da-outlinks-join THM

```
⊢  ∀ltg:(IdLnk×Id×Type), i:Id, da1,da2:k:Knd fp-> Type.
|  (ltg ∈ da-outlinks(da1 ⊕ da2;i))
|  ⇒
|  (ltg ∈ da-outlinks(da1;i)) ∨ (ltg ∈ da-outlinks(da2;i))
| by Auto THEN All (Unfold ‘da-outlinks‘)
|     THEN
|     Inst
|     Thm* ∀P:(T→𝔹), T’:Type, f:({x:T| P(x) }→T’), L:T List, x:T’.
|          (x ∈ mapfilter(f;P;L)) ⟺ (∃y:T. (y ∈ L) & P(y) & x = f(y))
|     {member_map_filter}
|     [lexpr{i’}
|     ;{k:Knd| k ∈ dom(da1) }
|     ;λk.has-src(i;k)
|     ;IdLnk×Id×Type
|     ;λk.da-outlink-f(da1;k)
|     ;fpf-dom-list(da1)
|     ;ltg]
|     THENA
|     (Reduce 0 THEN DVar ‘k’ THEN DVar ‘k’
|      THEN
|      RWO
|      Thm* ∀i:Id, k:Knd. has-src(i;k) ⟺ isrcv(k) & source(lnk(k)) = i
|      {assert_DASH_has_DASH_src}
|      -1
|      THEN
|      ExRepD)
|     THEN
|     Inst
|     Thm* ∀P:(T→𝔹), T’:Type, f:({x:T| P(x) }→T’), L:T List, x:T’.
|          (x ∈ mapfilter(f;P;L)) ⟺ (∃y:T. (y ∈ L) & P(y) & x = f(y))
|     {member_map_filter}
|     [lexpr{i’}
|     ;{k:Knd| k ∈ dom(da2) }
|     ;λk.has-src(i;k)
|     ;IdLnk×Id×Type
|     ;λk.da-outlink-f(da2;k)
|     ;fpf-dom-list(da2)
|     ;ltg]
|     THENA
|     (Reduce 0 THEN DVar ‘k’ THEN DVar ‘k’
|      THEN
|      RWO
|      Thm* ∀i:Id, k:Knd. has-src(i;k) ⟺ isrcv(k) & source(lnk(k)) = i
|      {assert_DASH_has_DASH_src}
|      -1
|      THEN
|      ExRepD)
|     THEN
|     Inst
|     Thm* ∀P:(T→𝔹), T’:Type, f:({x:T| P(x) }→T’), L:T List, x:T’.
|          (x ∈ mapfilter(f;P;L)) ⟺ (∃y:T. (y ∈ L) & P(y) & x = f(y))
|     {member_map_filter}
|     [lexpr{i’}
```

```
|     ;{k:Knd| k ∈ dom(da1 ⊕ da2) }
|     ;λk.has-src(i;k)
|     ;IdLnk×Id×Type
|     ;λk.da-outlink-f(da1 ⊕ da2;k)
|     ;fpf-dom-list(da1 ⊕ da2)
|     ;ltg]
|    THENA
|    (Reduce 0 THEN DVar ‘k’ THEN DVar ‘k’
|     THEN
|     RWO
|     Thm* ∀i:Id, k:Knd. has-src(i;k) ⟺ isrcv(k) & source(lnk(k)) = i
|     {assert_DASH_has_DASH_src}
|     -1
|     THEN
|     ExRepD)
|    THEN
|    All Reduce
|
1. ltg : IdLnk×Id×Type
2. i : Id
3. da1 : k:Knd fp-> Type
4. da2 : k:Knd fp-> Type
5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
|  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
|  has-src(i;k);fpf-dom-list(da1)))
|  ⟺
|  (∃y:{k:Knd| k ∈ dom(da1) }.
|   (y ∈ fpf-dom-list(da1))
|   & has-src(i;y) & ltg = da-outlink-f(da1;y))
7. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
|  has-src(i;k);fpf-dom-list(da2)))
|  ⟺
|  (∃y:{k:Knd| k ∈ dom(da2) }.
|   (y ∈ fpf-dom-list(da2))
|   & has-src(i;y) & ltg = da-outlink-f(da2;y))
8. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
|  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
|  ⟺
|  (∃y:{k:Knd| k ∈ dom(da1 ⊕ da2) }.
|   (y ∈ fpf-dom-list(da1 ⊕ da2))
|   & has-src(i;y) & ltg = da-outlink-f(da1 ⊕ da2;y))
⊢  (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
|  has-src(i;k);fpf-dom-list(da1)))
|  ∨ (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
|    has-src(i;k);fpf-dom-list(da2)))
| by ThinTrivial THEN Thin -2 THEN ExRepD

|
1. ltg : IdLnk×Id×Type
2. i : Id
3. da1 : k:Knd fp-> Type
4. da2 : k:Knd fp-> Type
5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
|  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
```

```
|  has-src(i;k);fpf-dom-list(da1)))
|  ⟺
|  (∃y:{k:Knd| k ∈ dom(da1) }.
|   (y ∈ fpf-dom-list(da1))
|   & has-src(i;y) & ltg = da-outlink-f(da1;y))
7. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
|  has-src(i;k);fpf-dom-list(da2)))
|  ⟺
|  (∃y:{k:Knd| k ∈ dom(da2) }.
|   (y ∈ fpf-dom-list(da2))
|   & has-src(i;y) & ltg = da-outlink-f(da2;y))
8. y : {k:Knd| k ∈ dom(da1 ⊕ da2) }
9. (y ∈ fpf-dom-list(da1 ⊕ da2))
10. has-src(i;y)
11. ltg = da-outlink-f(da1 ⊕ da2;y)
⊢  (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
|  has-src(i;k);fpf-dom-list(da1)))
|  ∨ (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
|    has-src(i;k);fpf-dom-list(da2)))
| by Analyze -4 THEN AssertBY y ∈ dom(da1 ⊕ da2) Unhide
|    THEN
|    Decide y ∈ dom(da1)
|
|\
| 1. ltg : IdLnk×Id×Type
| 2. i : Id
| 3. da1 : k:Knd fp-> Type
| 4. da2 : k:Knd fp-> Type
| 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
| |  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
| 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| |  has-src(i;k);fpf-dom-list(da1)))
| |  ⟺
| |  (∃y:{k:Knd| k ∈ dom(da1) }.
| |   (y ∈ fpf-dom-list(da1))
| |   & has-src(i;y) & ltg = da-outlink-f(da1;y))
| 7. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| |  has-src(i;k);fpf-dom-list(da2)))
| |  ⟺
| |  (∃y:{k:Knd| k ∈ dom(da2) }.
| |   (y ∈ fpf-dom-list(da2))
| |   & has-src(i;y) & ltg = da-outlink-f(da2;y))
| 8. y : Knd
| 9. y ∈ dom(da1 ⊕ da2)  [not for witness]
| 10. (y ∈ fpf-dom-list(da1 ⊕ da2))
| 11. has-src(i;y)
| 12. ltg = da-outlink-f(da1 ⊕ da2;y)
| 13. y ∈ dom(da1 ⊕ da2)
| 14. y ∈ dom(da1)
| ⊢  (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| |  has-src(i;k);fpf-dom-list(da1)))
| |  ∨ (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| |    has-src(i;k);fpf-dom-list(da2)))  by OrLeft
| |
| |\
```

```
| | 1. ltg : IdLnk×Id×Type
| | 2. i : Id
| | 3. da1 : k:Knd fp-> Type
| | 4. da2 : k:Knd fp-> Type
| | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
| | |  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
| | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | |  has-src(i;k);fpf-dom-list(da1)))
| | |  ⇒
| | |  (∃y:{k:Knd| k ∈ dom(da1) }.
| | |   (y ∈ fpf-dom-list(da1))
| | |   & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | |  has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:{k:Knd| k ∈ dom(da1) }.
| | |
| | |  (y ∈ fpf-dom-list(da1))
| | |  & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | |  has-src(i;k);fpf-dom-list(da2)))
| | |  ⇒
| | |  (∃y:{k:Knd| k ∈ dom(da2) }.
| | |   (y ∈ fpf-dom-list(da2))
| | |   & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | |  has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:{k:Knd| k ∈ dom(da2) }.
| | |
| | |  (y ∈ fpf-dom-list(da2))
| | |  & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | 10. y : Knd
| | 11. y ∈ dom(da1 ⊕ da2)  [not for witness]
| | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
| | 13. has-src(i;y)
| | 14. ltg = da-outlink-f(da1 ⊕ da2;y)
| | 15. y ∈ dom(da1 ⊕ da2)
| | 16. y ∈ dom(da1)
| | ⊢  (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | |  has-src(i;k);fpf-dom-list(da1)))
| | |
| | | by BackThruSomeHyp THEN InstConcl [y]
| | |
| | |\
| | | 1. ltg : IdLnk×Id×Type
| | | 2. i : Id
| | | 3. da1 : k:Knd fp-> Type
| | | 4. da2 : k:Knd fp-> Type
| | | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
| | | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da1)))
| | | |  ⇒
| | | |  (∃y:{k:Knd| k ∈ dom(da1) }.
| | | |   (y ∈ fpf-dom-list(da1))
| | | |   & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:....
| | | |
```

```
| | | |  (y ∈ fpf-dom-list(da1))
| | | |  & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da2)))
| | | |  ⇒
| | | |  (∃y:{k:Knd| k ∈ dom(da2) }.
| | | |   (y ∈ fpf-dom-list(da2))
| | | |   & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:....
| | | |
| | | |  (y ∈ fpf-dom-list(da2))
| | | |  & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | | 10. y : Knd
| | | 11. y ∈ dom(da1 ⊕ da2)  [not for witness]
| | | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
| | | 13. has-src(i;y)
| | | 14. ltg = da-outlink-f(da1 ⊕ da2;y)
| | | 15. y ∈ dom(da1 ⊕ da2)
| | | 16. y ∈ dom(da1)
| | | ⊢  (y ∈ fpf-dom-list(da1))
| | | | by AssertBY (fpf-dom-list(da1) ∈ {k:Knd| k ∈ dom(da1) } List)
| | | |    Auto
| | | |    THEN
| | | |    DupHyp -2
| | | |    THEN
| | | |    Unfold ‘fpf-dom‘ -1
| | | |    THEN
| | | |    Fold ‘fpf-dom-list‘ -1
| | | |    THEN
| | | |    RWO
| | | |    Thm* ∀eq:EqDecider(A), L:A List, x:A.
| | | |         deq-member(eq;x;L) ⟺ (x ∈ L)
| | | |    {assert_DASH_deq_DASH_member}
| | | |    -1
| | | |
| | | |\
| | | | .....wf .....
| | | | 1. ltg : IdLnk×Id×Type
| | | | 2. i : Id
| | | | 3. da1 : k:Knd fp-> Type
| | | | 4. da2 : k:Knd fp-> Type
| | | | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
| | | |    has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
| | | | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | | |    has-src(i;k);fpf-dom-list(da1)))
| | | |    ⇒
| | | |    (∃y:{k:Knd| k ∈ dom(da1) }.
| | | |     (y ∈ fpf-dom-list(da1))
| | | |     & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | | | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | | |    has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:{k:Knd
| | | |                                          | k ∈ dom(da1) }.
| | | |
| | | |    (y ∈ fpf-dom-list(da1))
```

```
| | | |    & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | | | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | | |    has-src(i;k);fpf-dom-list(da2)))
| | | |    ⇒
| | | |    (∃y:{k:Knd| k ∈ dom(da2) }.
| | | |     (y ∈ fpf-dom-list(da2))
| | | |     & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | | | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | | |    has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:{k:Knd
| | | |                                           | k ∈ dom(da2) }.
| | | |
| | | |    (y ∈ fpf-dom-list(da2))
| | | |    & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | | | 10. y : Knd
| | | | 11. y ∈ dom(da1 ⊕ da2)
| | | | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
| | | | 13. has-src(i;y)
| | | | 14. ltg = da-outlink-f(da1 ⊕ da2;y)
| | | | 15. y ∈ dom(da1 ⊕ da2)
| | | | 16. y ∈ dom(da1)
| | | | 17. fpf-dom-list(da1) ∈ {k:Knd| k ∈ dom(da1) } List
| | | | 18. deq-member(KindDeq;y;fpf-dom-list(da1))
| | | | ⊢  fpf-dom-list(da1) ∈ Knd List
| | | |   by DoSubsume THEN UseSubtypeRelLemmas THEN Analyze 0
| | | |
| | | .\
| | | . 1. ltg : IdLnk×Id×Type
| | | . 2. i : Id
| | | . 3. da1 : k:Knd fp-> Type
| | | . 4. da2 : k:Knd fp-> Type
| | | . 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
| | | .    has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
| | | . 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | | .    has-src(i;k);fpf-dom-list(da1)))
| | | .    ⇒
| | | .    (∃y:{k:Knd| k ∈ dom(da1) }.
| | | .     (y ∈ fpf-dom-list(da1))
| | | .     & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | | . 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | | .    has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:{k:Knd
| | | .                                           | k ∈ dom(da1) }.
| | | .
| | | .    (y ∈ fpf-dom-list(da1))
| | | .    & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | | . 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | | .    has-src(i;k);fpf-dom-list(da2)))
| | | .    ⇒
| | | .    (∃y:{k:Knd| k ∈ dom(da2) }.
| | | .     (y ∈ fpf-dom-list(da2))
| | | .     & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | | . 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | | .    has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:{k:Knd
| | | .                                           | k ∈ dom(da2) }.
| | | .
| | | .    (y ∈ fpf-dom-list(da2))
```

```
| | | .    & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | | . 10. y : Knd
| | | . 11. y ∈ dom(da1 ⊕ da2)  [not for witness]
| | | . 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
| | | . 13. has-src(i;y)
| | | . 14. ltg = da-outlink-f(da1 ⊕ da2;y)
| | | . 15. y ∈ dom(da1 ⊕ da2)
| | | . 16. y ∈ dom(da1)
| | | . 17. fpf-dom-list(da1) ∈ {k:Knd| k ∈ dom(da1) } List
| | | . 18. (y ∈ fpf-dom-list(da1))
| | | . ⊢  (y ∈ fpf-dom-list(da1))
| | | .   by RepeatFor 2 ParallelLast THEN ExRepD THEN Analyze
| | |
| | |\
| | | 1. ltg : IdLnk×Id×Type
| | | 2. i : Id
| | | 3. da1 : k:Knd fp-> Type
| | | 4. da2 : k:Knd fp-> Type
| | | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
| | | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da1)))
| | | |  ⇒
| | | |  (∃y:{k:Knd| k ∈ dom(da1) }.
| | | |   (y ∈ fpf-dom-list(da1))
| | | |   & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:....
| | | |
| | | |  (y ∈ fpf-dom-list(da1))
| | | |  & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da2)))
| | | |  ⇒
| | | |  (∃y:{k:Knd| k ∈ dom(da2) }.
| | | |   (y ∈ fpf-dom-list(da2))
| | | |   & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:....
| | | |
| | | |  (y ∈ fpf-dom-list(da2))
| | | |  & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | | 10. y : Knd
| | | 11. y ∈ dom(da1 ⊕ da2)
| | | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
| | | 13. has-src(i;y)
| | | 14. ltg = da-outlink-f(da1 ⊕ da2;y)
| | | 15. y ∈ dom(da1 ⊕ da2)
| | | 16. y ∈ dom(da1)
| | | ⊢  ltg = da-outlink-f(da1;y)
| | | | by Assert (da-outlink-f(da1;y) = da-outlink-f(da1 ⊕ da2;y))
| | | |
| | | .....assertion .....
| | | 1. ltg : IdLnk×Id×Type
| | | 2. i : Id
```

```
| | | 3. da1 : k:Knd fp-> Type
| | | 4. da2 : k:Knd fp-> Type
| | | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
| | | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da1)))
| | | |  ⇒
| | | |  (∃y:{k:Knd| k ∈ dom(da1) }.
| | | |   (y ∈ fpf-dom-list(da1))
| | | |   & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:....
| | | |
| | | |  (y ∈ fpf-dom-list(da1))
| | | |  & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da2)))
| | | |  ⇒
| | | |  (∃y:{k:Knd| k ∈ dom(da2) }.
| | | |   (y ∈ fpf-dom-list(da2))
| | | |   & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | | |  has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:....
| | | |
| | | |  (y ∈ fpf-dom-list(da2))
| | | |  & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | | 10. y : Knd
| | | 11. y ∈ dom(da1 ⊕ da2)
| | | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
| | | 13. has-src(i;y)
| | | 14. ltg = da-outlink-f(da1 ⊕ da2;y)
| | | 15. y ∈ dom(da1 ⊕ da2)
| | | 16. y ∈ dom(da1)
| | | ⊢  da-outlink-f(da1;y) = da-outlink-f(da1 ⊕ da2;y)
| | | | by RWO
| | | |    Thm* ∀i:Id, k:Knd.
| | | |         has-src(i;k) ⟺ isrcv(k) & source(lnk(k)) = i
| | | |    {assert_DASH_has_DASH_src}
| | | |    -4
| | | |    THEN
| | | |    ExRepD
| | | |    THEN
| | | |    Unfold `da-outlink-f` 0
| | | |    THEN
| | | |    Analyze
| | | |    THEN
| | | |    Analyze
| | | |
| | | .....subterm .....
| | | 1. ltg : IdLnk×Id×Type
| | | 2. i : Id
| | | 3. da1 : k:Knd fp-> Type
| | | 4. da2 : k:Knd fp-> Type
| | | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
| | |    has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
```

```
| | | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | |    has-src(i;k);fpf-dom-list(da1)))
| | |    ⇒
| | |    (∃y:{k:Knd| k ∈ dom(da1) }.
| | |     (y ∈ fpf-dom-list(da1))
| | |     & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | |    has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:....
| | |
| | |    (y ∈ fpf-dom-list(da1))
| | |    & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | |    has-src(i;k);fpf-dom-list(da2)))
| | |    ⇒
| | |    (∃y:{k:Knd| k ∈ dom(da2) }.
| | |     (y ∈ fpf-dom-list(da2))
| | |     & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | |    has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:....
| | |
| | |    (y ∈ fpf-dom-list(da2))
| | |    & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | | 10. y : Knd
| | | 11. y ∈ dom(da1 ⊕ da2)
| | | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
| | | 13. isrcv(y)
| | | 14. source(lnk(y)) = i
| | | 15. ltg = da-outlink-f(da1 ⊕ da2;y)
| | | 16. y ∈ dom(da1 ⊕ da2)
| | | 17. y ∈ dom(da1)
| | | ⊢  da1(y) = da1 ⊕ da2(y)
| | |   by RWO
| | |      Thm* ∀eq:EqDecider(A), f:a:A fp-> Top, g:Top, x:A.
| | |           f ⊕ g(x) ~ if x ∈ dom(f)→ f(x) else g(x) fi
| | |      {fpf_DASH_join_DASH_ap_DASH_sq}
| | |      0
| | |      THEN
| | |      SplitOnConclITE
| | |
| | .\
| | . .....subterm .....
| | . 1. ltg : IdLnk×Id×Type
| | . 2. i : Id
| | . 3. da1 : k:Knd fp-> Type
| | . 4. da2 : k:Knd fp-> Type
| | . 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
| | .    has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
| | . 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | .    has-src(i;k);fpf-dom-list(da1)))
| | .    ⇒
| | .    (∃y:{k:Knd| k ∈ dom(da1) }.
| | .     (y ∈ fpf-dom-list(da1))
| | .     & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | . 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| | .    has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:....
```

```
| | .
| | .    (y ∈ fpf-dom-list(da1))
| | .    & has-src(i;y) & ltg = da-outlink-f(da1;y))
| | . 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | .    has-src(i;k);fpf-dom-list(da2)))
| | .    ⇒
| | .    (∃y:{k:Knd| k ∈ dom(da2) }.
| | .     (y ∈ fpf-dom-list(da2))
| | .     & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | . 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| | .    has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:....
| | .
| | .    (y ∈ fpf-dom-list(da2))
| | .    & has-src(i;y) & ltg = da-outlink-f(da2;y))
| | . 10. y : Knd
| | . 11. y ∈ dom(da1 ⊕ da2)
| | . 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
| | . 13. has-src(i;y)
| | . 14. ltg = da-outlink-f(da1 ⊕ da2;y)
| | . 15. y ∈ dom(da1 ⊕ da2)
| | . 16. y ∈ dom(da1)
| | . 17. y1 : {k:Knd| k ∈ dom(da1) }
| | . 18. has-src(i;y1)
| | . ⊢  y1 ∈ {k:Knd| k ∈ dom(da1) & isrcv(k) }
| | .   by Analyze -2 THEN Analyze
| | .      THEN
| | .      RWO
| | .      Thm* ∀i:Id, k:Knd.
| | .           has-src(i;k) ⟺ isrcv(k) & source(lnk(k)) = i
| | .      {assert_DASH_has_DASH_src}
| | .      -1
| | .      THEN
| | .      ExRepD
| |
| .\
| . .....subterm .....
| . 1. ltg : IdLnk×Id×Type
| . 2. i : Id
| . 3. da1 : k:Knd fp-> Type
| . 4. da2 : k:Knd fp-> Type
| . 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
| .    has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
| . 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| .    has-src(i;k);fpf-dom-list(da1)))
| .    ⇒
| .    (∃y:{k:Knd| k ∈ dom(da1) }.
| .     (y ∈ fpf-dom-list(da1))
| .     & has-src(i;y) & ltg = da-outlink-f(da1;y))
| . 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
| .    has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:{k:Knd| k ∈ dom(da1) }.
| .
| .    (y ∈ fpf-dom-list(da1))
| .    & has-src(i;y) & ltg = da-outlink-f(da1;y))
| . 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| .    has-src(i;k);fpf-dom-list(da2)))
```

```
| .    ⇒
| .    (∃y:{k:Knd| k ∈ dom(da2) }.
| .     (y ∈ fpf-dom-list(da2))
| .     & has-src(i;y) & ltg = da-outlink-f(da2;y))
| . 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
| .    has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:{k:Knd| k ∈ dom(da2) }.
| .
| .    (y ∈ fpf-dom-list(da2))
| .    & has-src(i;y) & ltg = da-outlink-f(da2;y))
| . 10. y : Knd
| . 11. y ∈ dom(da1 ⊕ da2)
| . 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
| . 13. has-src(i;y)
| . 14. ltg = da-outlink-f(da1 ⊕ da2;y)
| . 15. y ∈ dom(da1 ⊕ da2)
| . 16. y ∈ dom(da1)
| . ⊢  mapfilter(λk.da-outlink-f(da2;k);λk.
| .    has-src(i;k);fpf-dom-list(da2))
| .    ∈ (IdLnk×Id×Type) List  by Fold ‘da-outlinks‘ 0
|
.\
. 1. ltg : IdLnk×Id×Type
. 2. i : Id
. 3. da1 : k:Knd fp-> Type
. 4. da2 : k:Knd fp-> Type
. 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
. |  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
. 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. |  has-src(i;k);fpf-dom-list(da1)))
. |  ⇐⇒
. |  (∃y:{k:Knd| k ∈ dom(da1) }.
. |   (y ∈ fpf-dom-list(da1))
. |   & has-src(i;y) & ltg = da-outlink-f(da1;y))
. 7. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. |  has-src(i;k);fpf-dom-list(da2)))
. |  ⇐⇒
. |  (∃y:{k:Knd| k ∈ dom(da2) }.
. |   (y ∈ fpf-dom-list(da2))
. |   & has-src(i;y) & ltg = da-outlink-f(da2;y))
. 8. y : Knd
. 9. y ∈ dom(da1 ⊕ da2)  [not for witness]
. 10. (y ∈ fpf-dom-list(da1 ⊕ da2))
. 11. has-src(i;y)
. 12. ltg = da-outlink-f(da1 ⊕ da2;y)
. 13. y ∈ dom(da1 ⊕ da2)
. 14. ¬y ∈ dom(da1)
. ⊢  (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. |  has-src(i;k);fpf-dom-list(da1)))
. |  ∨ (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. |    has-src(i;k);fpf-dom-list(da2)))  by OrRight
. |
. |\
. | 1. ltg : IdLnk×Id×Type
. | 2. i : Id
. | 3. da1 : k:Knd fp-> Type
```

```
. | 4. da2 : k:Knd fp-> Type
. | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
. | |  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
. | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | |  has-src(i;k);fpf-dom-list(da1)))
. | |  ⇒
. | |  (∃y:{k:Knd| k ∈ dom(da1) }.
. | |   (y ∈ fpf-dom-list(da1))
. | |   & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | |  has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:{k:Knd| k ∈ dom(da1) }.
. | |
. | |  (y ∈ fpf-dom-list(da1))
. | |  & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | |  has-src(i;k);fpf-dom-list(da2)))
. | |  ⇒
. | |  (∃y:{k:Knd| k ∈ dom(da2) }.
. | |   (y ∈ fpf-dom-list(da2))
. | |   & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | |  has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:{k:Knd| k ∈ dom(da2) }.
. | |
. | |  (y ∈ fpf-dom-list(da2))
. | |  & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | 10. y : Knd
. | 11. y ∈ dom(da1 ⊕ da2)  [not for witness]
. | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
. | 13. has-src(i;y)
. | 14. ltg = da-outlink-f(da1 ⊕ da2;y)
. | 15. y ∈ dom(da1 ⊕ da2)
. | 16. ¬y ∈ dom(da1)
. | ⊢  (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | |  has-src(i;k);fpf-dom-list(da2)))
. | | by BackThruSomeHyp THEN InstConcl [y]
. | |
. | |\
. | | .....wf .....
. | | 1. ltg : IdLnk×Id×Type
. | | 2. i : Id
. | | 3. da1 : k:Knd fp-> Type
. | | 4. da2 : k:Knd fp-> Type
. | | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
. | | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1)))
. | | |  ⇒
. | | |  (∃y:{k:Knd| k ∈ dom(da1) }.
. | | |   (y ∈ fpf-dom-list(da1))
. | | |   & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:....
. | | |
. | | |  (y ∈ fpf-dom-list(da1))
. | | |  & has-src(i;y) & ltg = da-outlink-f(da1;y))
```

```
. | | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da2)))
. | | |  ⇒
. | | |  (∃y:{k:Knd| k ∈ dom(da2) }.
. | | |   (y ∈ fpf-dom-list(da2))
. | | |   & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:....
. | | |
. | | |  (y ∈ fpf-dom-list(da2))
. | | |  & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 10. y : Knd
. | | 11. y ∈ dom(da1 ⊕ da2)
. | | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
. | | 13. has-src(i;y)
. | | 14. ltg = da-outlink-f(da1 ⊕ da2;y)
. | | 15. y ∈ dom(da1 ⊕ da2)
. | | 16. ¬y ∈ dom(da1)
. | | ⊢  y ∈ {k:Knd| k ∈ dom(da2) }  by Analyze
. | | |
. | | .....set predicate .....
. | | 1. ltg : IdLnk×Id×Type
. | | 2. i : Id
. | | 3. da1 : k:Knd fp-> Type
. | | 4. da2 : k:Knd fp-> Type
. | | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
. | |    has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
. | | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | |    has-src(i;k);fpf-dom-list(da1)))
. | |    ⇒
. | |    (∃y:{k:Knd| k ∈ dom(da1) }.
. | |     (y ∈ fpf-dom-list(da1))
. | |     & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | |    has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:....
. | |
. | |    (y ∈ fpf-dom-list(da1))
. | |    & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | |    has-src(i;k);fpf-dom-list(da2)))
. | |    ⇒
. | |    (∃y:{k:Knd| k ∈ dom(da2) }.
. | |     (y ∈ fpf-dom-list(da2))
. | |     & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | |    has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:....
. | |
. | |    (y ∈ fpf-dom-list(da2))
. | |    & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 10. y : Knd
. | | 11. y ∈ dom(da1 ⊕ da2)
. | | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
. | | 13. has-src(i;y)
. | | 14. ltg = da-outlink-f(da1 ⊕ da2;y)
. | | 15. y ∈ dom(da1 ⊕ da2)
```

```
. | | 16. ¬y ∈ dom(da1)
. | | ⊢  y ∈ dom(da2)
. | |   by RWO
. | |      Thm* ∀B:(A→Type), eq:EqDecider(A), f,g:a:A fp-> B(a), x:A.
. | |           x ∈ dom(f ⊕ g) ⟺ x ∈ dom(f) ∨ x ∈ dom(g)
. | |      {fpf_DASH_join_DASH_dom}
. | |      -2
. | |      THEN
. | |      Analyze -2
. | |
. | |\
. | | 1. ltg : IdLnk×Id×Type
. | | 2. i : Id
. | | 3. da1 : k:Knd fp-> Type
. | | 4. da2 : k:Knd fp-> Type
. | | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
. | | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1)))
. | | |  ⇒
. | | |  (∃y:{k:Knd| k ∈ dom(da1) }.
. | | |   (y ∈ fpf-dom-list(da1))
. | | |   & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:....
. | | |
. | | |  (y ∈ fpf-dom-list(da1))
. | | |  & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da2)))
. | | |  ⇒
. | | |  (∃y:{k:Knd| k ∈ dom(da2) }.
. | | |   (y ∈ fpf-dom-list(da2))
. | | |   & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:....
. | | |
. | | |  (y ∈ fpf-dom-list(da2))
. | | |  & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 10. y : Knd
. | | 11. y ∈ dom(da1 ⊕ da2)  [not for witness]
. | | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
. | | 13. has-src(i;y)
. | | 14. ltg = da-outlink-f(da1 ⊕ da2;y)
. | | 15. y ∈ dom(da1 ⊕ da2)
. | | 16. ¬y ∈ dom(da1)
. | | ⊢  (y ∈ fpf-dom-list(da2))
. | | | by RWO
. | | |    Thm* ∀B:(A→Type), eq:EqDecider(A), f,g:a:A fp-> B(a), x:A.
. | | |         x ∈ dom(f ⊕ g) ⟺ x ∈ dom(f) ∨ x ∈ dom(g)
. | | |    {fpf_DASH_join_DASH_dom}
. | | |    -2
. | | |    THEN
. | | |    Analyze -2
. | | |    THEN
```

```
. | | |    Thin -1
. | | |
. | | 1. ltg : IdLnk×Id×Type
. | | 2. i : Id
. | | 3. da1 : k:Knd fp-> Type
. | | 4. da2 : k:Knd fp-> Type
. | | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
. | | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1)))
. | | |  ⇒
. | | |  (∃y:{k:Knd| k ∈ dom(da1) }.
. | | |   (y ∈ fpf-dom-list(da1))
. | | |   & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:....
. | | |
. | | |  (y ∈ fpf-dom-list(da1))
. | | |  & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da2)))
. | | |  ⇒
. | | |  (∃y:{k:Knd| k ∈ dom(da2) }.
. | | |   (y ∈ fpf-dom-list(da2))
. | | |   & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:....
. | | |
. | | |  (y ∈ fpf-dom-list(da2))
. | | |  & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 10. y : Knd
. | | 11. y ∈ dom(da1 ⊕ da2)  [not for witness]
. | | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
. | | 13. has-src(i;y)
. | | 14. ltg = da-outlink-f(da1 ⊕ da2;y)
. | | 15. y ∈ dom(da2)
. | | ⊢  (y ∈ fpf-dom-list(da2))
. | | | by AssertBY (fpf-dom-list(da2) ∈ {k:Knd| k ∈ dom(da2) } List)
. | | |    Auto
. | | |    THEN
. | | |    DupHyp -2
. | | |    THEN
. | | |    Unfold ‘fpf-dom‘ -1
. | | |    THEN
. | | |    Fold ‘fpf-dom-list‘ -1
. | | |    THEN
. | | |    RWO
. | | |    Thm* ∀eq:EqDecider(A), L:A List, x:A.
. | | |         deq-member(eq;x;L) ⟺ (x ∈ L)
. | | |    {assert_DASH_deq_DASH_member}
. | | |    -1
. | | |
. | | |\
. | | | .....wf .....
. | | | 1. ltg : IdLnk×Id×Type
```

```
. | | | 2. i : Id
. | | | 3. da1 : k:Knd fp-> Type
. | | | 4. da2 : k:Knd fp-> Type
. | | | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
. | | |    has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
. | | | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | |    has-src(i;k);fpf-dom-list(da1)))
. | | |    ⇒
. | | |    (∃y:{k:Knd| k ∈ dom(da1) }.
. | | |     (y ∈ fpf-dom-list(da1))
. | | |     & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | |    has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:{k:Knd
. | | |                                          | k ∈ dom(da1) }.
. | | |
. | | |    (y ∈ fpf-dom-list(da1))
. | | |    & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | |    has-src(i;k);fpf-dom-list(da2)))
. | | |    ⇒
. | | |    (∃y:{k:Knd| k ∈ dom(da2) }.
. | | |     (y ∈ fpf-dom-list(da2))
. | | |     & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | |    has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:{k:Knd
. | | |                                          | k ∈ dom(da2) }.
. | | |
. | | |    (y ∈ fpf-dom-list(da2))
. | | |    & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | | 10. y : Knd
. | | | 11. y ∈ dom(da1 ⊕ da2)
. | | | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
. | | | 13. has-src(i;y)
. | | | 14. ltg = da-outlink-f(da1 ⊕ da2;y)
. | | | 15. y ∈ dom(da2)
. | | | 16. fpf-dom-list(da2) ∈ {k:Knd| k ∈ dom(da2) } List
. | | | 17. deq-member(KindDeq;y;fpf-dom-list(da2))
. | | | ⊢  fpf-dom-list(da2) ∈ Knd List
. | | |   by DoSubsume THEN UseSubtypeRelLemmas THEN Analyze 0
. | | |
. | | .\
. | | . 1. ltg : IdLnk×Id×Type
. | | . 2. i : Id
. | | . 3. da1 : k:Knd fp-> Type
. | | . 4. da2 : k:Knd fp-> Type
. | | . 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
. | | .    has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
. | | . 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | .    has-src(i;k);fpf-dom-list(da1)))
. | | .    ⇒
. | | .    (∃y:{k:Knd| k ∈ dom(da1) }.
. | | .     (y ∈ fpf-dom-list(da1))
. | | .     & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | . 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | .    has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:{k:Knd
```

```
. | | .                                             | k ∈ dom(da1) }.
. | | .
. | | .    (y ∈ fpf-dom-list(da1))
. | | .    & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | . 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | .    has-src(i;k);fpf-dom-list(da2)))
. | | .    ⇒
. | | .    (∃y:{k:Knd| k ∈ dom(da2) }.
. | | .     (y ∈ fpf-dom-list(da2))
. | | .     & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | . 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | .    has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:{k:Knd
. | | .                                             | k ∈ dom(da2) }.
. | | .
. | | .    (y ∈ fpf-dom-list(da2))
. | | .    & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | . 10. y : Knd
. | | . 11. y ∈ dom(da1 ⊕ da2)  [not for witness]
. | | . 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
. | | . 13. has-src(i;y)
. | | . 14. ltg = da-outlink-f(da1 ⊕ da2;y)
. | | . 15. y ∈ dom(da2)
. | | . 16. fpf-dom-list(da2) ∈ {k:Knd| k ∈ dom(da2) } List
. | | . 17. (y ∈ fpf-dom-list(da2))
. | | . ⊢  (y ∈ fpf-dom-list(da2))
. | | .   by RepeatFor 2 ParallelLast THEN ExRepD THEN Analyze
. | |
. | |\
. | | 1. ltg : IdLnk×Id×Type
. | | 2. i : Id
. | | 3. da1 : k:Knd fp-> Type
. | | 4. da2 : k:Knd fp-> Type
. | | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
. | | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1)))
. | | |  ⇒
. | | |  (∃y:{k:Knd| k ∈ dom(da1) }.
. | | |   (y ∈ fpf-dom-list(da1))
. | | |   & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:....
. | | |
. | | |  (y ∈ fpf-dom-list(da1))
. | | |  & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da2)))
. | | |  ⇒
. | | |  (∃y:{k:Knd| k ∈ dom(da2) }.
. | | |   (y ∈ fpf-dom-list(da2))
. | | |   & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:....
. | | |
. | | |  (y ∈ fpf-dom-list(da2))
```

```
. | | |  & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 10. y : Knd
. | | 11. y ∈ dom(da1 ⊕ da2)
. | | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
. | | 13. has-src(i;y)
. | | 14. ltg = da-outlink-f(da1 ⊕ da2;y)
. | | 15. y ∈ dom(da1 ⊕ da2)
. | | 16. ¬y ∈ dom(da1)
. | | ⊢  ltg = da-outlink-f(da2;y)
. | | | by Assert (da-outlink-f(da2;y) = da-outlink-f(da1 ⊕ da2;y))
. | | |
. | | .....assertion .....
. | | 1. ltg : IdLnk×Id×Type
. | | 2. i : Id
. | | 3. da1 : k:Knd fp-> Type
. | | 4. da2 : k:Knd fp-> Type
. | | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
. | | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1)))
. | | |  ⇒
. | | |  (∃y:{k:Knd| k ∈ dom(da1) }.
. | | |   (y ∈ fpf-dom-list(da1))
. | | |   & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:....
. | | |
. | | |  (y ∈ fpf-dom-list(da1))
. | | |  & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da2)))
. | | |  ⇒
. | | |  (∃y:{k:Knd| k ∈ dom(da2) }.
. | | |   (y ∈ fpf-dom-list(da2))
. | | |   & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:....
. | | |
. | | |  (y ∈ fpf-dom-list(da2))
. | | |  & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 10. y : Knd
. | | 11. y ∈ dom(da1 ⊕ da2)
. | | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
. | | 13. has-src(i;y)
. | | 14. ltg = da-outlink-f(da1 ⊕ da2;y)
. | | 15. y ∈ dom(da1 ⊕ da2)
. | | 16. ¬y ∈ dom(da1)
. | | ⊢  da-outlink-f(da2;y) = da-outlink-f(da1 ⊕ da2;y)
. | | | by RWO
. | | |    Thm* ∀i:Id, k:Knd.
. | | |         has-src(i;k) ⟺ isrcv(k) & source(lnk(k)) = i
. | | |    {assert_DASH_has_DASH_src}
. | | |    -4
. | | |    THEN
. | | |    ExRepD
```

```
. | | |    THEN
. | | |    Unfold `da-outlink-f` 0
. | | |    THEN
. | | |    Analyze
. | | |    THEN
. | | |    Analyze
. | | |
. | | .....subterm .....
. | | 1. ltg : IdLnk×Id×Type
. | | 2. i : Id
. | | 3. da1 : k:Knd fp-> Type
. | | 4. da2 : k:Knd fp-> Type
. | | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
. | | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1)))
. | | |  ⇒
. | | |  (∃y:{k:Knd| k ∈ dom(da1) }.
. | | |   (y ∈ fpf-dom-list(da1))
. | | |   & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:....
. | | |
. | | |  (y ∈ fpf-dom-list(da1))
. | | |  & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da2)))
. | | |  ⇒
. | | |  (∃y:{k:Knd| k ∈ dom(da2) }.
. | | |   (y ∈ fpf-dom-list(da2))
. | | |   & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | | |  has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:....
. | | |
. | | |  (y ∈ fpf-dom-list(da2))
. | | |  & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 10. y : Knd
. | | 11. y ∈ dom(da1 ⊕ da2)
. | | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
. | | 13. isrcv(y)
. | | 14. source(lnk(y)) = i
. | | 15. ltg = da-outlink-f(da1 ⊕ da2;y)
. | | 16. y ∈ dom(da1 ⊕ da2)
. | | 17. ¬y ∈ dom(da1)
. | | ⊢  da2(y) = da1 ⊕ da2(y)
. | | | by RWO
. | | |    Thm* ∀eq:EqDecider(A), f:a:A fp-> Top, g:Top, x:A.
. | | |         f ⊕ g(x) ~ if x ∈ dom(f)→ f(x) else g(x) fi
. | | |    {fpf_DASH_join_DASH_ap_DASH_sq}
. | | |    0
. | | |    THENA
. | | |    (Auto THEN All Thin)
. | | |    THEN
. | | |    SplitOnConclITE
. | | |    THENA
```

```
. | | |    (Auto THEN All Thin)
. | | |
. | | 1. ltg : IdLnk×Id×Type
. | | 2. i : Id
. | | 3. da1 : k:Knd fp-> Type
. | | 4. da2 : k:Knd fp-> Type
. | | 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
. | |    has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
. | | 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | |    has-src(i;k);fpf-dom-list(da1)))
. | |    ⇒
. | |    (∃y:{k:Knd| k ∈ dom(da1) }.
. | |     (y ∈ fpf-dom-list(da1))
. | |     & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | |    has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:....
. | |
. | |    (y ∈ fpf-dom-list(da1))
. | |    & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | | 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | |    has-src(i;k);fpf-dom-list(da2)))
. | |    ⇒
. | |    (∃y:{k:Knd| k ∈ dom(da2) }.
. | |     (y ∈ fpf-dom-list(da2))
. | |     & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | |    has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:....
. | |
. | |    (y ∈ fpf-dom-list(da2))
. | |    & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | | 10. y : Knd
. | | 11. y ∈ dom(da1 ⊕ da2)
. | | 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
. | | 13. isrcv(y)
. | | 14. source(lnk(y)) = i
. | | 15. ltg = da-outlink-f(da1 ⊕ da2;y)
. | | 16. y ∈ dom(da1 ⊕ da2)
. | | 17. ¬y ∈ dom(da1)
. | | 18. ¬y ∈ dom(da1)
. | | ⊢  y ∈ dom(da2)
. | |   by RWO
. | |      Thm* ∀B:(A→Type), eq:EqDecider(A), f,g:a:A fp-> B(a), x:A.
. | |           x ∈ dom(f ⊕ g) ⟺ x ∈ dom(f) ∨ x ∈ dom(g)
. | |      {fpf_DASH_join_DASH_dom}
. | |      -3
. | |      THEN
. | |      Analyze -3
. | |
. | .\
. | . .....subterm .....
. | . 1. ltg : IdLnk×Id×Type
. | . 2. i : Id
. | . 3. da1 : k:Knd fp-> Type
. | . 4. da2 : k:Knd fp-> Type
. | . 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
```

```
. | .    has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
. | . 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | .    has-src(i;k);fpf-dom-list(da1)))
. | .    ⇒
. | .    (∃y:{k:Knd| k ∈ dom(da1) }.
. | .     (y ∈ fpf-dom-list(da1))
. | .     & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | . 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. | .    has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:....
. | .
. | .    (y ∈ fpf-dom-list(da1))
. | .    & has-src(i;y) & ltg = da-outlink-f(da1;y))
. | . 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | .    has-src(i;k);fpf-dom-list(da2)))
. | .    ⇒
. | .    (∃y:{k:Knd| k ∈ dom(da2) }.
. | .     (y ∈ fpf-dom-list(da2))
. | .     & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | . 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. | .    has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:....
. | .
. | .    (y ∈ fpf-dom-list(da2))
. | .    & has-src(i;y) & ltg = da-outlink-f(da2;y))
. | . 10. y : Knd
. | . 11. y ∈ dom(da1 ⊕ da2)
. | . 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
. | . 13. has-src(i;y)
. | . 14. ltg = da-outlink-f(da1 ⊕ da2;y)
. | . 15. y ∈ dom(da1 ⊕ da2)
. | . 16. ¬y ∈ dom(da1)
. | . 17. y1 : {k:Knd| k ∈ dom(da2) }
. | . 18. has-src(i;y1)
. | . ⊢  y1 ∈ {k:Knd| k ∈ dom(da2) & isrcv(k) }
. | .   by Analyze -2
. | .      THEN
. | .      RWO
. | .      Thm* ∀i:Id, k:Knd.
. | .           has-src(i;k) ⟺ isrcv(k) & source(lnk(k)) = i
. | .      {assert_DASH_has_DASH_src}
. | .      -1
. | .      THEN
. | .      ExRepD
. |
. .\
. . .....subterm .....
. . 1. ltg : IdLnk×Id×Type
. . 2. i : Id
. . 3. da1 : k:Knd fp-> Type
. . 4. da2 : k:Knd fp-> Type
. . 5. (ltg ∈ mapfilter(λk.da-outlink-f(da1 ⊕ da2;k);λk.
. .    has-src(i;k);fpf-dom-list(da1 ⊕ da2)))
. . 6. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. .    has-src(i;k);fpf-dom-list(da1)))
. .    ⇒
. .    (∃y:{k:Knd| k ∈ dom(da1) }.
```

```
. .     (y ∈ fpf-dom-list(da1))
. .     & has-src(i;y) & ltg = da-outlink-f(da1;y))
. . 7. (ltg ∈ mapfilter(λk.da-outlink-f(da1;k);λk.
. .    has-src(i;k);fpf-dom-list(da1))) ⇐ (∃y:{k:Knd| k ∈ dom(da1) }.
. .
. .    (y ∈ fpf-dom-list(da1))
. .    & has-src(i;y) & ltg = da-outlink-f(da1;y))
. . 8. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. .    has-src(i;k);fpf-dom-list(da2)))
. .    ⇒
. .    (∃y:{k:Knd| k ∈ dom(da2) }.
. .     (y ∈ fpf-dom-list(da2))
. .     & has-src(i;y) & ltg = da-outlink-f(da2;y))
. . 9. (ltg ∈ mapfilter(λk.da-outlink-f(da2;k);λk.
. .    has-src(i;k);fpf-dom-list(da2))) ⇐ (∃y:{k:Knd| k ∈ dom(da2) }.
. .
. .    (y ∈ fpf-dom-list(da2))
. .    & has-src(i;y) & ltg = da-outlink-f(da2;y))
. . 10. y : Knd
. . 11. y ∈ dom(da1 ⊕ da2)
. . 12. (y ∈ fpf-dom-list(da1 ⊕ da2))
. . 13. has-src(i;y)
. . 14. ltg = da-outlink-f(da1 ⊕ da2;y)
. . 15. y ∈ dom(da1 ⊕ da2)
. . 16. ¬y ∈ dom(da1)
. . ⊢  mapfilter(λk.da-outlink-f(da1;k);λk.
. .    has-src(i;k);fpf-dom-list(da1))
. .    ∈ (IdLnk×Id×Type) List  by Fold ‘da-outlinks‘ 0
END
```