similarity for ordinals

Johan G. F. Belinfante
2010 November 5

```
In[1]:= SetDirectory["l:"]; << goedel.10nov05a

:Package Title: goedel.10nov05a                2010 November 5 at 5:30 a.m.

It is now:  2010 Nov 5 at 12:50

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40
```

summary

It is shown that the (two-sided) restriction of the subset relation **S** to one ordinal can not be similar to its restriction to a different ordinal.

abbreviation

The restriction of the subset relation to a class **x** will be abbreviated to **s[x]**.

```
In[2]:= s[x_] := composite[id[x], S, id[x]]
```

domain

An elementary result about domains of bijections is derived here. Recall that **bij[x, y]** denotes the set of all bijections with domain **x** and range **y**.

Lemma.

```
In[3]:= Map[not, SubstTest[and, implies[p1, p3], implies[and[p2, p3], p4],
          not[implies[and[p1, p2], p4]], {p1 → member[t, bij[x, y]], p2 → member[x, z],
           p3 → equal[domain[t], x], p4 -> member[domain[t], z]}]] // Reverse

Out[3]= or[member[domain[t], z], not[member[t, bij[x, y]]], not[member[x, z]]] == True

In[4]:= or[member[domain[t_], z_], not[member[t_, bij[x_, y_]]], not[member[x_, z_]]] := True
```

Theorem. If **t ∈ bij[ord[x], y]**, then **domain[t] ∈ Ω**.

```
In[5]:= SubstTest[implies, and[member[t, bij[u, y]], member[u, z]],
          member[domain[t], z], {u → ord[x], z → OMEGA}] // Reverse

Out[5]= or[member[domain[t], OMEGA], not[member[t, bij[ord[x], y]]]] == True

In[6]:= or[member[domain[t_], OMEGA], not[member[t_, bij[ord[x_], y_]]]] := True
```

range

This section is analogous to the preceding one, but deals with ranges instead of domains.

Lemma.

```
In[7]:= Map[not, SubstTest[and, implies[p1, p3], implies[and[p2, p3], p4],
           not[implies[and[p1, p2], p4]], {p1 → member[t, bij[x, y]], p2 → member[y, z],
            p3 → equal[range[t], y], p4 -> member[range[t], z]}]] // Reverse

Out[7]= or[member[range[t], z], not[member[t, bij[x, y]]], not[member[y, z]]] == True

In[8]:= or[member[range[t_], z_], not[member[t_, bij[x_, y_]]], not[member[y_, z_]]] := True
```

Theorem. If **t ∈ bij[x, ord[y]]**, then **range[t] ∈ Ω**.

```
In[9]:= SubstTest[implies, and[member[t, bij[x, v]], member[v, z]],
          member[range[t], z], {v → ord[y], z → OMEGA}] // Reverse

Out[9]= or[member[range[t], OMEGA], not[member[t, bij[x, ord[y]]]]] == True

In[10]:= or[member[range[t_], OMEGA], not[member[t_, bij[x_, ord[y_]]]]] := True
```

bij[x, y] ⊂ P[cart[x, y]]

Some further elementary results about bijections are derived here.

Theorem. If **t ∈ bij[x, y]**, then **t ⊂ x × y**.

```
In[11]:= SubstTest[implies, and[member[u, v], subclass[v, w]],
           member[u, w], {u → t, v -> bij[x, y], w → P[cart[x, y]]}] // Reverse

Out[11]= or[not[member[t, bij[x, y]]], subclass[t, cart[x, y]]] == True

In[12]:= or[not[member[t_, bij[x_, y_]]], subclass[t_, cart[x_, y_]]] := True
```

Corollary.

```
In[13]:= Map[not, SubstTest[and, implies[p1, p2],
            not[implies[p1, p3]], {p1 → member[t, bij[ord[x], ord[y]]],
             p2 → subclass[t, cart[ord[x], ord[y]]], p3 → subclass[t, cartsq[OMEGA]]}]] // Reverse

Out[13]= or[not[member[t, bij[ord[x], ord[y]]]], subclass[t, cart[OMEGA, OMEGA]]] == True

In[14]:= or[not[member[t_, bij[ord[x_], ord[y_]]]], subclass[t_, cart[OMEGA, OMEGA]]] := True
```

monotone bijections

In this section some consequencess of a rigidity result concerning monotone bijections between ordinals are derived.

Lemma. Monotonicity implies monotonicity of the inverse.

```
In[15]:= Map[not, SubstTest[and, implies[p1, p3], implies[p1, p4],
            implies[p1, p5], implies[p1, p6], not[implies[and[p1, p2], p7]],
            {p1 → member[t, bij[ord[x], ord[y]]], p2 -> subclass[composite[t, S, inverse[t]], S],
             p3 → subclass[t, cartsq[OMEGA]], p4 → member[domain[t], OMEGA],
             p5 → member[range[t], OMEGA], p6 → FUNCTION[inverse[t]],
             p7 -> subclass[composite[inverse[t], S, t], S]}]] // Reverse

Out[15]= or[not[member[t, bij[ord[x], ord[y]]]], not[subclass[composite[t, S, inverse[t]], S]],
          subclass[composite[inverse[t], S, t], S]] == True

In[16]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. The domain and range must be equal.

```
In[17]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
            implies[p1, p4], implies[p1, p5], not[implies[p1, p6]],
            {p1 → and[member[t, bij[ord[x], ord[y]]], subclass[composite[t, S, inverse[t]], S]],
             p2 → FUNCTION[t], p3 -> subclass[composite[inverse[t], S, t], S],
             p4 → member[domain[t], OMEGA], p5 → member[range[t], OMEGA],
             p6 → equal[domain[t], range[t]]}]] // Reverse

Out[17]= or[equal[domain[t], range[t]], not[member[t, bij[ord[x], ord[y]]]],
          not[subclass[composite[t, S, inverse[t]], S]]] == True

In[18]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. There can be no monotone bijection from one ordinal to a different ordinal.

```
In[19]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p1, p3], implies[p1, p4], not[implies[p1, p5]],
            {p1 -> and[member[t, bij[ord[x], ord[y]]], subclass[composite[t, S, inverse[t]], S]],
             p2 -> equal[domain[t], range[t]], p3 -> equal[domain[t], ord[x]],
             p4 → equal[range[t], ord[y]], p5 → equal[ord[x], ord[y]]}]] // Reverse

Out[19]= or[equal[ord[x], ord[y]], not[member[t, bij[ord[x], ord[y]]]],
          not[subclass[composite[t, S, inverse[t]], S]]] == True
```

```
In[20]:= or[equal[ord[x_], ord[y_]], not[member[t_, bij[ord[x_], ord[y_]]]],
           not[subclass[composite[t_, S, inverse[t_]], S]]] := True
```

Eliminating the variable **t** yields the following restatement.

Corollary. If **bij[ord[x], ord[y]] ⋂ monotone[S, S]** is not empty, then **ord[x] = ord[y]**.

```
In[21]:= Map[equal[V, #] &, SubstTest[class, t, implies[member[t, u], equal[v, w]],
           {u → intersection[bij[ord[x], ord[y]], monotone[S, S]], v → ord[x], w → ord[y]}]]

Out[21]= or[equal[0, intersection[bij[ord[x], ord[y]], monotone[S, S]]],
           equal[ord[x], ord[y]]] == True

In[22]:= or[equal[0, intersection[bij[ord[x_], ord[y_]], monotone[S, S]]],
           equal[ord[x_], ord[y_]]] := True
```

Corollary. (Restatement without wrappers.)

```
In[23]:= SubstTest[implies, and[equal[x, ord[u]], equal[y, ord[v]]],
           or[equal[0, intersection[bij[x, y], monotone[S, S]]], equal[x, y]],
           {u → x, v → y}] // Reverse

Out[23]= or[equal[0, intersection[bij[x, y], monotone[S, S]]],
           equal[x, y], not[member[x, OMEGA]], not[member[y, OMEGA]]] == True

In[24]:= or[equal[0, intersection[bij[x_, y_], monotone[S, S]]],
           equal[x_, y_], not[member[x_, OMEGA]], not[member[y_, OMEGA]]] := True
```

rigidity in terms of similarity

In this section, the rigidity results are again restated, this time in terms of similarity of restrictions of the subset relation to ordinals.

Theorem. If **s[ord[x]]** is similar to **s[ord[y]]**, then **ord[x] = ord[y]**.

```
In[25]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p2, p3], implies[p3, p4], not[implies[p1, p4]],
            {p1 -> member[pair[composite[id[ord[x]], S, id[ord[x]]], composite[id[ord[y]], S,
                 id[ord[y]]]], SIMILAR], p2 -> not[empty[intersection[bij[ord[x], ord[y]],
                 monotone[S, S], monotone[complement[S], complement[S]]]]],
             p3 -> not[empty[intersection[bij[ord[x], ord[y]], monotone[S, S]]]],
             p4 → equal[ord[x], ord[y]]}]] // Reverse

Out[25]= or[equal[ord[x], ord[y]], not[member[pair[composite[id[ord[x]], S, id[ord[x]]],
            composite[id[ord[y]], S, id[ord[y]]]], SIMILAR]]] == True

In[26]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Since the converse also holds, the above result can be stated as a logical equivalence, which can be made into a simpler rewrite rule.

Corollary. A better rewrite rule.

```
In[27]:= equiv[member[pair[composite[id[ord[x]], S, id[ord[x]]],
            composite[id[ord[y]], S, id[ord[y]]]], SIMILAR], equal[ord[x], ord[y]]]

Out[27]= True

In[28]:= member[pair[composite[id[ord[x_]], S, id[ord[x_]]],
            composite[id[ord[y_]], S, id[ord[y_]]]], SIMILAR] := equal[ord[x], ord[y]]
```

Corollary. (Eliminating the **ord** wrappers.)

```
In[29]:= SubstTest[implies, and[equal[x, ord[u]], equal[y, ord[v]],
            member[pair[s[x], s[y]], SIMILAR]], equal[x, y], {u → x, v → y}] // Reverse

Out[29]= or[equal[x, y], not[member[x, OMEGA]], not[member[y, OMEGA]], not[member[
             pair[composite[id[x], S, id[x]], composite[id[y], S, id[y]]], SIMILAR]]] == True

In[30]:= or[equal[x_, y_], not[member[x_, OMEGA]], not[member[y_, OMEGA]], not[member[
             pair[composite[id[x_], S, id[x_]], composite[id[y_], S, id[y_]]], SIMILAR]]] := True
```

variable-free restatement

A variable-free reformulation of the rigidity theorem derived in the preceding section can be obtained easily using **AssertTest**.

Theorem.

```
In[31]:= equal[composite[id[OMEGA], inverse[DUP], inverse[CART], inverse[IMAGE[id[S]]],
            SIMILAR, IMAGE[id[S]], CART, DUP, id[OMEGA]], id[OMEGA]] // AssertTest

Out[31]= equal[composite[id[OMEGA], inverse[DUP], inverse[CART], inverse[IMAGE[id[S]]],
            SIMILAR, IMAGE[id[S]], CART, DUP, id[OMEGA]], id[OMEGA]] == True

In[32]:= composite[id[OMEGA], inverse[DUP], inverse[CART], inverse[IMAGE[id[S]]],
           SIMILAR, IMAGE[id[S]], CART, DUP, id[OMEGA]] := id[OMEGA]
```

Corollary. The restriction of the similarity relation to the class of restrictions of the subset relation to ordinals is the identity relation on that class.

```
In[33]:= Map[composite[IMAGE[id[S]], CART, DUP, inverse[#]] &,
           Assoc[composite[IMAGE[id[S]], CART, DUP],
            composite[id[OMEGA], inverse[DUP], inverse[CART], inverse[IMAGE[id[S]]]],
            composite[SIMILAR, IMAGE[id[S]], CART, DUP, id[OMEGA]]]] // Reverse

Out[33]= composite[id[image[IMAGE[id[S]], image[CART, id[OMEGA]]]],
           SIMILAR, id[image[IMAGE[id[S]], image[CART, id[OMEGA]]]]] ==
          id[image[IMAGE[id[S]], image[CART, id[OMEGA]]]]
```

```
In[34]:= composite[id[image[IMAGE[id[S]], image[CART, id[OMEGA]]]],
           SIMILAR, id[image[IMAGE[id[S]], image[CART, id[OMEGA]]]]] :=
          id[image[IMAGE[id[S]], image[CART, id[OMEGA]]]]
```