2 @ s1)
. | |   & paren(Vertices(the_graph);s1)
. | |   & paren(Vertices(the_graph);s2)
. | |   & (∀j:Vertices(the_graph).
. | |      ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
. | |      & ((inl(j) ∈ s2)
. | |         ⟺
. | |         L2-the_graph->*j & ¬L1@0-the_graph->*j)))
. | 10. L1@0 : Vertices(the_graph) List
. | 11. L2@0 : Vertices(the_graph) List
. | 12. (L1 @ L2) = (L1@0 @ L2@0)
. | 13. e : Vertices(the_graph) List
. | 14. L1@0 = (L1 @ e)
. | 15. L2 = (e @ L2@0)
. | 16. s1 : traversal(the_graph)
. | 17. s2 : traversal(the_graph)
. | 18. ∀j:Vertices(the_graph).
. | |   ((inl(j) ∈ s1) ⟺ L1-the_graph->*j)
. | |   & ((inl(j) ∈ s2) ⟺ nil-the_graph->*j & ¬L1-the_graph->*j)
. | 19. j : Vertices(the_graph)
. | 20. e-the_graph->*j
. | ⊢  L1-the_graph->*j  by Assert L2-the_graph->*j
. | |
. | |\
. | | .....assertion .....
. | | 1. the_graph : Graph
. | | 2. L1 : Vertices(the_graph) List
. | | 3. L2 : Vertices(the_graph) List
. | | 4. s : traversal(the_graph)
. | | 5. L1-the_graph->*L2
. | | 6. df-traversal(the_graph;s)
. | | 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
. | | 8. ∀i:Vertices(the_graph).
. | |    L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
. | | 9. ∀L1@0,L2:Vertices(the_graph) List.
. | |    L1 = (L1@0 @ L2)
. | |    ⇒
. | |    (∃s1,s2:traversal(the_graph).
. | |     s = (s2 @ s1)
. | |     & paren(Vertices(the_graph);s1)
```

```
. | |     & paren(Vertices(the_graph);s2)
. | |     & (∀j:Vertices(the_graph).
. | |        ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
. | |        & ((inl(j) ∈ s2)
. | |           ⟺
. | |           L2-the_graph->*j & ¬L1@0-the_graph->*j)))
. | | 10. L1@0 : Vertices(the_graph) List
. | | 11. L2@0 : Vertices(the_graph) List
. | | 12. (L1 @ L2) = (L1@0 @ L2@0)
. | | 13. e : Vertices(the_graph) List
. | | 14. L1@0 = (L1 @ e)
. | | 15. L2 = (e @ L2@0)
. | | 16. s1 : traversal(the_graph)
. | | 17. s2 : traversal(the_graph)
. | | 18. ∀j:Vertices(the_graph).
. | |     ((inl(j) ∈ s1) ⟺ L1-the_graph->*j)
. | |     & ((inl(j) ∈ s2) ⟺ nil-the_graph->*j & ¬L1-the_graph->*j)
. | | 19. j : Vertices(the_graph)
. | | 20. e-the_graph->*j
. | | ⊢  L2-the_graph->*j
. | |   by ((SubstFor L2 0)
. | |       THEN
. | |       (RWO
. | |        Thm* For any graph
. | |             ∀A,B:V List, i:V. A @ B-->*i ⟺ A-->*i ∨ B-->*i
. | |        {list_DASH_connect_DASH_append}
. | |        0))
. | |      THEN
. | |      OrLeft
. | |
. | .\
. | . 1. the_graph : Graph
. | . 2. L1 : Vertices(the_graph) List
. | . 3. L2 : Vertices(the_graph) List
. | . 4. s : traversal(the_graph)
. | . 5. L1-the_graph->*L2
. | . 6. df-traversal(the_graph;s)
. | . 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
. | . 8. ∀i:Vertices(the_graph).
. | .    L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
. | . 9. ∀L1@0,L2:Vertices(the_graph) List.
. | .    L1 = (L1@0 @ L2)
. | .    ⇒
. | .    (∃s1,s2:traversal(the_graph).
. | .     s = (s2 @ s1)
. | .     & paren(Vertices(the_graph);s1)
. | .     & paren(Vertices(the_graph);s2)
. | .     & (∀j:Vertices(the_graph).
. | .        ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
. | .        & ((inl(j) ∈ s2)
. | .           ⟺
. | .           L2-the_graph->*j & ¬L1@0-the_graph->*j)))
. | . 10. L1@0 : Vertices(the_graph) List
. | . 11. L2@0 : Vertices(the_graph) List
. | . 12. (L1 @ L2) = (L1@0 @ L2@0)
```

```
. | . 13. e : Vertices(the_graph) List
. | . 14. L1@0 = (L1 @ e)
. | . 15. L2 = (e @ L2@0)
. | . 16. s1 : traversal(the_graph)
. | . 17. s2 : traversal(the_graph)
. | . 18. ∀j:Vertices(the_graph).
. | .      ((inl(j) ∈ s1) ⟺ L1-the_graph->*j)
. | .      & ((inl(j) ∈ s2) ⟺ nil-the_graph->*j & ¬L1-the_graph->*j)
. | . 19. j : Vertices(the_graph)
. | . 20. e-the_graph->*j
. | . 21. L2-the_graph->*j
. | . ⊢  L1-the_graph->*j
. | .   by ((RWO "list-list-connect-singleton<" -1)
. | .       THEN
. | .       (RWO "list-list-connect-singleton<" 0))
. | .      THEN
. | .      (Using [‘B’,L2]
. | .       (BackThru
. | .        Thm* For any graph
. | .             ∀A,B,C:V List. A-->*B ⇒ B-->*C ⇒ A-->*C
. | .        {list_DASH_list_DASH_connect_transitivity}))
. |
. |\
. | 1. the_graph : Graph
. | 2. L1 : Vertices(the_graph) List
. | 3. L2 : Vertices(the_graph) List
. | 4. s : traversal(the_graph)
. | 5. L1-the_graph->*L2
. | 6. df-traversal(the_graph;s)
. | 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
. | 8. ∀i:Vertices(the_graph).
. |    L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
. | 9. ∀L1@0,L2:Vertices(the_graph) List.
. |    L1 = (L1@0 @ L2)
. |    ⇒
. |    (∃s1,s2:traversal(the_graph).
. |     s = (s2 @ s1)
. |     & paren(Vertices(the_graph);s1)
. |     & paren(Vertices(the_graph);s2)
. |     & (∀j:Vertices(the_graph).
. |        ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
. |        & ((inl(j) ∈ s2)
. |           ⟺
. |           L2-the_graph->*j & ¬L1@0-the_graph->*j)))
. | 10. L1@0 : Vertices(the_graph) List
. | 11. L2@0 : Vertices(the_graph) List
. | 12. (L1 @ L2) = (L1@0 @ L2@0)
. | 13. e : Vertices(the_graph) List
. | 14. L1@0 = (L1 @ e)
. | 15. L2 = (e @ L2@0)
. | 16. s1 : traversal(the_graph)
. | 17. s2 : traversal(the_graph)
. | 18. ∀j:Vertices(the_graph).
. |     ((inl(j) ∈ s1) ⟺ L1-the_graph->*j)
. |     & ((inl(j) ∈ s2) ⟺ nil-the_graph->*j & ¬L1-the_graph->*j)
```

```
. | 19. j : Vertices(the_graph)
. | 20. (inl(j) ∈ s2)
. | 21. nil-the_graph->*j & ¬L1-the_graph->*j
. | ⊢  L2@0-the_graph->*j & ¬(L1-the_graph->*j ∨ e-the_graph->*j)
. |   by (Analyze -1) THEN (Analyze -2) THEN (Analyze -2)
. |      THEN
. |      (ObviousContradiction [-3])
. |
. .\
. . .....antecedent .....
. . 1. the_graph : Graph
. . 2. L1 : Vertices(the_graph) List
. . 3. L2 : Vertices(the_graph) List
. . 4. s : traversal(the_graph)
. . 5. L1-the_graph->*L2
. . 6. df-traversal(the_graph;s)
. . 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
. . 8. ∀i:Vertices(the_graph).
. . |  L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
. . 9. ∀L1@0,L2:Vertices(the_graph) List.
. . |  L1 = (L1@0 @ L2)
. . |  ⇒
. . |  (∃s1,s2:traversal(the_graph).
. . |   s = (s2 @ s1)
. . |   & paren(Vertices(the_graph);s1)
. . |   & paren(Vertices(the_graph);s2)
. . |   & (∀j:Vertices(the_graph).
. . |      ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
. . |      & ((inl(j) ∈ s2)
. . |         ⟺
. . |         L2-the_graph->*j & ¬L1@0-the_graph->*j)))
. . 10. L1@0 : Vertices(the_graph) List
. . 11. L2@0 : Vertices(the_graph) List
. . 12. (L1 @ L2) = (L1@0 @ L2@0)
. . 13. e : Vertices(the_graph) List
. . 14. L1@0 = (L1 @ e)
. . 15. L2 = (e @ L2@0)
. . 16. s1 : traversal(the_graph)
. . 17. s2 : traversal(the_graph)
. . 18. ∀j:Vertices(the_graph).
. . |   ((inl(j) ∈ s1) ⟺ L1-the_graph->*j)
. . |   & ((inl(j) ∈ s2) ⟺ nil-the_graph->*j & ¬L1-the_graph->*j)
. . 19. j : Vertices(the_graph)
. . 20. L2@0-the_graph->*j & ¬(L1-the_graph->*j ∨ e-the_graph->*j)
. . ⊢  nil-the_graph->*j & ¬L1-the_graph->*j
. . | by (Analyze -1) THEN (Analyze -1) THEN OrLeft
. . |
. . 1. the_graph : Graph
. . 2. L1 : Vertices(the_graph) List
. . 3. L2 : Vertices(the_graph) List
. . 4. s : traversal(the_graph)
. . 5. L1-the_graph->*L2
. . 6. df-traversal(the_graph;s)
. . 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
. . 8. ∀i:Vertices(the_graph).
```

```
. . |  L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
. . 9. ∀L1@0,L2:Vertices(the_graph) List.
. . |  L1 = (L1@0 @ L2)
. . |  ⇒
. . |  (∃s1,s2:traversal(the_graph).
. . |   s = (s2 @ s1)
. . |   & paren(Vertices(the_graph);s1)
. . |   & paren(Vertices(the_graph);s2)
. . |   & (∀j:Vertices(the_graph).
. . |      ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
. . |      & ((inl(j) ∈ s2)
. . |         ⟺
. . |         L2-the_graph->*j & ¬L1@0-the_graph->*j)))
. . 10. L1@0 : Vertices(the_graph) List
. . 11. L2@0 : Vertices(the_graph) List
. . 12. (L1 @ L2) = (L1@0 @ L2@0)
. . 13. e : Vertices(the_graph) List
. . 14. L1@0 = (L1 @ e)
. . 15. L2 = (e @ L2@0)
. . 16. s1 : traversal(the_graph)
. . 17. s2 : traversal(the_graph)
. . 18. ∀j:Vertices(the_graph).
. . |   ((inl(j) ∈ s1) ⟺ L1-the_graph->*j)
. . |   & ((inl(j) ∈ s2) ⟺ nil-the_graph->*j & ¬L1-the_graph->*j)
. . 19. j : Vertices(the_graph)
. . 20. L2@0-the_graph->*j
. . ⊢  L1-the_graph->*j  by Assert L2-the_graph->*j
. . |
. . |\
. . | .....assertion .....
. . | 1. the_graph : Graph
. . | 2. L1 : Vertices(the_graph) List
. . | 3. L2 : Vertices(the_graph) List
. . | 4. s : traversal(the_graph)
. . | 5. L1-the_graph->*L2
. . | 6. df-traversal(the_graph;s)
. . | 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
. . | 8. ∀i:Vertices(the_graph).
. . |    L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
. . | 9. ∀L1@0,L2:Vertices(the_graph) List.
. . |    L1 = (L1@0 @ L2)
. . |    ⇒
. . |    (∃s1,s2:traversal(the_graph).
. . |     s = (s2 @ s1)
. . |     & paren(Vertices(the_graph);s1)
. . |     & paren(Vertices(the_graph);s2)
. . |     & (∀j:Vertices(the_graph).
. . |        ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
. . |        & ((inl(j) ∈ s2)
. . |           ⟺
. . |           L2-the_graph->*j & ¬L1@0-the_graph->*j)))
. . | 10. L1@0 : Vertices(the_graph) List
. . | 11. L2@0 : Vertices(the_graph) List
. . | 12. (L1 @ L2) = (L1@0 @ L2@0)
. . | 13. e : Vertices(the_graph) List
```

```
. . | 14. L1@0 = (L1 @ e)
. . | 15. L2 = (e @ L2@0)
. . | 16. s1 : traversal(the_graph)
. . | 17. s2 : traversal(the_graph)
. . | 18. ∀j:Vertices(the_graph).
. . |     ((inl(j) ∈ s1) ⟺ L1-the_graph->*j)
. . |     & ((inl(j) ∈ s2) ⟺ nil-the_graph->*j & ¬L1-the_graph->*j)
. . | 19. j : Vertices(the_graph)
. . | 20. L2@0-the_graph->*j
. . | ⊢  L2-the_graph->*j
. . |   by ((SubstFor L2 0)
. . |       THEN
. . |       (RWO
. . |        Thm* For any graph
. . |             ∀A,B:V List, i:V. A @ B-->*i ⟺ A-->*i ∨ B-->*i
. . |        {list_DASH_connect_DASH_append}
. . |        0))
. . |      THEN
. . |      OrRight
. . |
. . .\
. . . 1. the_graph : Graph
. . . 2. L1 : Vertices(the_graph) List
. . . 3. L2 : Vertices(the_graph) List
. . . 4. s : traversal(the_graph)
. . . 5. L1-the_graph->*L2
. . . 6. df-traversal(the_graph;s)
. . . 7. ∀i:Vertices(the_graph). (inl(i) ∈ s) ⇒ L1-the_graph->*i
. . . 8. ∀i:Vertices(the_graph).
. . .    L1 @ L2-the_graph->*i ⇒ ¬non-trivial-loop(the_graph;i)
. . . 9. ∀L1@0,L2:Vertices(the_graph) List.
. . .    L1 = (L1@0 @ L2)
. . .    ⇒
. . .    (∃s1,s2:traversal(the_graph).
. . .     s = (s2 @ s1)
. . .     & paren(Vertices(the_graph);s1)
. . .     & paren(Vertices(the_graph);s2)
. . .     & (∀j:Vertices(the_graph).
. . .        ((inl(j) ∈ s1) ⟺ L1@0-the_graph->*j)
. . .        & ((inl(j) ∈ s2)
. . .           ⟺
. . .           L2-the_graph->*j & ¬L1@0-the_graph->*j)))
. . . 10. L1@0 : Vertices(the_graph) List
. . . 11. L2@0 : Vertices(the_graph) List
. . . 12. (L1 @ L2) = (L1@0 @ L2@0)
. . . 13. e : Vertices(the_graph) List
. . . 14. L1@0 = (L1 @ e)
. . . 15. L2 = (e @ L2@0)
. . . 16. s1 : traversal(the_graph)
. . . 17. s2 : traversal(the_graph)
. . . 18. ∀j:Vertices(the_graph).
. . .     ((inl(j) ∈ s1) ⟺ L1-the_graph->*j)
. . .     & ((inl(j) ∈ s2) ⟺ nil-the_graph->*j & ¬L1-the_graph->*j)
. . . 19. j : Vertices(the_graph)
. . . 20. L2@0-the_graph->*j
```

```
. . . 21. L2-the_graph->*j
. . . ⊢  L1-the_graph->*j
. . .   by ((RWO "list-list-connect-singleton<" -1)
. . .       THEN
. . .       (RWO "list-list-connect-singleton<" 0))
. . .      THEN
. . .      (Using [‘B’,L2]
. . .       (BackThru
. . .        Thm* For any graph
. . .             ∀A,B,C:V List. A-->*B ⇒ B-->*C ⇒ A-->*C
. . .        {list_DASH_list_DASH_connect_transitivity}))
END
```