rigidity theorem for ordinals

Johan G. F. Belinfante
2010 October 28

```
In[1]:= SetDirectory["l:"]; << goedel.10oct27a

        :Package Title: goedel.10oct27a            2010 October 27 at 2:45 p.m.

        It is now:  2010 Oct 28 at 10:2

        Loading Simplification Rules

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

        weightlimit = 40
```

summary

In this notebook a rigidity theorem about the class Ω of ordinals is derived. It is shown below that any strictly monotone function $\mathbf{f} \subset \Omega \times \Omega$ with a full domain and a full range must be an identity function. As a corollary of this it follows that there can be no strictly monotone function from one ordinal onto a different ordinal.

derivation

If a strictly monotone function $\mathbf{f} \subset \Omega \times \Omega$ has a full domain, then $\mathbf{f} \subset \mathbf{S}$. One version of this is already available.

```
In[2]:= or[not[FUNCTION[f]], not[subclass[f, cart[OMEGA, OMEGA]]],
         not[subclass[f, composite[S, IMAGE[f]]]],
         not[subclass[U[domain[f]], domain[f]]], subclass[f, S]]

Out[2]= True
```

For functions on ordinals there are several equivalent formulations of the strict monotonicity condition. Here is another version of the above result.

Theorem. If a strictly monotone function $\mathbf{x} \subset \Omega \times \Omega$ has a full domain, then $\mathbf{x} \subset \mathbf{S}$.

```
In[3]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p5],
           implies[and[p1, p2, p4, p5], p6], not[implies[and[p1, p2, p3, p4], p6]],
           {p1 → FUNCTION[x], p2 → subclass[x, cart[OMEGA, OMEGA]],
            p3 → subclass[composite[inverse[x], S, x], S], p4 → full[domain[x]],
            p5 → subclass[x, composite[S, IMAGE[x]]], p6 → subclass[x, S]}]] // Reverse

Out[3]= or[not[FUNCTION[x]], not[subclass[x, cart[OMEGA, OMEGA]]],
          not[subclass[composite[inverse[x], S, x], S]],
          not[subclass[U[domain[x]], domain[x]]], subclass[x, S]] == True

In[4]:= or[not[FUNCTION[x_]], not[subclass[x_, cart[OMEGA, OMEGA]]],
          not[subclass[composite[inverse[x_], S, x_], S]],
          not[subclass[U[domain[x_]], domain[x_]]], subclass[x_, S]] := True
```

Replacing **x** with its inverse yields the following.

Lemma.

```
In[5]:= SubstTest[implies, and[FUNCTION[t], subclass[t, cart[OMEGA, OMEGA]],
           subclass[composite[inverse[t], S, t], S], full[domain[t]]],
          subclass[t, S], t → inverse[x]] // Reverse

Out[5]= or[and[not[equal[OMEGA, range[x]]], not[member[range[x], OMEGA]]],
          not[FUNCTION[inverse[x]]], not[subclass[composite[x, S, inverse[x]], S]],
          not[subclass[domain[x], OMEGA]], subclass[inverse[x], S]] == True

In[6]:= (% /. x → x_) /. Equal → SetDelayed
```

This result can be cleaned up slightly.

Theorem.

```
In[7]:= Map[not, SubstTest[and, implies[and[p4, p7], not[p5]],
           or[not[p1], not[p3], p5, not[p6], p8], p1, p2, p3, p4, not[p8],
           {p1 → FUNCTION[inverse[x]], p2 → subclass[x, cart[OMEGA, OMEGA]],
            p3 -> subclass[composite[x, S, inverse[x]], S], p4 → full[range[x]],
            p5 -> and[not[equal[OMEGA, range[x]]], not[member[range[x], OMEGA]]],
            p6 → subclass[domain[x], OMEGA], p7 → subclass[range[x], OMEGA],
            p8 → subclass[inverse[x], S]}]] // Reverse

Out[7]= or[not[FUNCTION[inverse[x]]], not[subclass[x, cart[OMEGA, OMEGA]]],
          not[subclass[composite[x, S, inverse[x]], S]],
          not[subclass[U[range[x]], range[x]]], subclass[inverse[x], S]] == True

In[8]:= or[not[FUNCTION[inverse[x_]]], not[subclass[composite[x_, S, inverse[x_]], S]],
          not[subclass[x_, cart[OMEGA, OMEGA]]],
          not[subclass[U[range[x_]], range[x_]]], subclass[inverse[x_], S]] := True
```

If the domain and range are both full, one can combine the above results to obtain a rigidity theorem. (The following step in the proof of this theorem has been deliberately omitted to speed up execution: **implies[and[p1, p3, p4, p5], p7]**.)

Rigidity Theorem. If a strictly monotone function $\mathbf{x} \subset \Omega \times \Omega$ has a full domain and a full range, then $\mathbf{x} \subset \mathbf{Id}$.

```
In[9]:= Map[not, SubstTest[and, implies[and[p1, p2], p6], implies[p1, p4], implies[p1, p5],
          implies[and[p6, p7], p8], not[implies[and[p1, p2, p3], p8]], {p1 → and[FUNCTION[x],
            subclass[x, cart[OMEGA, OMEGA]], subclass[composite[inverse[x], S, x], S]],
           p2 → full[domain[x]], p3 → full[range[x]], p4 → FUNCTION[inverse[x]],
           p5 -> subclass[composite[x, S, inverse[x]], S], p6 → subclass[x, S],
           p7 → subclass[inverse[x], S], p8 → subclass[x, Id]}]] // Reverse

Out[9]= or[not[FUNCTION[x]], not[subclass[x, cart[OMEGA, OMEGA]]],
          not[subclass[composite[inverse[x], S, x], S]], not[subclass[U[domain[x]], domain[x]]],
          not[subclass[U[range[x]], range[x]]], subclass[x, Id]] == True

In[10]:= or[not[FUNCTION[x_]], not[subclass[x_, cart[OMEGA, OMEGA]]],
           not[subclass[composite[inverse[x_], S, x_], S]],
           not[subclass[U[domain[x_]], domain[x_]]],
           not[subclass[U[range[x_]], range[x_]]], subclass[x_, Id]] := True
```

corollaries

A full subclass of Ω is either an ordinal or Ω itself. If the domain of a bijection is a set, so is its range, and vice versa. So there are just two cases to be considered: either the domain and range are both equal to Ω or both are ordinals.

Lemma. This lemma is needed to overcome a limitation on equality substitution in the **GOEDEL** program.

```
In[11]:= SubstTest[or, not[equal[y, domain[x]]], not[equal[z, range[x]]], not[FUNCTION[x]],
           subclass[x, cart[y, z]], {y → domain[x], z → domain[x]}] // Reverse

Out[11]= or[not[equal[domain[x], range[x]]],
           not[FUNCTION[x]], subclass[x, cart[domain[x], domain[x]]]] == True

In[12]:= or[not[equal[domain[x_], range[x_]]], not[FUNCTION[x_]],
           subclass[x_, cart[domain[x_], domain[x_]]]] := True
```

The derivation of the following theorem is speeded up by omitting this proof step: **implies[and[p1, p2, p5, p6, p7], p8]**.

Theorem. The case that both domain and range are equal to Ω.

```
In[13]:= Map[not, SubstTest[and, implies[and[p1, p3, p4], p5], implies[p3, p6],
            implies[p4, p7], implies[and[p3, p8], p9], not[implies[and[p1, p2, p3, p4], p9]],
            {p1 → FUNCTION[x], p2 → subclass[composite[inverse[x], S, x], S],
             p3 → equal[domain[x], OMEGA], p4 → equal[range[x], OMEGA],
             p5 → subclass[x, cart[OMEGA, OMEGA]], p6 → full[domain[x]],
             p7 → full[range[x]], p8 → subclass[x, Id], p9 → equal[x, id[OMEGA]]}]] // Reverse

Out[13]= or[equal[x, id[OMEGA]], not[equal[OMEGA, domain[x]]], not[equal[OMEGA, range[x]]],
           not[FUNCTION[x]], not[subclass[composite[inverse[x], S, x], S]]] == True

In[14]:= or[equal[x_, id[OMEGA]], not[equal[OMEGA, domain[x_]]], not[equal[OMEGA, range[x_]]],
           not[FUNCTION[x_]], not[subclass[composite[inverse[x_], S, x_], S]]] := True
```

The case that the domain and range are both ordinals will now be considered.

Lemma. If the domain and range of a function $\mathbf{x}$ are both ordinals, then $\mathbf{x} \subset \Omega \times \Omega$.

```
In[15]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
           implies[p1, p4], implies[and[p2, p3, p4], p5], not[implies[p1, p5]],
           {p1 -> and[FUNCTION[x], member[domain[x], OMEGA], member[range[x], OMEGA]],
            p2 → subclass[x, cart[V, V]], p3 → subclass[domain[x], OMEGA],
            p4 → subclass[range[x], OMEGA], p5 → subclass[x, cart[OMEGA, OMEGA]]}]] // Reverse

Out[15]= or[not[FUNCTION[x]], not[member[domain[x], OMEGA]],
          not[member[range[x], OMEGA]], subclass[x, cart[OMEGA, OMEGA]]] == True

In[16]:= or[not[FUNCTION[x_]], not[member[domain[x_], OMEGA]],
          not[member[range[x_], OMEGA]], subclass[x_, cart[OMEGA, OMEGA]]] := True
```

Theorem. A strictly monotone function whose domain and range are both ordinals is an identity function.

```
In[17]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
           implies[p1, p4], implies[and[p1, p2, p3, p4], p5], not[implies[p1, p5]],
           {p1 → and[FUNCTION[x], subclass[composite[inverse[x], S, x], S],
              member[domain[x], OMEGA], member[range[x], OMEGA]],
            p2 → subclass[x, cart[OMEGA, OMEGA]], p3 → full[domain[x]],
            p4 → full[range[x]], p5 → subclass[x, Id]}]] // Reverse

Out[17]= or[not[FUNCTION[x]], not[member[domain[x], OMEGA]], not[member[range[x], OMEGA]],
          not[subclass[composite[inverse[x], S, x], S]], subclass[x, Id]] == True

In[18]:= or[not[FUNCTION[x_]], not[member[domain[x_], OMEGA]], not[member[range[x_], OMEGA]],
          not[subclass[composite[inverse[x_], S, x_], S]], subclass[x_, Id]] := True
```

The fact that the function is an identity function implies that the domain and range must be equal.

Corollary. There can be no strictly monotone function from one ordinal onto a different ordinal.

```
In[19]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 → and[FUNCTION[x], subclass[composite[inverse[x], S, x], S],
              member[domain[x], OMEGA], member[range[x], OMEGA]],
            p2 → subclass[x, Id], p3 → equal[domain[x], range[x]]}]] // Reverse

Out[19]= or[equal[domain[x], range[x]], not[FUNCTION[x]], not[member[domain[x], OMEGA]],
          not[member[range[x], OMEGA]], not[subclass[composite[inverse[x], S, x], S]]] == True

In[20]:= or[equal[domain[x_], range[x_]], not[FUNCTION[x_]],
          not[member[domain[x_], OMEGA]], not[member[range[x_], OMEGA]],
          not[subclass[composite[inverse[x_], S, x_], S]]] := True
```

A slightly more symmetric statement of the preceding theorem is obtained by replacing the condition that $\mathbf{x}$ be a function with a monotonicity condition.

Corollary.

```
In[21]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p2, p3], p4],
           not[implies[and[p1, p3], p4]], {p1 → subclass[P[t], monotone[S, S]],
            p2 → FUNCTION[t], p3 → and[subclass[composite[inverse[t], S, t], S],
              member[domain[t], OMEGA], member[range[t], OMEGA]],
            p4 -> equal[domain[t], range[t]]}] /. t → composite[Id, x]] // Reverse

Out[21]= or[equal[domain[x], range[x]], not[member[domain[x], OMEGA]],
          not[member[range[x], OMEGA]], not[subclass[composite[x, S, inverse[x]], S]],
          not[subclass[composite[inverse[x], S, x], S]]] == True

In[22]:= or[equal[domain[x_], range[x_]], not[member[domain[x_], OMEGA]],
          not[member[range[x_], OMEGA]], not[subclass[composite[x_, S, inverse[x_]], S]],
          not[subclass[composite[inverse[x_], S, x_], S]]] := True
```