```
vertex-subset-properties  THM

⊢  For any graph
|  ∀the_obj:GraphObject(the_graph), P:(V→𝔹).
|  no_repeats(V;vertex-subset(the_obj;x.P(x)))
|  & (∀x:V. (x ∈ vertex-subset(the_obj;x.P(x))) ⟺ P(x))
| by UnivCD
|    THEN
|    (Inst Thm: graphobj_DASH_properties [the_graph;the_obj])
|    THEN
|    (Analyze -1)
|    THEN
|    (Analyze -1)
|
1. the_graph : Graph
2. the_obj : GraphObject(the_graph)
3. P : Vertices(the_graph)→𝔹
4. ∀x,y:Vertices(the_graph). the_obj.eq(x,y) ⟺ x = y
5. ∀T:Type, s:T, x:Vertices(the_graph), f:(T→Vertices(the_graph)→T).
|  ∃L:Vertices(the_graph) List.
|  (∀y:Vertices(the_graph). x-the_graph->y ⟺ (y ∈ L))
|  & the_obj.eacc(f,s,x) = list_accum(s',x'.f(s',x');s;L)
6. ∀T:Type, s:T, f:(T→Vertices(the_graph)→T).
|  ∃L:Vertices(the_graph) List.
|  no_repeats(Vertices(the_graph);L)
|  & (∀y:Vertices(the_graph). (y ∈ L))
|  & the_obj.vacc(f,s) = list_accum(s',x'.f(s',x');s;L)
⊢  no_repeats(Vertices(the_graph);vertex-subset(the_obj;x.P(x)))
|  & (∀x:Vertices(the_graph).
|      (x ∈ vertex-subset(the_obj;x.P(x))) ⟺ P(x))
| by (Unfold `vertex-subset` 0)
|    THEN
|    (InstHyp
|     [Vertices(the_graph) List;nil;λl,x. if P(x)→ [x / l] else l fi]
|     -1)
|    THEN
|    ExRepD
|    THEN
|    (HypSubst -1 0)
|    THEN
|    (Thin -1)
|
1. the_graph : Graph
2. the_obj : GraphObject(the_graph)
3. P : Vertices(the_graph)→𝔹
4. ∀x,y:Vertices(the_graph). the_obj.eq(x,y) ⟺ x = y
5. ∀T:Type, s:T, x:Vertices(the_graph), f:(T→Vertices(the_graph)→T).
|  ∃L:Vertices(the_graph) List.
|  (∀y:Vertices(the_graph). x-the_graph->y ⟺ (y ∈ L))
|  & the_obj.eacc(f,s,x) = list_accum(s',x'.f(s',x');s;L)
6. ∀T:Type, s:T, f:(T→Vertices(the_graph)→T).
|  ∃L:Vertices(the_graph) List.
|  no_repeats(Vertices(the_graph);L)
|  & (∀y:Vertices(the_graph). (y ∈ L))
|  & the_obj.vacc(f,s) = list_accum(s',x'.f(s',x');s;L)
```

```
7. L : Vertices(the_graph) List
8. no_repeats(Vertices(the_graph);L)
9. ∀y:Vertices(the_graph). (y ∈ L)
⊢  no_repeats(Vertices(the_graph);list_accum(s',x'.(λl,x.
|  if P(x)→ [x / l] else l fi)(s',x');nil;L))
|  & (∀x:Vertices(the_graph).
|     (x ∈ list_accum(s',x'.(λl,x. if P(x)→ [x / l]
|                                else l fi)(s',x');nil;L))
|     ⟺
|     P(x))
| by (((All ReduceSOAps)
|      THEN
|      (RWO
|       Thm* ∀P:(T→𝔹), L2,L1:T List.
|            list_accum(l,x.if P(x)→ [x / l]
|                           else l fi;L1;L2) ~ (rev(filter(P;L2))
|            @ L1)
|       {filter_list_accum}
|       0))
|     THEN
|     (RWO Thm* ∀as:T List. (as @ nil) = as    {append_back_nil} 0)
|     THEN
|     (RWO
|      Thm* ∀x:T, L:T List. (x ∈ rev(L)) ⟺ (x ∈ L)
|      {member_reverse}
|      0)
|     THEN
|     (RWO
|      Thm* ∀P:(T→𝔹), L:T List, x:T.
|           (x ∈ filter(P;L)) ⟺ (x ∈ L) & P(x)
|      {member_filter}
|      0)
|     THEN
|     (RWO
|      Thm* ∀L:T List. no_repeats(T;rev(L)) ⟺ no_repeats(T;L)
|      {no_repeats_reverse}
|      0))
|    THEN
|    EasyHyp
|
1. the_graph : Graph
2. the_obj : GraphObject(the_graph)
3. P : Vertices(the_graph)→𝔹
4. ∀x,y:Vertices(the_graph). the_obj.eq(x,y) ⟺ x = y
5. ∀T:Type, s:T, x:Vertices(the_graph), f:(T→Vertices(the_graph)→T).
   ∃L:Vertices(the_graph) List.
   (∀y:Vertices(the_graph). x-the_graph->y ⟺ (y ∈ L))

   & the_obj.eacc(f,s,x) = list_accum(s',x'.f(s',x');s;L)
6. ∀T:Type, s:T, f:(T→Vertices(the_graph)→T).
   ∃L:Vertices(the_graph) List.
   no_repeats(Vertices(the_graph);L)
   & (∀y:Vertices(the_graph). (y ∈ L))
   & the_obj.vacc(f,s) = list_accum(s',x'.f(s',x');s;L)
7. L : Vertices(the_graph) List
8. no_repeats(Vertices(the_graph);L)
```

```
9. ∀y:Vertices(the_graph). (y ∈ L)
⊢  no_repeats(Vertices(the_graph);filter(P;L))
  by BackThru
     Thm* ∀P:(T→𝔹), l:T List.
          no_repeats(T;l) ⇒ no_repeats(T;filter(P;l))
     {no_repeats_filter}
END
```