similarity and fixed point sets

Johan G. F. Belinfante
2010 May 10

```
In[1]:= SetDirectory["l:"]; << goedel.10may09a;<< tools.m

        :Package Title: goedel.10may09a               2010 May 9 at 9:30 a.m.

        It is now:  2010 May 10 at 15:25

        Loading Simplification Rules

        TOOLS.M                                    Revised 2010 February 26

        weightlimit = 40
```

summary

Similar relations have equipollent fixed point sets.

derivation

Lemma.

```
In[2]:= SubstTest[implies, FUNCTION[z], equal[fix[composite[inverse[z], y, z]],
           image[inverse[z], fix[y]]], z → inverse[x]] // Reverse

Out[2]= or[equal[fix[composite[x, y, inverse[x]]], image[x, fix[y]]],
          not[FUNCTION[inverse[x]]] == True

In[3]:= or[equal[fix[composite[x_, y_, inverse[x_]]], image[x_, fix[y_]]],
          not[FUNCTION[inverse[x_]]] := True
```

Lemma.

```
In[4]:= (Map[not, SubstTest[and, implies[p0, p2], implies[p1, p3],
             implies[and[p0, p2, p3], p4], not[implies[and[p0, p1], p4]],
             {p0 → and[member[t, BIJ], equal[y, composite[t, x, inverse[t]]]],
              p1 -> subclass[x, cartsq[domain[t]]],
              p2 → equal[fix[y], image[t, fix[x]]], p3 → subclass[fix[x], domain[t]],
              p4 → member[pair[fix[x], fix[y]], Q]}]] // Reverse) /. t → setpart[z]

Out[4]= or[member[pair[fix[x], fix[y]], Q],
          not[equal[y, composite[setpart[z], x, inverse[setpart[z]]]]],
          not[FUNCTION[inverse[setpart[z]]]], not[FUNCTION[setpart[z]]],
          not[subclass[x, cart[domain[setpart[z]], domain[setpart[z]]]]] == True
```

```
In[5]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[6]:= member[pair[u, v], composite[inverse[IMAGE[inverse[DUP]]], x]] // AssertTest

Out[6]= member[pair[u, v], composite[inverse[IMAGE[inverse[DUP]]], x]] ==
         and[member[u, V], member[v, V], member[pair[u, fix[v]], x]]

In[7]:= member[pair[u_, v_], composite[inverse[IMAGE[inverse[DUP]]], x_]] :=
         and[member[u, V], member[v, V], member[pair[u, fix[v]], x]]
```

Theorem. Eliminating all variables yields an inclusion.

```
In[8]:= Map[empty[composite[#, inverse[SECOND]]] &,
         Map[composite[complement[#], id[cart[V, V]]] &, SubstTest[class, pair[pair[t, x], y],
           or[not[member[pair[pair[setpart[t], setpart[x]], setpart[y]], u]],
            member[pair[setpart[x], setpart[y]], v]],
           {u -> composite[IMG, cross[composite[CROSS, DUP, id[BIJ]], Id], id[composite[
                inverse[S], CART, DUP, IMAGE[FIRST]]]], v → composite[inverse[FIX], Q, FIX]}]]]

Out[8]= subclass[composite[IMAGE[inverse[DUP]], SIMILAR, inverse[IMAGE[inverse[DUP]]]], Q] ==
         True

In[9]:= % /. Equal → SetDelayed
```

A reverse inclusion will now be derived.

Lemma. Identity functions on equipollent sets are similar.

```
In[10]:= Map[subclass[#, SIMILAR] &,
           ImageComp[cross[IDP, IDP], inverse[cross[IDP, IDP]], SIMILAR]] // Reverse

Out[10]= subclass[composite[IMAGE[DUP], Q, inverse[IMAGE[DUP]]], SIMILAR] == True

In[11]:= subclass[composite[IMAGE[DUP], Q, inverse[IMAGE[DUP]]], SIMILAR] := True
```

Lemma. Inclusion in the reverse direction.

```
In[12]:= SubstTest[implies, subclass[u, v],
           subclass[image[t, u], image[t, v]], {t → cross[FIX, FIX],
            u → composite[IMAGE[DUP], Q, inverse[IMAGE[DUP]]], v → SIMILAR}] // Reverse

Out[12]= subclass[Q,
           composite[IMAGE[inverse[DUP]], SIMILAR, inverse[IMAGE[inverse[DUP]]]]] == True

In[13]:= % /. Equal → SetDelayed
```

Combining the two inclusions in opposite directions yields an equation.

Theorem.

```
In[14]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u → Q, v → composite[IMAGE[inverse[DUP]], SIMILAR, inverse[IMAGE[inverse[DUP]]]]}]

Out[14]= equal[Q, composite[IMAGE[inverse[DUP]], SIMILAR, inverse[IMAGE[inverse[DUP]]]]] == True

In[15]:= composite[IMAGE[inverse[DUP]], SIMILAR, inverse[IMAGE[inverse[DUP]]]] := Q
```

Corollary. Similar relations have equipollent fixed point sets.

```
In[16]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
            {t → cross[FIX, FIX], u → set[PAIR[x, y]], v → SIMILAR}] // Reverse // MapNotNot

Out[16]= or[member[pair[fix[x], fix[y]], Q], not[member[pair[x, y], SIMILAR]]] == True

In[17]:= or[member[pair[fix[x_], fix[y_]], Q], not[member[pair[x_, y_], SIMILAR]]] := True
```

Corollary. A relation similar to an irreflexive relation is irreflexive.

```
In[20]:= Map[not, SubstTest[and, implies[p1, p3], implies[and[p2, p3], p4],
            not[implies[and[p1, p2], p4]], {p1 → member[pair[x, y], SIMILAR], p2 → empty[fix[x]],
             p3 -> member[pair[fix[x], fix[y]], Q], p4 → empty[fix[y]]}]] // Reverse

Out[20]= or[equal[0, fix[y]], not[equal[0, fix[x]]], not[member[pair[x, y], SIMILAR]]] == True

In[21]:= or[equal[0, fix[y_]], not[equal[0, fix[x_]]],
           not[member[pair[x_, y_], SIMILAR]]] := True
```

Lemma. Simplification rule.

```
In[23]:= SubstTest[image, SIMILAR, intersection[t, P[cart[V, V]]],
          t → P[union[x, complement[cart[V, V]]]]]

Out[23]= image[SIMILAR, P[union[x, complement[cart[V, V]]]]] ==
          image[SIMILAR, P[composite[Id, x]]]

In[24]:= image[SIMILAR, P[union[x_, complement[cart[V, V]]]]] :=
          image[SIMILAR, P[composite[Id, x]]]
```

Lemma.

```
In[25]:= Map[empty[composite[Id, complement[#]]] &, SubstTest[class, pair[x, y],
           or[equal[0, fix[y]], not[equal[0, fix[x]]], not[member[pair[x, y], z]]], z → SIMILAR]]

Out[25]= equal[0, fix[U[image[SIMILAR, P[Di]]]]] == True

In[27]:= fix[U[image[SIMILAR, P[Di]]]] := 0
```

Theorem.

```
In[28]:= SubstTest[and, subclass[u, v], subclass[v, u], {u -> image[SIMILAR, P[Di]], v → P[Di]}]

Out[28]= equal[image[SIMILAR, P[Di]], P[Di]] == True

In[31]:= image[SIMILAR, P[Di]] := P[Di]
```