```
spanner-root-unique  THM

⊢  ∀T:(Id→𝔹), to,from:(|T|→(IdLnk List)), f:(Edge(T)→𝔹).
|  bi-tree(T;to;from)
|  ⇒
|  spanner(f;T;to;from)
|  ⇒
|  (∀i,j:|T|.
|   spanner-root(f;T;to;from;i)
|   ⇒
|   spanner-root(f;T;to;from;j) ⇒ i = j)
| by RepeatFor 5 (Analyze 0) THEN DupHyp -1 THEN Analyze -1
|
1. T : Id→𝔹
2. to : |T|→(IdLnk List)
3. from : |T|→(IdLnk List)
4. f : Edge(T)→𝔹
5. bi-tree(T;to;from)
6. bi-graph(T;to;from)
7. ∀i,j:|T|.
|  ∃p:Edge(T) List.
|  lconnects(p;i;j) & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
9. |T|
10. spanner(f;T;to;from)
11. i : |T|
12. j : |T|
13. spanner-root(f;T;to;from;i)
14. spanner-root(f;T;to;from;j)
⊢  i = j
| by Assert
|    (∀p:Edge(T) List.
|     0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l)))
|
|\
| .....assertion .....
| 1. T : Id→𝔹
| 2. to : |T|→(IdLnk List)
| 3. from : |T|→(IdLnk List)
| 4. f : Edge(T)→𝔹
| 5. bi-tree(T;to;from)
| 6. bi-graph(T;to;from)
| 7. ∀i,j:|T|.
| |  ∃p:Edge(T) List.
| |  lconnects(p;i;j) & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| 9. |T|
| 10. spanner(f;T;to;from)
| 11. i : |T|
| 12. j : |T|
| 13. spanner-root(f;T;to;from;i)
| 14. spanner-root(f;T;to;from;j)
| ⊢  ∀p:Edge(T) List. 0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l))
| | by InductionOnList THEN Reduce 0 THEN MaAuto THEN Reduce 0
| |    THEN
```

```
| |    RWO
| |    Thm* ∀P:(T→Prop), x:T, L:T List.
| |         (∀y∈[x / L].P(y)) ⟺ P(x) & (∀y∈L.P(y))
| |    {l_all_cons}
| |    0
| |    THEN
| |    MaAuto
| |    THEN
| |    DVar ‘v’
| |    THEN
| |    All Reduce
| |    THEN
| |    Try (Unfold ‘last‘ -1 THEN Reduce -1 THEN Trivial)
| |    THEN
| |    Try (BackThru Thm* ∀P:(T→Prop). (∀x∈nil.P(x))    {l_all_nil})
| |    THEN
| |    RWO
| |    Thm* ∀l:IdLnk, p:IdLnk List.
| |         lpath([l / p])
| |         ⟺
| |         lpath(p)
| |         & (¬||p|| = 0 ∈ ℤ
| |            ⇒
| |            destination(l) = source(hd(p)) & ¬hd(p) = lnk-inv(l))
| |    {lpath_cons}
| |    -2
| |    THEN
| |    MaAuto
| |    THEN
| |    RWO
| |    Thm* ∀L:T List, x:T. ¬null(L) ⇒ last([x / L]) = last(L)
| |    {last_cons}
| |    -1
| |    THEN
| |    Reduce 0
| |
| |\
| | 1. T : Id→𝔹
| | 2. to : |T|→(IdLnk List)
| | 3. from : |T|→(IdLnk List)
| | 4. f : Edge(T)→𝔹
| | 5. bi-tree(T;to;from)
| | 6. bi-graph(T;to;from)
| | 7. ∀i,j:|T|.
| | |  ∃p:Edge(T) List.
| | |  lconnects(p;i;j) & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | 9. |T|
| | 10. spanner(f;T;to;from)
| | 11. i : |T|
| | 12. j : |T|
| | 13. spanner-root(f;T;to;from;i)
| | 14. spanner-root(f;T;to;from;j)
| | 15. Edge(T) List
| | 16. u : Edge(T)
```

```
| | 17. u1 : Edge(T)
| | 18. v1 : Edge(T) List
| | 19. 0<||v1||+1
| | |   ⇒
| | |   lpath([u1 / v1]) ⇒ f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | 20. 0<||v1||+1+1
| | 21. lpath([u1 / v1])
| | 22. ¬||[u1 / v1]|| = 0
| | |   ⇒
| | |   destination(u) = source(hd([u1 / v1]))
| | |   & ¬hd([u1 / v1]) = lnk-inv(u)
| | 23. f(last([u1 / v1]))
| | ⊢  f(u)
| | | by Reduce -2 THEN Analyze -2
| | |    THEN
| | |    AssertBY (∀l∈[u1 / v1].f(l)) BackThruSomeHyp
| | |
| | 1. T : Id→𝔹
| | 2. to : |T|→(IdLnk List)
| | 3. from : |T|→(IdLnk List)
| | 4. f : Edge(T)→𝔹
| | 5. bi-tree(T;to;from)
| | 6. bi-graph(T;to;from)
| | 7. ∀i,j:|T|.
| | |  ∃p:Edge(T) List.
| | |  lconnects(p;i;j) & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | 9. |T|
| | 10. spanner(f;T;to;from)
| | 11. i : |T|
| | 12. j : |T|
| | 13. spanner-root(f;T;to;from;i)
| | 14. spanner-root(f;T;to;from;j)
| | 15. Edge(T) List
| | 16. u : Edge(T)
| | 17. u1 : Edge(T)
| | 18. v1 : Edge(T) List
| | 19. 0<||v1||+1
| | |   ⇒
| | |   lpath([u1 / v1]) ⇒ f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | 20. 0<||v1||+1+1
| | 21. lpath([u1 / v1])
| | 22. f(last([u1 / v1]))
| | 23. destination(u) = source(u1)
| | 24. ¬u1 = lnk-inv(u)
| | 25. (∀l∈[u1 / v1].f(l))
| | ⊢  f(u)
| | | by RWO
| | |    Thm* ∀P:(T→Prop), x:T, L:T List.
| | |          (∀y∈[x / L].P(y)) ⟺ P(x) & (∀y∈L.P(y))
| | |    {l_all_cons}
| | |    -1
| | |
| | 1. T : Id→𝔹
| | 2. to : |T|→(IdLnk List)
```

```
| | 3. from : |T|→(IdLnk List)
| | 4. f : Edge(T)→𝔹
| | 5. bi-tree(T;to;from)
| | 6. bi-graph(T;to;from)
| | 7. ∀i,j:|T|.
| | |  ∃p:Edge(T) List.
| | |  lconnects(p;i;j) & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | 9. |T|
| | 10. spanner(f;T;to;from)
| | 11. i : |T|
| | 12. j : |T|
| | 13. spanner-root(f;T;to;from;i)
| | 14. spanner-root(f;T;to;from;j)
| | 15. Edge(T) List
| | 16. u : Edge(T)
| | 17. u1 : Edge(T)
| | 18. v1 : Edge(T) List
| | 19. 0<||v1||+1
| | |   ⇒
| | |   lpath([u1 / v1]) ⇒ f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | 20. 0<||v1||+1+1
| | 21. lpath([u1 / v1])
| | 22. f(last([u1 / v1]))
| | 23. destination(u) = source(u1)
| | 24. ¬u1 = lnk-inv(u)
| | 25. f(u1)
| | 26. (∀l∈v1.f(l))
| | ⊢  f(u)
| | | by DupHyp 10 THEN Unfold ‘spanner‘ -1 THEN Analyze -1
| | |    THEN
| | |    InstHyp [u1] -2
| | |    THEN
| | |    InstHyp [source(u1);u;inverse(u1)] -2
| | |
| | |\
| | | .....wf .....
| | | 1. T : Id→𝔹
| | | 2. to : |T|→(IdLnk List)
| | | 3. from : |T|→(IdLnk List)
| | | 4. f : Edge(T)→𝔹
| | | 5. bi-tree(T;to;from)
| | | 6. bi-graph(T;to;from)
| | | 7. ∀i,j:|T|.
| | |    ∃p:Edge(T) List.
| | |    lconnects(p;i;j)
| | |    & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | | 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | | 9. |T|
| | | 10. spanner(f;T;to;from)
| | | 11. i : |T|
| | | 12. j : |T|
| | | 13. spanner-root(f;T;to;from;i)
| | | 14. spanner-root(f;T;to;from;j)
| | | 15. Edge(T) List
```

```
| | | 16. u : Edge(T)
| | | 17. u1 : Edge(T)
| | | 18. v1 : Edge(T) List
| | | 19. 0<||v1||+1
| | |     ⇒
| | |     lpath([u1 / v1])
| | |     ⇒
| | |     f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | | 20. 0<||v1||+1+1
| | | 21. lpath([u1 / v1])
| | | 22. f(last([u1 / v1]))
| | | 23. destination(u) = source(u1)
| | | 24. ¬u1 = lnk-inv(u)
| | | 25. f(u1)
| | | 26. (∀l∈v1.f(l))
| | | 27. ∀l:Edge(T). f(l) = ¬₂f(inverse(l))
| | | 28. ∀i:|T|, l1,l2:Edge(T).
| | |     (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | | 29. f(u1) = ¬₂f(inverse(u1))
| | | ⊢  source(u1) ∈ |T|
| | |   by BackThru
| | |      Thm* ∀T:(Id→𝔹), to,from:(|T|→(IdLnk List)), u:Edge(T).
| | |           bi-graph(T;to;from) ⇒ source(u) ∈ |T|
| | |      {src_DASH_edge}
| | |
| | |\
| | | .....antecedent .....
| | | 1. T : Id→𝔹
| | | 2. to : |T|→(IdLnk List)
| | | 3. from : |T|→(IdLnk List)
| | | 4. f : Edge(T)→𝔹
| | | 5. bi-tree(T;to;from)
| | | 6. bi-graph(T;to;from)
| | | 7. ∀i,j:|T|.
| | | |  ∃p:Edge(T) List.
| | | |  lconnects(p;i;j)
| | | |  & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | | 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | | 9. |T|
| | | 10. spanner(f;T;to;from)
| | | 11. i : |T|
| | | 12. j : |T|
| | | 13. spanner-root(f;T;to;from;i)
| | | 14. spanner-root(f;T;to;from;j)
| | | 15. Edge(T) List
| | | 16. u : Edge(T)
| | | 17. u1 : Edge(T)
| | | 18. v1 : Edge(T) List
| | | 19. 0<||v1||+1
| | | |   ⇒
| | | |   lpath([u1 / v1])
| | | |   ⇒
| | | |   f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | | 20. 0<||v1||+1+1
| | | 21. lpath([u1 / v1])
```

```
| | | 22. f(last([u1 / v1]))
| | | 23. destination(u) = source(u1)
| | | 24. ¬u1 = lnk-inv(u)
| | | 25. f(u1)
| | | 26. (∀l∈v1.f(l))
| | | 27. ∀l:Edge(T). f(l) = ¬₂f(inverse(l))
| | | 28. ∀i:|T|, l1,l2:Edge(T).
| | | |   (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | | 29. f(u1) = ¬₂f(inverse(u1))
| | | ⊢  (u ∈ to(source(u1)))
| | | | by RevHypSubst' -7 0 THEN DupHyp 6 THEN Unfold 'bi-graph' -1
| | | |    THEN
| | | |    ExRepD
| | | |
| | | 1. T : Id→𝔹
| | | 2. to : |T|→(IdLnk List)
| | | 3. from : |T|→(IdLnk List)
| | | 4. f : Edge(T)→𝔹
| | | 5. bi-tree(T;to;from)
| | | 6. bi-graph(T;to;from)
| | | 7. ∀i,j:|T|.
| | | |  ∃p:Edge(T) List.
| | | |  lconnects(p;i;j)
| | | |  & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | | 8. L : |T| List
| | | 9. ∀i:|T|. (i ∈ L)
| | | 10. |T|
| | | 11. spanner(f;T;to;from)
| | | 12. i : |T|
| | | 13. j : |T|
| | | 14. spanner-root(f;T;to;from;i)
| | | 15. spanner-root(f;T;to;from;j)
| | | 16. Edge(T) List
| | | 17. u : Edge(T)
| | | 18. u1 : Edge(T)
| | | 19. v1 : Edge(T) List
| | | 20. 0<||v1||+1
| | | |   ⇒
| | | |   lpath([u1 / v1])
| | | |   ⇒
| | | |   f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | | 21. 0<||v1||+1+1
| | | 22. lpath([u1 / v1])
| | | 23. f(last([u1 / v1]))
| | | 24. destination(u) = source(u1)
| | | 25. ¬u1 = lnk-inv(u)
| | | 26. f(u1)
| | | 27. (∀l∈v1.f(l))
| | | 28. ∀l:Edge(T). f(l) = ¬₂f(inverse(l))
| | | 29. ∀i:|T|, l1,l2:Edge(T).
| | | |   (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | | 30. f(u1) = ¬₂f(inverse(u1))
| | | 31. ∀i:|T|.
| | | |   (∀l∈to(i).destination(l) = i
| | | |   & T(source(l))
```

```
| | | |   & (l ∈ from(source(l)))
| | | |   & (lnk-inv(l) ∈ from(i)))
| | | |   & (∀l∈from(i).source(l) = i
| | | |     & T(destination(l))
| | | |     & (l ∈ to(destination(l)))
| | | |     & (lnk-inv(l) ∈ to(i)))
| | | ⊢  (u ∈ to(destination(u)))
| | | | by BackThru
| | | |    Thm* ∀G:(Id→𝔹), to,from:(|G|→(IdLnk List)), e:Edge(G),
| | | |          i:|G|.
| | | |         bi-graph(G;to;from)
| | | |         ⇒
| | | |         ((e ∈ to(i)) ⟺ destination(e) = i)
| | | |    {edge_DASH_to}
| | | |
| | | .....wf .....
| | | 1. T : Id→𝔹
| | | 2. to : |T|→(IdLnk List)
| | | 3. from : |T|→(IdLnk List)
| | | 4. f : Edge(T)→𝔹
| | | 5. bi-tree(T;to;from)
| | | 6. bi-graph(T;to;from)
| | | 7. ∀i,j:|T|.
| | |    ∃p:Edge(T) List.
| | |    lconnects(p;i;j)
| | |    & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | | 8. L : |T| List
| | | 9. ∀i:|T|. (i ∈ L)
| | | 10. |T|
| | | 11. spanner(f;T;to;from)
| | | 12. i : |T|
| | | 13. j : |T|
| | | 14. spanner-root(f;T;to;from;i)
| | | 15. spanner-root(f;T;to;from;j)
| | | 16. Edge(T) List
| | | 17. u : Edge(T)
| | | 18. u1 : Edge(T)
| | | 19. v1 : Edge(T) List
| | | 20. 0<||v1||+1
| | |     ⇒
| | |     lpath([u1 / v1])
| | |     ⇒
| | |     f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | | 21. 0<||v1||+1+1
| | | 22. lpath([u1 / v1])
| | | 23. f(last([u1 / v1]))
| | | 24. destination(u) = source(u1)
| | | 25. ¬u1 = lnk-inv(u)
| | | 26. f(u1)
| | | 27. (∀l∈v1.f(l))
| | | 28. ∀l:Edge(T). f(l) = ¬₂f(inverse(l))
| | | 29. ∀i:|T|, l1,l2:Edge(T).
| | |     (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | | 30. f(u1) = ¬₂f(inverse(u1))
| | | 31. ∀i:|T|.
```

```
| | |     (∀l∈to(i).destination(l) = i
| | |     & T(source(l))
| | |     & (l ∈ from(source(l)))
| | |     & (lnk-inv(l) ∈ from(i)))
| | |     & (∀l∈from(i).source(l) = i
| | |       & T(destination(l))
| | |       & (l ∈ to(destination(l)))
| | |       & (lnk-inv(l) ∈ to(i)))
| | | ⊢  destination(u) ∈ |T|
| | |   by BackThru
| | |      Thm* ∀T:(Id→𝔹), to,from:(|T|→(IdLnk List)), u:Edge(T).
| | |           bi-graph(T;to;from) ⇒ destination(u) ∈ |T|
| | |      {dst_DASH_edge}
| | |
| | |\
| | | .....antecedent .....
| | | 1. T : Id→𝔹
| | | 2. to : |T|→(IdLnk List)
| | | 3. from : |T|→(IdLnk List)
| | | 4. f : Edge(T)→𝔹
| | | 5. bi-tree(T;to;from)
| | | 6. bi-graph(T;to;from)
| | | 7. ∀i,j:|T|.
| | | |  ∃p:Edge(T) List.
| | | |  lconnects(p;i;j)
| | | |  & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | | 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | | 9. |T|
| | | 10. spanner(f;T;to;from)
| | | 11. i : |T|
| | | 12. j : |T|
| | | 13. spanner-root(f;T;to;from;i)
| | | 14. spanner-root(f;T;to;from;j)
| | | 15. Edge(T) List
| | | 16. u : Edge(T)
| | | 17. u1 : Edge(T)
| | | 18. v1 : Edge(T) List
| | | 19. 0<||v1||+1
| | | |   ⇒
| | | |   lpath([u1 / v1])
| | | |   ⇒
| | | |   f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | | 20. 0<||v1||+1+1
| | | 21. lpath([u1 / v1])
| | | 22. f(last([u1 / v1]))
| | | 23. destination(u) = source(u1)
| | | 24. ¬u1 = lnk-inv(u)
| | | 25. f(u1)
| | | 26. (∀l∈v1.f(l))
| | | 27. ∀l:Edge(T). f(l) = ¬₂f(inverse(l))
| | | 28. ∀i:|T|, l1,l2:Edge(T).
| | | |   (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | | 29. f(u1) = ¬₂f(inverse(u1))
| | | ⊢  (inverse(u1) ∈ to(source(u1)))
| | | | by Subst’ (source(u1) = destination(inverse(u1))) 0
```

```
| | | |
| | | |\
| | | | 1. T : Id→𝔹
| | | | 2. to : |T|→(IdLnk List)
| | | | 3. from : |T|→(IdLnk List)
| | | | 4. f : Edge(T)→𝔹
| | | | 5. bi-tree(T;to;from)
| | | | 6. bi-graph(T;to;from)
| | | | 7. ∀i,j:|T|.
| | | |    ∃p:Edge(T) List.
| | | |    lconnects(p;i;j)
| | | |    & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | | | 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | | | 9. |T|
| | | | 10. spanner(f;T;to;from)
| | | | 11. i : |T|
| | | | 12. j : |T|
| | | | 13. spanner-root(f;T;to;from;i)
| | | | 14. spanner-root(f;T;to;from;j)
| | | | 15. Edge(T) List
| | | | 16. u : Edge(T)
| | | | 17. u1 : Edge(T)
| | | | 18. v1 : Edge(T) List
| | | | 19. 0<||v1||+1
| | | |     ⇒
| | | |     lpath([u1 / v1])
| | | |     ⇒
| | | |     f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | | | 20. 0<||v1||+1+1
| | | | 21. lpath([u1 / v1])
| | | | 22. f(last([u1 / v1]))
| | | | 23. destination(u) = source(u1)
| | | | 24. ¬u1 = lnk-inv(u)
| | | | 25. f(u1)
| | | | 26. (∀l∈v1.f(l))
| | | | 27. ∀l:Edge(T). f(l) = ¬₂f(inverse(l))
| | | | 28. ∀i:|T|, l1,l2:Edge(T).
| | | |     (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | | | 29. f(u1) = ¬₂f(inverse(u1))
| | | | ⊢  inverse(u1) ∈ IdLnk
| | | |   by DoSubsume THEN Analyze 0 THEN MaAuto
| | | |
| | | |\
| | | | 1. T : Id→𝔹
| | | | 2. to : |T|→(IdLnk List)
| | | | 3. from : |T|→(IdLnk List)
| | | | 4. f : Edge(T)→𝔹
| | | | 5. bi-tree(T;to;from)
| | | | 6. bi-graph(T;to;from)
| | | | 7. ∀i,j:|T|.
| | | |    ∃p:Edge(T) List.
| | | |    lconnects(p;i;j)
| | | |    & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | | | 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | | | 9. |T|
```

```
| | | | 10. spanner(f;T;to;from)
| | | | 11. i : |T|
| | | | 12. j : |T|
| | | | 13. spanner-root(f;T;to;from;i)
| | | | 14. spanner-root(f;T;to;from;j)
| | | | 15. Edge(T) List
| | | | 16. u : Edge(T)
| | | | 17. u1 : Edge(T)
| | | | 18. v1 : Edge(T) List
| | | | 19. 0<||v1||+1
| | | |     ⇒
| | | |     lpath([u1 / v1])
| | | |     ⇒
| | | |     f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | | | 20. 0<||v1||+1+1
| | | | 21. lpath([u1 / v1])
| | | | 22. f(last([u1 / v1]))
| | | | 23. destination(u) = source(u1)
| | | | 24. ¬u1 = lnk-inv(u)
| | | | 25. f(u1)
| | | | 26. (∀l∈v1.f(l))
| | | | 27. ∀l:Edge(T). f(l) = ¬₂f(inverse(l))
| | | | 28. ∀i:|T|, l1,l2:Edge(T).
| | | |     (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | | | 29. f(u1) = ¬₂f(inverse(u1))
| | | | ⊢  source(u1) = destination(inverse(u1))
| | | |   by Unfold ‘bi-graph-inv‘ 0
| | | |      THEN
| | | |      RWO
| | | |      Thm* ∀l:IdLnk. destination(lnk-inv(l)) = source(l)
| | | |      {ldst_DASH_inv}
| | | |      0
| | | |      THEN
| | | |      MaAuto
| | | |
| | | .\
| | | . 1. T : Id→𝔹
| | | . 2. to : |T|→(IdLnk List)
| | | . 3. from : |T|→(IdLnk List)
| | | . 4. f : Edge(T)→𝔹
| | | . 5. bi-tree(T;to;from)
| | | . 6. bi-graph(T;to;from)
| | | . 7. ∀i,j:|T|.
| | | . |  ∃p:Edge(T) List.
| | | . |  lconnects(p;i;j)
| | | . |  & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | | . 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | | . 9. |T|
| | | . 10. spanner(f;T;to;from)
| | | . 11. i : |T|
| | | . 12. j : |T|
| | | . 13. spanner-root(f;T;to;from;i)
| | | . 14. spanner-root(f;T;to;from;j)
| | | . 15. Edge(T) List
| | | . 16. u : Edge(T)
```

```
| | | . 17. u1 : Edge(T)
| | | . 18. v1 : Edge(T) List
| | | . 19. 0<||v1||+1
| | | . |   ⇒
| | | . |   lpath([u1 / v1])
| | | . |   ⇒
| | | . |   f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | | . 20. 0<||v1||+1+1
| | | . 21. lpath([u1 / v1])
| | | . 22. f(last([u1 / v1]))
| | | . 23. destination(u) = source(u1)
| | | . 24. ¬u1 = lnk-inv(u)
| | | . 25. f(u1)
| | | . 26. (∀l∈v1.f(l))
| | | . 27. ∀l:Edge(T). f(l) = ¬₂f(inverse(l))
| | | . 28. ∀i:|T|, l1,l2:Edge(T).
| | | . |   (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | | . 29. f(u1) = ¬₂f(inverse(u1))
| | | . ⊢  (inverse(u1) ∈ to(destination(inverse(u1))))
| | | . | by BackThru
| | | . |    Thm* ∀G:(Id→𝔹), to,from:(|G|→(IdLnk List)), e:Edge(G),
| | | . |          i:|G|.
| | | . |         bi-graph(G;to;from)
| | | . |         ⇒
| | | . |         ((e ∈ to(i)) ⟺ destination(e) = i)
| | | . |    {edge_DASH_to}
| | | . |
| | | . |\
| | | . | .....wf .....
| | | . | 1. T : Id→𝔹
| | | . | 2. to : |T|→(IdLnk List)
| | | . | 3. from : |T|→(IdLnk List)
| | | . | 4. f : Edge(T)→𝔹
| | | . | 5. bi-tree(T;to;from)
| | | . | 6. bi-graph(T;to;from)
| | | . | 7. ∀i,j:|T|.
| | | . |    ∃p:Edge(T) List.
| | | . |    lconnects(p;i;j)
| | | . |    & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | | . | 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | | . | 9. |T|
| | | . | 10. spanner(f;T;to;from)
| | | . | 11. i : |T|
| | | . | 12. j : |T|
| | | . | 13. spanner-root(f;T;to;from;i)
| | | . | 14. spanner-root(f;T;to;from;j)
| | | . | 15. Edge(T) List
| | | . | 16. u : Edge(T)
| | | . | 17. u1 : Edge(T)
| | | . | 18. v1 : Edge(T) List
| | | . | 19. 0<||v1||+1
| | | . |     ⇒
| | | . |     lpath([u1 / v1])
| | | . |     ⇒
| | | . |     f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
```

```
| | | . | 20. 0<||v1||+1+1
| | | . | 21. lpath([u1 / v1])
| | | . | 22. f(last([u1 / v1]))
| | | . | 23. destination(u) = source(u1)
| | | . | 24. ¬u1 = lnk-inv(u)
| | | . | 25. f(u1)
| | | . | 26. (∀l∈v1.f(l))
| | | . | 27. ∀l:Edge(T). f(l) = ¬₂f(inverse(l))
| | | . | 28. ∀i:|T|, l1,l2:Edge(T).
| | | . |     (l1 ∈ to(i))
| | | . |     ⇒
| | | . |     (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | | . | 29. f(u1) = ¬₂f(inverse(u1))
| | | . | ⊢  destination(inverse(u1)) ∈ |T|
| | | . |   by BackThru
| | | . |      Thm* ∀T:(Id→𝔹), to,from:(|T|→(IdLnk List)),
| | | . |            u:Edge(T).
| | | . |           bi-graph(T;to;from) ⇒ destination(u) ∈ |T|
| | | . |      {dst_DASH_edge}
| | | . |
| | | . .\
| | | . . .....subterm .....
| | | . . 1. T : Id→𝔹
| | | . . 2. to : |T|→(IdLnk List)
| | | . . 3. from : |T|→(IdLnk List)
| | | . . 4. f : Edge(T)→𝔹
| | | . . 5. bi-tree(T;to;from)
| | | . . 6. bi-graph(T;to;from)
| | | . . 7. ∀i,j:|T|.
| | | . .    ∃p:Edge(T) List.
| | | . .    lconnects(p;i;j)
| | | . .    & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | | . . 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | | . . 9. |T|
| | | . . 10. spanner(f;T;to;from)
| | | . . 11. i : |T|
| | | . . 12. j : |T|
| | | . . 13. spanner-root(f;T;to;from;i)
| | | . . 14. spanner-root(f;T;to;from;j)
| | | . . 15. Edge(T) List
| | | . . 16. u : Edge(T)
| | | . . 17. u1 : Edge(T)
| | | . . 18. v1 : Edge(T) List
| | | . . 19. 0<||v1||+1
| | | . .     ⇒
| | | . .     lpath([u1 / v1])
| | | . .     ⇒
| | | . .     f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | | . . 20. 0<||v1||+1+1
| | | . . 21. lpath([u1 / v1])
| | | . . 22. f(last([u1 / v1]))
| | | . . 23. destination(u) = source(u1)
| | | . . 24. ¬u1 = lnk-inv(u)
| | | . . 25. f(u1)
| | | . . 26. (∀l∈v1.f(l))
```

```
| | | . . 27. ∀l:Edge(T). f(l) = ¬2f(inverse(l))
| | | . . 28. ∀i:|T|, l1,l2:Edge(T).
| | | . .     (l1 ∈ to(i))
| | | . .     ⇒
| | | . .     (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | | . . 29. f(u1) = ¬2f(inverse(u1))
| | | . . ⊢  inverse(u1) ∈ IdLnk
| | | . .   by DoSubsume THEN Analyze 0 THEN MaAuto
| | |
| | |\
| | | .....antecedent .....
| | | 1. T : Id→𝔹
| | | 2. to : |T|→(IdLnk List)
| | | 3. from : |T|→(IdLnk List)
| | | 4. f : Edge(T)→𝔹
| | | 5. bi-tree(T;to;from)
| | | 6. bi-graph(T;to;from)
| | | 7. ∀i,j:|T|.
| | | |  ∃p:Edge(T) List.
| | | |  lconnects(p;i;j)
| | | |  & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | | 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | | 9. |T|
| | | 10. spanner(f;T;to;from)
| | | 11. i : |T|
| | | 12. j : |T|
| | | 13. spanner-root(f;T;to;from;i)
| | | 14. spanner-root(f;T;to;from;j)
| | | 15. Edge(T) List
| | | 16. u : Edge(T)
| | | 17. u1 : Edge(T)
| | | 18. v1 : Edge(T) List
| | | 19. 0<||v1||+1
| | | |   ⇒
| | | |   lpath([u1 / v1])
| | | |   ⇒
| | | |   f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | | 20. 0<||v1||+1+1
| | | 21. lpath([u1 / v1])
| | | 22. f(last([u1 / v1]))
| | | 23. destination(u) = source(u1)
| | | 24. ¬u1 = lnk-inv(u)
| | | 25. f(u1)
| | | 26. (∀l∈v1.f(l))
| | | 27. ∀l:Edge(T). f(l) = ¬2f(inverse(l))
| | | 28. ∀i:|T|, l1,l2:Edge(T).
| | | |   (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | | 29. f(u1) = ¬2f(inverse(u1))
| | | ⊢  ¬u = inverse(u1)
| | | | by Unfold `bi-graph-inv` 0 THEN Analyze 0 THENA MaAuto
| | | |
| | | 1. T : Id→𝔹
| | | 2. to : |T|→(IdLnk List)
| | | 3. from : |T|→(IdLnk List)
| | | 4. f : Edge(T)→𝔹
```

```
| | | 5. bi-tree(T;to;from)
| | | 6. bi-graph(T;to;from)
| | | 7. ∀i,j:|T|.
| | | |  ∃p:Edge(T) List.
| | | |  lconnects(p;i;j)
| | | |  & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | | 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | | 9. |T|
| | | 10. spanner(f;T;to;from)
| | | 11. i : |T|
| | | 12. j : |T|
| | | 13. spanner-root(f;T;to;from;i)
| | | 14. spanner-root(f;T;to;from;j)
| | | 15. Edge(T) List
| | | 16. u : Edge(T)
| | | 17. u1 : Edge(T)
| | | 18. v1 : Edge(T) List
| | | 19. 0<||v1||+1
| | | |   ⇒
| | | |   lpath([u1 / v1])
| | | |   ⇒
| | | |   f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | | 20. 0<||v1||+1+1
| | | 21. lpath([u1 / v1])
| | | 22. f(last([u1 / v1]))
| | | 23. destination(u) = source(u1)
| | | 24. ¬u1 = lnk-inv(u)
| | | 25. f(u1)
| | | 26. (∀l∈v1.f(l))
| | | 27. ∀l:Edge(T). f(l) = ¬₂f(inverse(l))
| | | 28. ∀i:|T|, l1,l2:Edge(T).
| | | |   (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | | 29. f(u1) = ¬₂f(inverse(u1))
| | | 30. u = lnk-inv(u1)
| | | ⊢  False  by Analyze -7 THEN HypSubst' -1 0
| | | |
| | | 1. T : Id→𝔹
| | | 2. to : |T|→(IdLnk List)
| | | 3. from : |T|→(IdLnk List)
| | | 4. f : Edge(T)→𝔹
| | | 5. bi-tree(T;to;from)
| | | 6. bi-graph(T;to;from)
| | | 7. ∀i,j:|T|.
| | |    ∃p:Edge(T) List.
| | |    lconnects(p;i;j)
| | |    & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | | 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | | 9. |T|
| | | 10. spanner(f;T;to;from)
| | | 11. i : |T|
| | | 12. j : |T|
| | | 13. spanner-root(f;T;to;from;i)
| | | 14. spanner-root(f;T;to;from;j)
| | | 15. Edge(T) List
| | | 16. u : Edge(T)
```

```
| | | 17. u1 : Edge(T)
| | | 18. v1 : Edge(T) List
| | | 19. 0<||v1||+1
| | |     ⇒
| | |     lpath([u1 / v1])
| | |     ⇒
| | |     f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | | 20. 0<||v1||+1+1
| | | 21. lpath([u1 / v1])
| | | 22. f(last([u1 / v1]))
| | | 23. destination(u) = source(u1)
| | | 24. f(u1)
| | | 25. (∀l∈v1.f(l))
| | | 26. ∀l:Edge(T). f(l) = ¬2f(inverse(l))
| | | 27. ∀i:|T|, l1,l2:Edge(T).
| | |     (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | | 28. f(u1) = ¬2f(inverse(u1))
| | | 29. u = lnk-inv(u1)
| | | ⊢  u1 = lnk-inv(lnk-inv(u1))
| | |   by RWO
| | |      Thm* ∀l:IdLnk. lnk-inv(lnk-inv(l)) = l ∈ IdLnk
| | |      {lnk_DASH_inv_DASH_inv}
| | |      0
| | |      THEN
| | |      MaAuto
| | |
| | .\
| | . 1. T : Id→𝔹
| | . 2. to : |T|→(IdLnk List)
| | . 3. from : |T|→(IdLnk List)
| | . 4. f : Edge(T)→𝔹
| | . 5. bi-tree(T;to;from)
| | . 6. bi-graph(T;to;from)
| | . 7. ∀i,j:|T|.
| | . |  ∃p:Edge(T) List.
| | . |  lconnects(p;i;j)
| | . |  & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | . 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | . 9. |T|
| | . 10. spanner(f;T;to;from)
| | . 11. i : |T|
| | . 12. j : |T|
| | . 13. spanner-root(f;T;to;from;i)
| | . 14. spanner-root(f;T;to;from;j)
| | . 15. Edge(T) List
| | . 16. u : Edge(T)
| | . 17. u1 : Edge(T)
| | . 18. v1 : Edge(T) List
| | . 19. 0<||v1||+1
| | . |   ⇒
| | . |   lpath([u1 / v1])
| | . |   ⇒
| | . |   f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | . 20. 0<||v1||+1+1
| | . 21. lpath([u1 / v1])
```

```
| | . 22. f(last([u1 / v1]))
| | . 23. destination(u) = source(u1)
| | . 24. ¬u1 = lnk-inv(u)
| | . 25. f(u1)
| | . 26. (∀l∈v1.f(l))
| | . 27. ∀l:Edge(T). f(l) = ¬₂f(inverse(l))
| | . 28. ∀i:|T|, l1,l2:Edge(T).
| | . |   (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | . 29. f(u1) = ¬₂f(inverse(u1))
| | . 30. f(u) ∨ f(inverse(u1))
| | . ⊢  f(u)
| | . | by Analyze -1 THEN Subst' (f(inverse(u1)) = ¬₂f(u1)) -1
| | . |
| | . |\
| | . | 1. T : Id→𝔹
| | . | 2. to : |T|→(IdLnk List)
| | . | 3. from : |T|→(IdLnk List)
| | . | 4. f : Edge(T)→𝔹
| | . | 5. bi-tree(T;to;from)
| | . | 6. bi-graph(T;to;from)
| | . | 7. ∀i,j:|T|.
| | . | |  ∃p:Edge(T) List.
| | . | |  lconnects(p;i;j)
| | . | |  & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | . | 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | . | 9. |T|
| | . | 10. spanner(f;T;to;from)
| | . | 11. i : |T|
| | . | 12. j : |T|
| | . | 13. spanner-root(f;T;to;from;i)
| | . | 14. spanner-root(f;T;to;from;j)
| | . | 15. Edge(T) List
| | . | 16. u : Edge(T)
| | . | 17. u1 : Edge(T)
| | . | 18. v1 : Edge(T) List
| | . | 19. 0<||v1||+1
| | . | |   ⇒
| | . | |   lpath([u1 / v1])
| | . | |   ⇒
| | . | |   f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | . | 20. 0<||v1||+1+1
| | . | 21. lpath([u1 / v1])
| | . | 22. f(last([u1 / v1]))
| | . | 23. destination(u) = source(u1)
| | . | 24. ¬u1 = lnk-inv(u)
| | . | 25. f(u1)
| | . | 26. (∀l∈v1.f(l))
| | . | 27. ∀l:Edge(T). f(l) = ¬₂f(inverse(l))
| | . | 28. ∀i:|T|, l1,l2:Edge(T).
| | . | |   (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | . | 29. f(u1) = ¬₂f(inverse(u1))
| | . | 30. f(inverse(u1))
| | . | ⊢  f(inverse(u1)) = ¬₂f(u1)  by HypSubst' -2 0
| | . | |
| | . | 1. T : Id→𝔹
```

```
| | . | 2. to : |T|→(IdLnk List)
| | . | 3. from : |T|→(IdLnk List)
| | . | 4. f : Edge(T)→𝔹
| | . | 5. bi-tree(T;to;from)
| | . | 6. bi-graph(T;to;from)
| | . | 7. ∀i,j:|T|.
| | . |    ∃p:Edge(T) List.
| | . |    lconnects(p;i;j)
| | . |    & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | . | 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | . | 9. |T|
| | . | 10. spanner(f;T;to;from)
| | . | 11. i : |T|
| | . | 12. j : |T|
| | . | 13. spanner-root(f;T;to;from;i)
| | . | 14. spanner-root(f;T;to;from;j)
| | . | 15. Edge(T) List
| | . | 16. u : Edge(T)
| | . | 17. u1 : Edge(T)
| | . | 18. v1 : Edge(T) List
| | . | 19. 0<||v1||+1
| | . |     ⇒
| | . |     lpath([u1 / v1])
| | . |     ⇒
| | . |     f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | . | 20. 0<||v1||+1+1
| | . | 21. lpath([u1 / v1])
| | . | 22. f(last([u1 / v1]))
| | . | 23. destination(u) = source(u1)
| | . | 24. ¬u1 = lnk-inv(u)
| | . | 25. f(u1)
| | . | 26. (∀l∈v1.f(l))
| | . | 27. ∀l:Edge(T). f(l) = ¬₂f(inverse(l))
| | . | 28. ∀i:|T|, l1,l2:Edge(T).
| | . |     (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | . | 29. f(u1) = ¬₂f(inverse(u1))
| | . | 30. f(inverse(u1))
| | . | ⊢  f(inverse(u1)) = ¬₂¬₂f(inverse(u1))
| | . |   by RWO Thm* ∀p:𝔹. ¬₂¬₂p = p    {bnot_bnot_elim} 0
| | . |
| | . .\
| | . . 1. T : Id→𝔹
| | . . 2. to : |T|→(IdLnk List)
| | . . 3. from : |T|→(IdLnk List)
| | . . 4. f : Edge(T)→𝔹
| | . . 5. bi-tree(T;to;from)
| | . . 6. bi-graph(T;to;from)
| | . . 7. ∀i,j:|T|.
| | . .    ∃p:Edge(T) List.
| | . .    lconnects(p;i;j)
| | . .    & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| | . . 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| | . . 9. |T|
| | . . 10. spanner(f;T;to;from)
| | . . 11. i : |T|
```

```
| | . . 12. j : |T|
| | . . 13. spanner-root(f;T;to;from;i)
| | . . 14. spanner-root(f;T;to;from;j)
| | . . 15. Edge(T) List
| | . . 16. u : Edge(T)
| | . . 17. u1 : Edge(T)
| | . . 18. v1 : Edge(T) List
| | . . 19. 0<||v1||+1
| | . .     ⇒
| | . .     lpath([u1 / v1])
| | . .     ⇒
| | . .     f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| | . . 20. 0<||v1||+1+1
| | . . 21. lpath([u1 / v1])
| | . . 22. f(last([u1 / v1]))
| | . . 23. destination(u) = source(u1)
| | . . 24. ¬u1 = lnk-inv(u)
| | . . 25. f(u1)
| | . . 26. (∀l∈v1.f(l))
| | . . 27. ∀l:Edge(T). f(l) = ¬2f(inverse(l))
| | . . 28. ∀i:|T|, l1,l2:Edge(T).
| | . .     (l1 ∈ to(i)) ⇒ (l2 ∈ to(i)) ⇒ ¬l1 = l2 ⇒ f(l1) ∨ f(l2)
| | . . 29. f(u1) = ¬2f(inverse(u1))
| | . . 30. ¬2f(u1)
| | . . ⊢  f(u)  by RW assert_pushdownC -1
| |
| .\
| . 1. T : Id→𝔹
| . 2. to : |T|→(IdLnk List)
| . 3. from : |T|→(IdLnk List)
| . 4. f : Edge(T)→𝔹
| . 5. bi-tree(T;to;from)
| . 6. bi-graph(T;to;from)
| . 7. ∀i,j:|T|.
| .    ∃p:Edge(T) List.
| .    lconnects(p;i;j) & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
| . 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
| . 9. |T|
| . 10. spanner(f;T;to;from)
| . 11. i : |T|
| . 12. j : |T|
| . 13. spanner-root(f;T;to;from;i)
| . 14. spanner-root(f;T;to;from;j)
| . 15. Edge(T) List
| . 16. u : Edge(T)
| . 17. u1 : Edge(T)
| . 18. v1 : Edge(T) List
| . 19. 0<||v1||+1
| .     ⇒
| .     lpath([u1 / v1]) ⇒ f(last([u1 / v1])) ⇒ (∀l∈[u1 / v1].f(l))
| . 20. 0<||v1||+1+1
| . 21. lpath([u1 / v1])
| . 22. ¬||[u1 / v1]|| = 0
| .     ⇒
| .     destination(u) = source(hd([u1 / v1]))
```

```
| .     & ¬hd([u1 / v1]) = lnk-inv(u)
| . 23. f(last([u1 / v1]))
| . ⊢  (∀l∈[u1 / v1].f(l))  by BackThruSomeHyp
|
.\
. 1. T : Id→𝔹
. 2. to : |T|→(IdLnk List)
. 3. from : |T|→(IdLnk List)
. 4. f : Edge(T)→𝔹
. 5. bi-tree(T;to;from)
. 6. bi-graph(T;to;from)
. 7. ∀i,j:|T|.
. |  ∃p:Edge(T) List.
. |  lconnects(p;i;j) & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. 8. ∃L:|T| List. ∀i:|T|. (i ∈ L)
. 9. |T|
. 10. spanner(f;T;to;from)
. 11. i : |T|
. 12. j : |T|
. 13. spanner-root(f;T;to;from;i)
. 14. spanner-root(f;T;to;from;j)
. 15. ∀p:Edge(T) List.
. |   0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l))
. ⊢  i = j
. | by DupHyp 5 THEN Unfold ‘bi-tree‘ -1 THEN ExRepD
. |    THEN
. |    InstHyp [i;j] -4
. |    THEN
. |    ExRepD
. |    THEN
. |    Analyze -2
. |    THEN
. |    ExRepD
. |
. 1. T : Id→𝔹
. 2. to : |T|→(IdLnk List)
. 3. from : |T|→(IdLnk List)
. 4. f : Edge(T)→𝔹
. 5. bi-tree(T;to;from)
. 6. bi-graph(T;to;from)
. 7. ∀i,j:|T|.
. |  ∃p:Edge(T) List.
. |  lconnects(p;i;j) & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. 8. L1 : |T| List
. 9. ∀i:|T|. (i ∈ L1)
. 10. |T|
. 11. spanner(f;T;to;from)
. 12. i : |T|
. 13. j : |T|
. 14. spanner-root(f;T;to;from;i)
. 15. spanner-root(f;T;to;from;j)
. 16. ∀p:Edge(T) List.
. |   0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l))
. 17. bi-graph(T;to;from)
. 18. ∀i,j:|T|.
```

```
. |   ∃p:Edge(T) List.
. |   lconnects(p;i;j) & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. 19. L : |T| List
. 20. ∀i:|T|. (i ∈ L)
. 21. |T|
. 22. p : Edge(T) List
. 23. lpath(p)
. 24. ||p|| = 0 ⇒ i = j
. 25. ¬||p|| = 0 ⇒ i = source(hd(p)) & j = destination(last(p))
. 26. ∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p
. ⊢  i = j  by DVar ‘p’ THEN All Reduce
. |
. |\
. | 1. T : Id→𝔹
. | 2. to : |T|→(IdLnk List)
. | 3. from : |T|→(IdLnk List)
. | 4. f : Edge(T)→𝔹
. | 5. bi-tree(T;to;from)
. | 6. bi-graph(T;to;from)
. | 7. ∀i,j:|T|.
. |    ∃p:Edge(T) List.
. |    lconnects(p;i;j) & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. | 8. L1 : |T| List
. | 9. ∀i:|T|. (i ∈ L1)
. | 10. |T|
. | 11. spanner(f;T;to;from)
. | 12. i : |T|
. | 13. j : |T|
. | 14. spanner-root(f;T;to;from;i)
. | 15. spanner-root(f;T;to;from;j)
. | 16. ∀p:Edge(T) List.
. |     0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l))
. | 17. bi-graph(T;to;from)
. | 18. ∀i,j:|T|.
. |     ∃p:Edge(T) List.
. |     lconnects(p;i;j)
. |     & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. | 19. L : |T| List
. | 20. ∀i:|T|. (i ∈ L)
. | 21. |T|
. | 22. lpath(nil)
. | 23. 0 = 0 ⇒ i = j
. | 24. ¬0 = 0 ⇒ i = source("?") & j = destination(last(nil))
. | 25. ∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = nil
. | ⊢  i = j  by Analyze -3 THEN ParallelLast
. |
. .\
. . 1. T : Id→𝔹
. . 2. to : |T|→(IdLnk List)
. . 3. from : |T|→(IdLnk List)
. . 4. f : Edge(T)→𝔹
. . 5. bi-tree(T;to;from)
. . 6. bi-graph(T;to;from)
. . 7. ∀i,j:|T|.
. . |  ∃p:Edge(T) List.
```

```
. . |  lconnects(p;i;j) & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . 8. L1 : |T| List
. . 9. ∀i:|T|. (i ∈ L1)
. . 10. |T|
. . 11. spanner(f;T;to;from)
. . 12. i : |T|
. . 13. j : |T|
. . 14. spanner-root(f;T;to;from;i)
. . 15. spanner-root(f;T;to;from;j)
. . 16. ∀p:Edge(T) List.
. . |   0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l))
. . 17. bi-graph(T;to;from)
. . 18. ∀i,j:|T|.
. . |   ∃p:Edge(T) List.
. . |   lconnects(p;i;j)
. . |   & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . 19. L : |T| List
. . 20. ∀i:|T|. (i ∈ L)
. . 21. |T|
. . 22. u : Edge(T)
. . 23. v : Edge(T) List
. . 24. lpath([u / v])
. . 25. ||v||+1 = 0 ⇒ i = j
. . 26. ¬||v||+1 = 0 ⇒ i = source(u) & j = destination(last([u / v]))
. . 27. ∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = [u / v]
. . ⊢  i = j  by Assert (∀l∈[u / v].f(l))
. . |
. . |\
. . | .....assertion .....
. . | 1. T : Id→𝔹
. . | 2. to : |T|→(IdLnk List)
. . | 3. from : |T|→(IdLnk List)
. . | 4. f : Edge(T)→𝔹
. . | 5. bi-tree(T;to;from)
. . | 6. bi-graph(T;to;from)
. . | 7. ∀i,j:|T|.
. . | |  ∃p:Edge(T) List.
. . | |  lconnects(p;i;j)
. . | |  & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . | 8. L1 : |T| List
. . | 9. ∀i:|T|. (i ∈ L1)
. . | 10. |T|
. . | 11. spanner(f;T;to;from)
. . | 12. i : |T|
. . | 13. j : |T|
. . | 14. spanner-root(f;T;to;from;i)
. . | 15. spanner-root(f;T;to;from;j)
. . | 16. ∀p:Edge(T) List.
. . | |   0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l))
. . | 17. bi-graph(T;to;from)
. . | 18. ∀i,j:|T|.
. . | |   ∃p:Edge(T) List.
. . | |   lconnects(p;i;j)
. . | |   & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . | 19. L : |T| List
```

```
. . | 20. ∀i:|T|. (i ∈ L)
. . | 21. |T|
. . | 22. u : Edge(T)
. . | 23. v : Edge(T) List
. . | 24. lpath([u / v])
. . | 25. ||v||+1 = 0 ⇒ i = j
. . | 26. ¬||v||+1 = 0
. . | |   ⇒
. . | |   i = source(u) & j = destination(last([u / v]))
. . | 27. ∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = [u / v]
. . | ⊢  (∀l∈[u / v].f(l))  by BackThruSomeHyp
. . | |
. . | 1. T : Id→𝔹
. . | 2. to : |T|→(IdLnk List)
. . | 3. from : |T|→(IdLnk List)
. . | 4. f : Edge(T)→𝔹
. . | 5. bi-tree(T;to;from)
. . | 6. bi-graph(T;to;from)
. . | 7. ∀i,j:|T|.
. . | |  ∃p:Edge(T) List.
. . | |  lconnects(p;i;j)
. . | |  & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . | 8. L1 : |T| List
. . | 9. ∀i:|T|. (i ∈ L1)
. . | 10. |T|
. . | 11. spanner(f;T;to;from)
. . | 12. i : |T|
. . | 13. j : |T|
. . | 14. spanner-root(f;T;to;from;i)
. . | 15. spanner-root(f;T;to;from;j)
. . | 16. ∀p:Edge(T) List.
. . | |   0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l))
. . | 17. bi-graph(T;to;from)
. . | 18. ∀i,j:|T|.
. . | |   ∃p:Edge(T) List.
. . | |   lconnects(p;i;j)
. . | |   & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . | 19. L : |T| List
. . | 20. ∀i:|T|. (i ∈ L)
. . | 21. |T|
. . | 22. u : Edge(T)
. . | 23. v : Edge(T) List
. . | 24. lpath([u / v])
. . | 25. ||v||+1 = 0 ⇒ i = j
. . | 26. ¬||v||+1 = 0
. . | |   ⇒
. . | |   i = source(u) & j = destination(last([u / v]))
. . | 27. ∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = [u / v]
. . | ⊢  f(last([u / v]))
. . | | by AllHyps
. . | |    (λh.Unfold ‘spanner-root‘ h THEN BackThru h THEN Reduce 0)
. . | |
. . | 1. T : Id→𝔹
. . | 2. to : |T|→(IdLnk List)
. . | 3. from : |T|→(IdLnk List)
```

```
. . | 4. f : Edge(T)→𝔹
. . | 5. bi-tree(T;to;from)
. . | 6. bi-graph(T;to;from)
. . | 7. ∀i,j:|T|.
. . | |  ∃p:Edge(T) List.
. . | |  lconnects(p;i;j)
. . | |  & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . | 8. L1 : |T| List
. . | 9. ∀i:|T|. (i ∈ L1)
. . | 10. |T|
. . | 11. spanner(f;T;to;from)
. . | 12. i : |T|
. . | 13. j : |T|
. . | 14. spanner-root(f;T;to;from;i)
. . | 15. ∀l:Edge(T). (l ∈ to(j)) ⇒ f(l)
. . | 16. ∀p:Edge(T) List.
. . | |   0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l))
. . | 17. bi-graph(T;to;from)
. . | 18. ∀i,j:|T|.
. . | |   ∃p:Edge(T) List.
. . | |   lconnects(p;i;j)
. . | |   & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . | 19. L : |T| List
. . | 20. ∀i:|T|. (i ∈ L)
. . | 21. |T|
. . | 22. u : Edge(T)
. . | 23. v : Edge(T) List
. . | 24. lpath([u / v])
. . | 25. ||v||+1 = 0 ⇒ i = j
. . | 26. ¬||v||+1 = 0
. . | |   ⇒
. . | |   i = source(u) & j = destination(last([u / v]))
. . | 27. ∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = [u / v]
. . | ⊢  (last([u / v]) ∈ to(j))
. . | | by Analyze -2 THEN HypSubst' -1 0
. . | |    THEN
. . | |    BackThru
. . | |    Thm* ∀G:(Id→𝔹), to,from:(|G|→(IdLnk List)), e:Edge(G),
. . | |          i:|G|.
. . | |         bi-graph(G;to;from)
. . | |         ⇒
. . | |         ((e ∈ to(i)) ⟺ destination(e) = i)
. . | |    {edge_DASH_to}
. . | |    THEN
. . | |    Reduce 0
. . | |
. . | .....wf .....
. . | 1. T : Id→𝔹
. . | 2. to : |T|→(IdLnk List)
. . | 3. from : |T|→(IdLnk List)
. . | 4. f : Edge(T)→𝔹
. . | 5. bi-tree(T;to;from)
. . | 6. bi-graph(T;to;from)
. . | 7. ∀i,j:|T|.
. . |    ∃p:Edge(T) List.
```

```
. . |    lconnects(p;i;j)
. . |    & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . | 8. L1 : |T| List
. . | 9. ∀i:|T|. (i ∈ L1)
. . | 10. |T|
. . | 11. spanner(f;T;to;from)
. . | 12. i : |T|
. . | 13. j : |T|
. . | 14. spanner-root(f;T;to;from;i)
. . | 15. ∀l:Edge(T). (l ∈ to(j)) ⇒ f(l)
. . | 16. ∀p:Edge(T) List.
. . |     0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l))
. . | 17. bi-graph(T;to;from)
. . | 18. ∀i,j:|T|.
. . |     ∃p:Edge(T) List.
. . |     lconnects(p;i;j)
. . |     & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . | 19. L : |T| List
. . | 20. ∀i:|T|. (i ∈ L)
. . | 21. |T|
. . | 22. u : Edge(T)
. . | 23. v : Edge(T) List
. . | 24. lpath([u / v])
. . | 25. ||v||+1 = 0 ⇒ i = j
. . | 26. ∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = [u / v]
. . | 27. i = source(u)
. . | 28. j = destination(last([u / v]))
. . | ⊢  destination(last([u / v])) ∈ |T|
. . |   by BackThru
. . |      Thm* ∀T:(Id→𝔹), to,from:(|T|→(IdLnk List)), u:Edge(T).
. . |           bi-graph(T;to;from) ⇒ destination(u) ∈ |T|
. . |      {dst_DASH_edge}
. . |      THEN
. . |      Reduce 0
. . |
. . .\
. . . 1. T : Id→𝔹
. . . 2. to : |T|→(IdLnk List)
. . . 3. from : |T|→(IdLnk List)
. . . 4. f : Edge(T)→𝔹
. . . 5. bi-tree(T;to;from)
. . . 6. bi-graph(T;to;from)
. . . 7. ∀i,j:|T|.
. . . |  ∃p:Edge(T) List.
. . . |  lconnects(p;i;j)
. . . |  & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . . 8. L1 : |T| List
. . . 9. ∀i:|T|. (i ∈ L1)
. . . 10. |T|
. . . 11. spanner(f;T;to;from)
. . . 12. i : |T|
. . . 13. j : |T|
. . . 14. spanner-root(f;T;to;from;i)
. . . 15. spanner-root(f;T;to;from;j)
. . . 16. ∀p:Edge(T) List.
```

```
. . . |   0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l))
. . . 17. bi-graph(T;to;from)
. . . 18. ∀i,j:|T|.
. . . |   ∃p:Edge(T) List.
. . . |   lconnects(p;i;j)
. . . |   & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . . 19. L : |T| List
. . . 20. ∀i:|T|. (i ∈ L)
. . . 21. |T|
. . . 22. u : Edge(T)
. . . 23. v : Edge(T) List
. . . 24. lpath([u / v])
. . . 25. ||v||+1 = 0 ⇒ i = j
. . . 26. ¬||v||+1 = 0
. . . |   ⇒
. . . |   i = source(u) & j = destination(last([u / v]))
. . . 27. ∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = [u / v]
. . . 28. (∀l∈[u / v].f(l))
. . . ⊢  i = j
. . . | by RWO
. . . |    Thm* ∀P:(T→Prop), x:T, L:T List.
. . . |         (∀y∈[x / L].P(y)) ⟺ P(x) & (∀y∈L.P(y))
. . . |    {l_all_cons}
. . . |    -1
. . . |
. . . 1. T : Id→𝔹
. . . 2. to : |T|→(IdLnk List)
. . . 3. from : |T|→(IdLnk List)
. . . 4. f : Edge(T)→𝔹
. . . 5. bi-tree(T;to;from)
. . . 6. bi-graph(T;to;from)
. . . 7. ∀i,j:|T|.
. . . |  ∃p:Edge(T) List.
. . . |  lconnects(p;i;j)
. . . |  & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . . 8. L1 : |T| List
. . . 9. ∀i:|T|. (i ∈ L1)
. . . 10. |T|
. . . 11. spanner(f;T;to;from)
. . . 12. i : |T|
. . . 13. j : |T|
. . . 14. spanner-root(f;T;to;from;i)
. . . 15. spanner-root(f;T;to;from;j)
. . . 16. ∀p:Edge(T) List.
. . . |   0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l))
. . . 17. bi-graph(T;to;from)
. . . 18. ∀i,j:|T|.
. . . |   ∃p:Edge(T) List.
. . . |   lconnects(p;i;j)
. . . |   & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . . 19. L : |T| List
. . . 20. ∀i:|T|. (i ∈ L)
. . . 21. |T|
. . . 22. u : Edge(T)
. . . 23. v : Edge(T) List
```

```
. . . 24. lpath([u / v])
. . . 25. ||v||+1 = 0 ⇒ i = j
. . . 26. ¬||v||+1 = 0
. . . |   ⇒
. . . |   i = source(u) & j = destination(last([u / v]))
. . . 27. ∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = [u / v]
. . . 28. f(u)
. . . 29. (∀l∈v.f(l))
. . . ⊢  i = j  by Analyze -4
. . . |
. . . 1. T : Id→𝔹
. . . 2. to : |T|→(IdLnk List)
. . . 3. from : |T|→(IdLnk List)
. . . 4. f : Edge(T)→𝔹
. . . 5. bi-tree(T;to;from)
. . . 6. bi-graph(T;to;from)
. . . 7. ∀i,j:|T|.
. . . |  ∃p:Edge(T) List.
. . . |  lconnects(p;i;j)
. . . |  & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . . 8. L1 : |T| List
. . . 9. ∀i:|T|. (i ∈ L1)
. . . 10. |T|
. . . 11. spanner(f;T;to;from)
. . . 12. i : |T|
. . . 13. j : |T|
. . . 14. spanner-root(f;T;to;from;i)
. . . 15. spanner-root(f;T;to;from;j)
. . . 16. ∀p:Edge(T) List.
. . . |   0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l))
. . . 17. bi-graph(T;to;from)
. . . 18. ∀i,j:|T|.
. . . |   ∃p:Edge(T) List.
. . . |   lconnects(p;i;j)
. . . |   & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . . 19. L : |T| List
. . . 20. ∀i:|T|. (i ∈ L)
. . . 21. |T|
. . . 22. u : Edge(T)
. . . 23. v : Edge(T) List
. . . 24. lpath([u / v])
. . . 25. ||v||+1 = 0 ⇒ i = j
. . . 26. ∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = [u / v]
. . . 27. f(u)
. . . 28. (∀l∈v.f(l))
. . . 29. i = source(u)
. . . 30. j = destination(last([u / v]))
. . . ⊢  i = j  by Assert (f(inverse(u)))
. . . |
. . . |\
. . . | .....assertion .....
. . . | 1. T : Id→𝔹
. . . | 2. to : |T|→(IdLnk List)
. . . | 3. from : |T|→(IdLnk List)
. . . | 4. f : Edge(T)→𝔹
```

```
. . . | 5. bi-tree(T;to;from)
. . . | 6. bi-graph(T;to;from)
. . . | 7. ∀i,j:|T|.
. . . |    ∃p:Edge(T) List.
. . . |    lconnects(p;i;j)
. . . |    & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . . | 8. L1 : |T| List
. . . | 9. ∀i:|T|. (i ∈ L1)
. . . | 10. |T|
. . . | 11. spanner(f;T;to;from)
. . . | 12. i : |T|
. . . | 13. j : |T|
. . . | 14. spanner-root(f;T;to;from;i)
. . . | 15. spanner-root(f;T;to;from;j)
. . . | 16. ∀p:Edge(T) List.
. . . |     0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l))
. . . | 17. bi-graph(T;to;from)
. . . | 18. ∀i,j:|T|.
. . . |     ∃p:Edge(T) List.
. . . |     lconnects(p;i;j)
. . . |     & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . . | 19. L : |T| List
. . . | 20. ∀i:|T|. (i ∈ L)
. . . | 21. |T|
. . . | 22. u : Edge(T)
. . . | 23. v : Edge(T) List
. . . | 24. lpath([u / v])
. . . | 25. ||v||+1 = 0 ⇒ i = j
. . . | 26. ∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = [u / v]
. . . | 27. f(u)
. . . | 28. (∀l∈v.f(l))
. . . | 29. i = source(u)
. . . | 30. j = destination(last([u / v]))
. . . | ⊢  f(inverse(u))
. . . |   by AllHyps
. . . |      (λh.
. . . |       Unfold `spanner-root` h THEN BackThru h
. . . |       THEN
. . . |       HypSubst' -2 0
. . . |       THEN
. . . |       BackThru
. . . |       Thm* ∀G:(Id→𝔹), to,from:(|G|→(IdLnk List)), e:Edge(G),
. . . |             i:|G|.
. . . |            bi-graph(G;to;from)
. . . |            ⇒
. . . |            ((inverse(e) ∈ to(i)) ⟺ source(e) = i)
. . . |       {edge_DASH_inv_DASH_to}
. . . |       THEN
. . . |       BackThru
. . . |       Thm* ∀T:(Id→𝔹), to,from:(|T|→(IdLnk List)), u:Edge(T).
. . . |            bi-graph(T;to;from) ⇒ source(u) ∈ |T|
. . . |       {src_DASH_edge})
. . . |
. . . .\
. . . . 1. T : Id→𝔹
```

```
. . . . . 2. to : |T|→(IdLnk List)
. . . . . 3. from : |T|→(IdLnk List)
. . . . . 4. f : Edge(T)→𝔹
. . . . . 5. bi-tree(T;to;from)
. . . . . 6. bi-graph(T;to;from)
. . . . . 7. ∀i,j:|T|.
. . . .      ∃p:Edge(T) List.
. . . .      lconnects(p;i;j)
. . . .      & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . . . . 8. L1 : |T| List
. . . . . 9. ∀i:|T|. (i ∈ L1)
. . . . . 10. |T|
. . . . . 11. spanner(f;T;to;from)
. . . . . 12. i : |T|
. . . . . 13. j : |T|
. . . . . 14. spanner-root(f;T;to;from;i)
. . . . . 15. spanner-root(f;T;to;from;j)

. . . . . 16. ∀p:Edge(T) List.
. . . .       0<||p|| ⇒ lpath(p) ⇒ f(last(p)) ⇒ (∀l∈p.f(l))
. . . . . 17. bi-graph(T;to;from)
. . . . . 18. ∀i,j:|T|.
. . . .       ∃p:Edge(T) List.
. . . .       lconnects(p;i;j)
. . . .       & (∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = p)
. . . . . 19. L : |T| List
. . . . . 20. ∀i:|T|. (i ∈ L)
. . . . . 21. |T|
. . . . . 22. u : Edge(T)
. . . . . 23. v : Edge(T) List
. . . . . 24. lpath([u / v])
. . . . . 25. ||v||+1 = 0 ⇒ i = j
. . . . . 26. ∀q:Edge(T) List. lconnects(q;i;j) ⇒ q = [u / v]
. . . . . 27. f(u)
. . . . . 28. (∀l∈v.f(l))
. . . . . 29. i = source(u)
. . . . . 30. j = destination(last([u / v]))
. . . . . 31. f(inverse(u))
. . . . . ⊢  i = j
. . . .     by AllHyps
. . . .        (λh.
. . . .         DupHyp h THEN Unfold ‘spanner‘ -1 THEN Analyze -1
. . . .         THEN
. . . .         InstHyp [u] -2
. . . .         THEN
. . . .         HypSubst’ -1 -8
. . . .         THEN
. . . .         RW assert_pushdownC -8)
END
```