similarity and boundedness

Johan G. F. Belinfante
2010 May 13

```
In[1]:= SetDirectory["l:"]; << goedel.10may12a;<< tools.m
```

```
:Package Title: goedel.10may12a               2010 May 12 at 2:25 p.m.

It is now:  2010 May 13 at 11:6

Loading Simplification Rules

TOOLS.M                                    Revised 2010 February 26

weightlimit = 40
```

summary

If **x** and **y** are similar relations, then **domain[UB[x]]** and **domain[UB[y]]** are equipollent.

an equipollence theorem

Lemma. Application of an equipollence criterion to **domain[UB[x]]**.

```
In[2]:= Map[implies[member[x, y], #] &,
           SubstTest[or, member[pair[y, image[IMAGE[oopart[t]], y]], Q], not[member[y, V]],
            not[subclass[U[y], domain[oopart[t]]]], y → domain[UB[x]]] // Reverse // MapNotNot
```

```
Out[2]= or[member[pair[domain[UB[x]], image[IMAGE[oopart[t]], domain[UB[x]]]], Q],
          not[member[x, y]], not[subclass[domain[x], domain[oopart[t]]]]] == True
```

```
In[3]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

```
In[4]:= Map[not, SubstTest[and, implies[and[p0, p2], p3], not[implies[and[p0, p1], p3]],
           {p0 → member[x, y], p1 → subclass[x, cartsq[domain[oopart[t]]]],
            p2 -> subclass[domain[x], domain[oopart[t]]], p3 -> member[
              pair[domain[UB[x]], image[IMAGE[oopart[t]], domain[UB[x]]]], Q]}]] // Reverse
```

```
Out[4]= or[member[pair[domain[UB[x]], image[IMAGE[oopart[t]], domain[UB[x]]]], Q], not[
           member[x, y]], not[subclass[x, cart[domain[oopart[t]], domain[oopart[t]]]]]] == True
```

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

Lemma. Eliminate **oopart**.

```
In[6]:= SubstTest[implies,
          and[equal[t, oopart[z]], member[x, V], subclass[x, cart[domain[t], domain[t]]]],
          member[pair[domain[UB[x]], image[IMAGE[t], domain[UB[x]]]], Q], z → t] // Reverse

Out[6]= or[member[pair[domain[UB[x]], image[IMAGE[t], domain[UB[x]]]], Q],
          not[FUNCTION[t]], not[FUNCTION[inverse[t]]], not[member[x, V]],
          not[subclass[x, cart[domain[t], domain[t]]]]] == True

In[7]:= or[member[pair[domain[UB[x_]], image[IMAGE[t_], domain[UB[x_]]]], Q],
          not[FUNCTION[t_]], not[FUNCTION[inverse[t_]]], not[member[x_, V]],
          not[subclass[x_, cart[domain[t_], domain[t_]]]]] := True
```

The next step is to transfer sethood condition from **x** to **t**.

Lemma.

```
In[8]:= SubstTest[implies, and[subclass[x, t], member[t, V]],
          member[x, V], t → cartsq[domain[setpart[y]]]] // Reverse

Out[8]= or[member[x, V], not[subclass[x, cart[domain[setpart[y]], domain[setpart[y]]]]]] == True

In[9]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed

In[10]:= SubstTest[implies, equal[y, setpart[t]],
           or[member[x, V], not[subclass[x, cart[domain[y], domain[y]]]]], t → y] // Reverse

Out[10]= or[member[x, V], not[member[y, V]], not[subclass[x, cart[domain[y], domain[y]]]]] == True

In[11]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. Transfer sethood condition from **x** to **t**.

```
In[12]:= Map[not, SubstTest[and, implies[and[p0, p1, p2], p3],
            implies[and[p0, p1], p2], not[implies[and[p0, p1], p3]],
            {p0 → member[t, BIJ], p1 -> subclass[x, cart[domain[t], domain[t]]], p2 → member[x, V],
             p3 -> member[pair[domain[UB[x]], image[IMAGE[t], domain[UB[x]]]], Q]}]] // Reverse

Out[12]= or[member[pair[domain[UB[x]], image[IMAGE[t], domain[UB[x]]]], Q],
           not[FUNCTION[t]], not[FUNCTION[inverse[t]]], not[member[t, V]],
           not[subclass[x, cart[domain[t], domain[t]]]]] == True

In[13]:= (% /. {x → x_, t → t_}) /. Equal → SetDelayed
```

relating domain[UB[x]] to domain[UB[y]]

In this section a simple formula relates **domain[UB[x]]** to **domain[UB[y]]** when **y** is related to **x** by a similarity transformation.

Lemma.

```
In[20]:= SubstTest[implies, subclass[range[u], v], equal[domain[u], image[inverse[u], v]],
           u → composite[UB[x], id[complement[set[0]]]]] // Reverse

Out[20]= or[equal[intersection[complement[set[0]], domain[UB[x]]],
            intersection[complement[set[0]], image[inverse[UB[x]], v]]],
           not[subclass[range[x], v]]] == True

In[21]:= or[equal[intersection[complement[set[0]], domain[UB[x_]]],
            intersection[complement[set[0]], image[inverse[UB[x]], v_]]],
           not[subclass[range[x_], v_]]] := True
```

Lemma.

```
In[22]:= SubstTest[implies, equal[u, v], equal[union[set[0], u], union[set[0], v]],
           {u -> intersection[complement[set[0]], domain[UB[x]]],
            v -> intersection[complement[set[0]], image[inverse[UB[x]], v]]}] // Reverse

Out[22]= or[equal[domain[UB[x]], union[image[inverse[UB[x]], v], set[0]]],
           not[equal[intersection[complement[set[0]], domain[UB[x]]],
             intersection[complement[set[0]], image[inverse[UB[x]], v]]]]] == True

In[23]:= (% /. {x → x_, v → v_}) /. Equal → SetDelayed
```

Lemma.

```
In[24]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p3, p4],
            not[implies[p1, p4]], {p1 -> subclass[x, cartsq[domain[oopart[t]]]],
             p2 → subclass[range[x], domain[oopart[t]]],
             p3 → equal[intersection[complement[set[0]], domain[UB[x]]],
               intersection[complement[set[0]], image[inverse[UB[x]], domain[oopart[t]]]]],
             p4 -> equal[domain[UB[x]], union[image[inverse[UB[x]], domain[oopart[t]]],
                set[0]]]}]] // Reverse

Out[24]= or[equal[domain[UB[x]], union[image[inverse[UB[x]], domain[oopart[t]]], set[0]]],
           not[subclass[x, cart[domain[oopart[t]], domain[oopart[t]]]]]] == True

In[25]:= (% /. {x → x_, t → t_}) /. Equal → SetDelayed
```

Lemma.

```
In[26]:= (Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
             {p1 → subclass[x, cart[domain[oopart[t]], domain[oopart[t]]]], p2 →
               equal[domain[UB[x]], union[image[inverse[UB[x]], domain[oopart[t]]], set[0]]],
              p3 → equal[image[z, domain[UB[x]]], union[
                 image[z, image[inverse[UB[x]], domain[oopart[t]]]], image[z, set[0]]]]}]] //
           Reverse) /. z → composite[IMAGE[oopart[t]], id[P[domain[oopart[t]]]]]

Out[26]= or[equal[image[IMAGE[oopart[t]], intersection[domain[UB[x]], P[domain[oopart[t]]]]],
            union[image[IMAGE[oopart[t]], intersection[
               image[inverse[UB[x]], domain[oopart[t]]], P[domain[oopart[t]]]], set[0]]],
           not[subclass[x, cart[domain[oopart[t]], domain[oopart[t]]]]]] == True

In[27]:= (% /. {t → t_, x → x_}) /. Equal → SetDelayed
```

Lemma.

```
In[16]:= SubstTest[implies, equal[v, w],
           equal[image[u, v], image[u, w]], {u → IMAGE[oopart[t]], v → domain[UB[x]],
            w -> intersection[domain[UB[x]], P[domain[oopart[t]]]]}] // Reverse

Out[16]= or[equal[image[IMAGE[oopart[t]], domain[UB[x]]],
           image[IMAGE[oopart[t]], intersection[domain[UB[x]], P[domain[oopart[t]]]]]],
          not[subclass[domain[x], domain[oopart[t]]]]] == True

In[17]:= (% /. {x → x_, t → t_}) /. Equal → SetDelayed
```

Theorem. If **x** is related to **y** by a similarity transformation **oopart[t]**, then **domain[UB[y]]** is the image of **domain[UB[x]]** under **IMAGE[oopart[x]]**.

```
In[29]:= Map[not, SubstTest[and, implies[and[p1, p2], p5], implies[p2, p6],
           implies[and[p2, p6], p7], implies[and[p5, p7], p8], not[implies[and[p1, p2], p8]],
           {p1 -> equal[y, composite[oopart[t], x, inverse[oopart[t]]]],
            p2 → subclass[x, cartsq[domain[oopart[t]]]], p5 → equal[domain[UB[y]],
              image[IMAGE[oopart[t]], intersection[domain[UB[x]], P[domain[oopart[t]]]]]],
            p6 -> subclass[domain[x], domain[oopart[t]]],
            p7 -> equal[image[IMAGE[oopart[t]], domain[UB[x]]],
              image[IMAGE[oopart[t]], intersection[domain[UB[x]], P[domain[oopart[t]]]]]],
            p8 → equal[domain[UB[y]], image[IMAGE[oopart[t]], domain[UB[x]]]]}]] // Reverse

Out[29]= or[equal[domain[UB[y]], image[IMAGE[oopart[t]], domain[UB[x]]]],
          not[equal[y, composite[oopart[t], x, inverse[oopart[t]]]]],
          not[subclass[x, cart[domain[oopart[t]], domain[oopart[t]]]]] == True

In[30]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

Corollary. (Removal of the **oopart** wrapper.)

```
In[31]:= SubstTest[implies, equal[t, oopart[z]],
           or[equal[domain[UB[y]], image[IMAGE[t], domain[UB[x]]]],
            not[equal[y, composite[t, x, inverse[t]]]],
            not[subclass[x, cart[domain[t], domain[t]]]]], z → t] // Reverse

Out[31]= or[equal[domain[UB[y]], image[IMAGE[t], domain[UB[x]]]],
          not[equal[y, composite[t, x, inverse[t]]]], not[FUNCTION[t]],
          not[FUNCTION[inverse[t]]], not[subclass[x, cart[domain[t], domain[t]]]]] == True

In[32]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

The next step is to combine this result with the equipollence theorem of the preceding section.

Theorem. If **x** and **y** are related by a similarity transformation **t**, then **domain[UB[x]]** and **domain[UB[y]]** are equipollent.

```
In[33]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p4], implies[and[p1, p2, p3], p5],
           implies[and[p4, p5], p6], not[implies[and[p1, p2, p3], p6]],
           {p1 → member[t, BIJ], p2 -> equal[y, composite[t, x, inverse[t]]],
            p3 -> subclass[x, cart[domain[t], domain[t]]],
            p4 -> member[pair[domain[UB[x]], image[IMAGE[t], domain[UB[x]]]], Q],
            p5 -> equal[domain[UB[y]], image[IMAGE[t], domain[UB[x]]]],
            p6 -> member[pair[domain[UB[x]], domain[UB[y]]], Q]}]] // Reverse

Out[33]= or[member[pair[domain[UB[x]], domain[UB[y]]], Q],
           not[equal[y, composite[t, x, inverse[t]]]], not[FUNCTION[t]],
           not[FUNCTION[inverse[t]]], not[member[t, V]],
           not[subclass[x, cart[domain[t], domain[t]]]] == True

In[34]:= (% /. {t → t_, x → x_, y → y_}) /. Equal→ SetDelayed
```

---

eliminating variables

In this section a variable-free restatement is obtained.

Lemma.

```
In[35]:= member[pair[u, v], composite[inverse[UBD], x, UBD]] // AssertTest

Out[35]= member[pair[u, v], composite[inverse[UBD], x, UBD]] ==
          and[member[u, V], member[v, V], member[pair[domain[UB[u]], domain[UB[v]]], x]]

In[36]:= member[pair[u_, v_], composite[inverse[UBD], x_, UBD]] :=
          and[member[u, V], member[v, V], member[pair[domain[UB[u]], domain[UB[v]]], x]]
```

Lemma. Introducing **setpart** wrappers to prepare for removing all variables.

```
In[37]:= SubstTest[implies,
           and[member[t, BIJ], equal[setpart[y], composite[t, setpart[x], inverse[t]]],
            subclass[setpart[x], cart[domain[t], domain[t]]]],
           member[pair[setpart[x], setpart[y]], composite[inverse[UBD], Q, UBD]],
           t → setpart[t]] // Reverse

Out[37]= or[member[pair[domain[UB[setpart[x]]], domain[UB[setpart[y]]]], Q],
           not[equal[composite[setpart[t], setpart[x], inverse[setpart[t]]], setpart[y]]],
           not[FUNCTION[inverse[setpart[t]]]], not[FUNCTION[setpart[t]]],
           not[subclass[setpart[x], cart[domain[setpart[t]], domain[setpart[t]]]]] == True

In[38]:= (% /. {t → t_, x → x_, y → y_}) /. Equal→ SetDelayed
```

Theorem. (Variable elimination.)

```
In[40]:= Map[empty[composite[complement[#], inverse[SECOND]]] &,
          SubstTest[class, pair[pair[t, x], y],
           implies[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[UBD], Q, UBD]}]]

Out[40]= subclass[composite[UBD, SIMILAR, inverse[UBD]], Q] == True

In[41]:= subclass[composite[UBD, SIMILAR, inverse[UBD]], Q] := True
```

Corollary.

```
In[42]:= (or[not[subclass[composite[funpart[t], x, inverse[funpart[t]]], y]],
             subclass[x, composite[inverse[funpart[t]], y, funpart[t]]]] //
            AssertTest) /. {x → SIMILAR, t → UBD, y → Q}

Out[42]= subclass[SIMILAR, composite[inverse[UBD], Q, UBD]] == True

In[43]:= subclass[SIMILAR, composite[inverse[UBD], Q, UBD]] := True
```

Theorem. If **x** and **y** are similar relations, then **domain[UB[x]]** and **domain[UB[y]]** are equipollent.

```
In[44]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
            subclass[u, w], {u → set[PAIR[x, y]], v → SIMILAR,
             w -> composite[inverse[UBD], Q, UBD]}] // Reverse // MapNotNot

Out[44]= or[member[pair[domain[UB[x]], domain[UB[y]]], Q],
           not[member[pair[x, y], SIMILAR]]] == True

In[45]:= or[member[pair[domain[UB[x_]], domain[UB[y_]]], Q],
           not[member[pair[x_, y_], SIMILAR]]] := True
```