wellordering theorem

Johan G. F. Belinfante
2012 April 12

In[1]:= **SetDirectory["l:"]; << goedel.12apr12a**

```
:Package Title: goedel.12apr12a                    2012 April 12 at 1:15 p.m.

Loading takes about seventeen minutes, half that time due to builtin pauses.

It is now:  2012 Apr 12 at 14:52

Loading Simplification Rules

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

weightlimit = 40

Loading completed.

It is now:  2012 Apr 12 at 15:9
```

summary

A set **x** is ***well-orderable*** if there is a well-ordering **w** with **x = fix[w]**. In this notebook it is shown that if a set can be well-ordered, then it is equipollent to an ordinal. The converse is also true. It follows that a set can be well-ordered if and only if it is equipollent to an ordinal. This is true independently of the axiom of choice. The proof uses well-founded recursion, and the fact that a set is equipollent to an ordinal if and only if it is equipollent to a set of ordinals.

In[2]:= **image[Q, P[OMEGA]]**

Out[2]= image[Q, OMEGA]

The standard formulation of the well-ordering theorem follows as a corollary, that **axch** is equivalent to the statement than any set can be well-ordered. The following slightly weaker form of this is already available in the **GOEDEL** program.

In[3]:= **equal[V, image[Q, OMEGA]]**

Out[3]= axch

This says that **axch** is equivalent to the statement that any set is equipollent to an ordinal.

strict monotonicity for well-orderings

Since any well-order is a total order, strictly monotone functions are bijections.

Theorem.

```
In[4]:= SubstTest[intersection, map[fix[to[t]], y],
          monotone[intersection[Di, to[t]], Di], t → wo[x]] // Reverse

Out[4]= intersection[map[fix[wo[x]], y], monotone[intersection[Di, wo[x]], Di]] ==
         intersection[BIJ, map[fix[wo[x]], y]]

In[5]:= intersection[map[fix[wo[x_]], y_], monotone[intersection[Di, wo[x_]], Di]] :=
         intersection[BIJ, map[fix[wo[x]], y]]
```

From the general theory of well-founded recursion, it follows that the rank function for a small wellfounded relation is strictly monotone.

```
In[6]:= subclass[composite[
          rec[composite[TC, IMAGE[SECOND], SECOND], inverse[wf[setpart[x]]]], wf[setpart[x]],
          inverse[rec[composite[TC, IMAGE[SECOND], SECOND], inverse[wf[setpart[x]]]]]], E]

Out[6]= True
```

Technical Lemma, One can add a harmless factor **id[y]**.

```
In[7]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
          subclass[u, w], {u -> composite[rec[composite[TC, IMAGE[SECOND], SECOND],
              inverse[wf[setpart[x]]]], wf[setpart[x]], id[y],
             inverse[rec[composite[TC, IMAGE[SECOND], SECOND], inverse[wf[setpart[x]]]]]],
           v -> composite[rec[composite[TC, IMAGE[SECOND], SECOND], inverse[wf[setpart[x]]]],
             wf[setpart[x]], inverse[rec[composite[TC, IMAGE[SECOND], SECOND],
               inverse[wf[setpart[x]]]]]], w -> E}] // Reverse

Out[7]= subclass[composite[rec[composite[TC, IMAGE[SECOND], SECOND], inverse[wf[setpart[x]]]],
           wf[setpart[x]], id[y], inverse[
            rec[composite[TC, IMAGE[SECOND], SECOND], inverse[wf[setpart[x]]]]]], E] == True

In[8]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Explicit reference to the recursion construction can be eliminated. All one needs to know is the existence of a function with the above property.

Theorem. There is a strictly monotone function from the range of any small well-founded relation to a set of ordinals.

```
In[9]:= Map[not, SubstTest[implies, member[u, v], not[empty[v]],
           {u -> composite[rec[composite[TC, IMAGE[SECOND], SECOND], inverse[wf[setpart[x]]]],
              id[range[wf[setpart[x]]]]], v -> intersection[
              map[range[wf[setpart[x]]], OMEGA], monotone[wf[setpart[x]], E]]}]] // Reverse

Out[9]= equal[0, intersection[map[range[wf[setpart[x]]], OMEGA], monotone[wf[setpart[x]], E]]] ==
         False

In[10]:= equal[0, intersection[
            map[range[wf[setpart[x_]]], OMEGA], monotone[wf[setpart[x_]], E]]] := False
```

Lemma. An inclusion.

```
In[11]:= SubstTest[implies, subclass[u, v], subclass[monotone[x, u], monotone[x, v]],
          {u -> composite[id[OMEGA], E], v → Di}] // Reverse

Out[11]= subclass[monotone[x, composite[id[OMEGA], E]], monotone[x, Di]] == True

In[12]:= subclass[monotone[x_, composite[id[OMEGA], E]], monotone[x_, Di]] := True
```

Lemma. Another version of the statement that there is a strictly monotone mapping from the range of a small well-founded relation to the class of ordinals.

```
In[13]:= Map[not, SubstTest[implies, member[u, v], not[empty[v]],
           {u -> composite[rec[composite[TC, IMAGE[SECOND], SECOND], inverse[wf[setpart[x]]]],
              id[range[wf[setpart[x]]]]], v -> intersection[map[range[wf[setpart[x]]], OMEGA],
              monotone[wf[setpart[x]], composite[id[OMEGA], E]]]}]] // Reverse

Out[13]= equal[0, intersection[map[range[wf[setpart[x]]], OMEGA],
           monotone[wf[setpart[x]], composite[id[OMEGA], E]]]] == False

In[14]:= equal[0, intersection[map[range[wf[setpart[x_]]], OMEGA],
           monotone[wf[setpart[x_]], composite[id[OMEGA], E]]]] := False
```

Theorem. Yet another statement about the existence of a strictly monotone function, using the diversity relation **Di**.

```
In[15]:= SubstTest[and, empty[v], subclass[u, v],
          {u -> intersection[map[range[wf[setpart[x]]], OMEGA],
             monotone[wf[setpart[x]], composite[id[OMEGA], E]]], v -> intersection[
             map[range[wf[setpart[x]]], OMEGA], monotone[wf[setpart[x]], Di]]}] // Reverse

Out[15]= equal[0, intersection[
           map[range[wf[setpart[x]]], OMEGA], monotone[wf[setpart[x]], Di]]] == False

In[16]:= equal[0, intersection[map[range[wf[setpart[x_]]], OMEGA],
           monotone[wf[setpart[x_]], Di]]] := False
```

The strict part of any well-order is well-founded. The following result follows from the above theorem.

Corollary. Existence of strictly monotone mappings to Ω for well-orders.

```
In[17]:= SubstTest[equal, 0, intersection[map[range[wf[setpart[t]]], OMEGA],
           monotone[wf[setpart[t]], Di]], t → intersection[wo[setpart[x]], Di]] // Reverse

Out[17]= equal[0, intersection[map[fix[composite[wo[setpart[x]], Di]], OMEGA],
           monotone[intersection[Di, wo[setpart[x]]], Di]]] == False

In[18]:= equal[0, intersection[map[fix[composite[wo[setpart[x_]], Di]], OMEGA],
           monotone[intersection[Di, wo[setpart[x_]]], Di]]] := False
```

Lemma. A simplification rule.

```
In[19]:= SubstTest[intersection, monotone[intersection[Di, to[t]], Di],
           P[cart[fix[to[t]], V]], t → wo[x]] // Reverse

Out[19]= intersection[monotone[intersection[Di, wo[x]], Di], P[cart[fix[wo[x]], V]]] ==
          intersection[image[INVERSE, FUNS], P[cart[fix[wo[x]], V]]]

In[20]:= intersection[monotone[intersection[Di, wo[x_]], Di], P[cart[fix[wo[x_]], V]]] :=
          intersection[image[INVERSE, FUNS], P[cart[fix[wo[x]], V]]]
```

Theorem. There is one-to-one mapping from the range of the strict part of a well-order to a set of ordinals.

```
In[21]:= Map[empty, AssInt[
            monotone[intersection[Di, wo[setpart[x]]], Di], P[cart[fix[wo[setpart[x]]], V]],
            map[fix[composite[wo[setpart[x]], Di]], OMEGA]]] // Reverse

Out[21]= equal[0, intersection[BIJ, map[fix[composite[wo[setpart[x]], Di]], OMEGA]]] == False

In[22]:= equal[0, intersection[BIJ, map[fix[composite[wo[setpart[x_]], Di]], OMEGA]]] := False
```

Corollary. The range of the strict part of a small well-order is equipollent to an ordinal.

```
In[23]:= SubstTest[implies, not[disjoint[BIJ, map[u, v]]], member[u, image[Q, P[v]]],
           {u -> fix[composite[wo[setpart[x]], Di]], v -> OMEGA}] // Reverse

Out[23]= member[fix[composite[wo[setpart[x]], Di]], image[Q, OMEGA]] == True

In[24]:= member[fix[composite[wo[setpart[x_]], Di]], image[Q, OMEGA]] := True
```

well-orderable sets

The range of the strict part of a total order is contained in the fixed point class of the total order. These two classes differ only a little.

Theorem. A simplification rule for total orders.

```
In[26]:= equal[union[fix[funpart[to[x]]], fix[to[x]]], fix[to[x]]]

Out[26]= True

In[27]:= union[fix[funpart[to[x_]]], fix[to[x_]]] := fix[to[x]]
```

Corollary. A simplification rule for well orders.

```
In[28]:= SubstTest[union, fix[funpart[to[t]]], fix[to[t]], t → wo[x]] // Reverse

Out[28]= union[fix[funpart[wo[x]]], fix[wo[x]]] == fix[wo[x]]

In[29]:= union[fix[funpart[wo[x_]]], fix[wo[x_]]] := fix[wo[x]]
```

Lemma. The function part of a total order is finite.

```
In[30]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
           {u → fix[funpart[to[x]]], v → union[set[0], range[SINGLETON]], w → FINITE}] // Reverse

Out[30]= member[fix[funpart[to[x]]], FINITE] == True

In[31]:= member[fix[funpart[to[x_]]], FINITE] := True
```

Theorem. A similar rule for the inverse of a total order.

```
In[32]:= SubstTest[member, fix[funpart[to[t]]], FINITE, t → inverse[to[x]]] // Reverse

Out[32]= member[fix[funpart[inverse[to[x]]]], FINITE] == True

In[33]:= member[fix[funpart[inverse[to[x_]]]], FINITE] := True
```

Theorem. A similar result for a well-ordering.

```
In[34]:= SubstTest[member, fix[funpart[to[t]]], FINITE, t → wo[x]] // Reverse

Out[34]= member[fix[funpart[wo[x]]], FINITE] == True

In[35]:= member[fix[funpart[wo[x_]]], FINITE] := True
```

Theorem. .Similar result for the inverse of a well-ordering.

```
In[36]:= SubstTest[member, fix[funpart[inverse[to[t]]]], FINITE, t → wo[x]] // Reverse

Out[36]= member[fix[funpart[inverse[wo[x]]]], FINITE] == True

In[37]:= member[fix[funpart[inverse[wo[x_]]]], FINITE] := True
```

Theorem. The fixed point set of a (small) well-order is equipollent to an ordinal.

```
In[38]:= SubstTest[implies, and[member[u, image[Q, OMEGA]], member[v, FINITE]],
           member[union[u, v], image[Q, OMEGA]],
           {u → fix[composite[wo[setpart[x]], Di]], v → union[fix[funpart[wo[setpart[x]]]],
              fix[funpart[inverse[wo[setpart[x]]]]]]}] // Reverse

Out[38]= member[fix[wo[setpart[x]]], image[Q, OMEGA]] == True

In[39]:= member[fix[wo[setpart[x_]]], image[Q, OMEGA]] := True
```

The following corollary is obtained by eliminating the compound wrapper **wo[setpart[x]]**.

Corollary.

```
In[40]:= Map[implies[member[x, y], #] &, SubstTest[implies,
           equal[x, wo[setpart[t]]], member[fix[x], image[Q, OMEGA]], t → x]] // Reverse

Out[40]= or[member[fix[x], image[Q, OMEGA]], not[member[x, y]], not[WELLORDER[x]]] == True

In[41]:= or[member[fix[x_], image[Q, OMEGA]], not[member[x_, y_]], not[WELLORDER[x_]]] := True
```

The next step will be to eliminate the variable **x**.

Lemma. A simplification rule.

```
In[42]:= Assoc[IMAGE[inverse[DUP]],
           IMAGE[id[cart[V, V]]], inverse[IMAGE[id[cart[V, V]]]]] // Reverse

Out[42]= composite[IMAGE[inverse[DUP]], inverse[IMAGE[id[cart[V, V]]]]] ==
          composite[IMAGE[inverse[DUP]], id[P[cart[V, V]]]]

In[43]:= composite[IMAGE[inverse[DUP]], inverse[IMAGE[id[cart[V, V]]]]] :=
          composite[IMAGE[inverse[DUP]], id[P[cart[V, V]]]]
```

Lemma. A simplification rule.

```
In[44]:= ImageComp[IMAGE[inverse[DUP]], inverse[IMAGE[id[cart[V, V]]]], x] // Reverse

Out[44]= image[IMAGE[inverse[DUP]], image[inverse[IMAGE[id[cart[V, V]]]], x]] ==
          image[IMAGE[inverse[DUP]], intersection[x, P[cart[V, V]]]]

In[45]:= image[IMAGE[inverse[DUP]], image[inverse[IMAGE[id[cart[V, V]]]], x_]] :=
          image[IMAGE[inverse[DUP]], intersection[x, P[cart[V, V]]]]
```

A variable-free statement is derived using **reify** and **case**.

Lemma. The class of fixed point sets of well-orderings is a subclass of the class of sets equipollent to an ordinal.

```
In[46]:= Map[equal[domain[#], V] &,
          SubstTest[reify, x, case[member[fix[wo[setpart[x]]], y]], y -> image[Q, OMEGA]]]

Out[46]= subclass[image[IMAGE[inverse[DUP]], WO], image[Q, OMEGA]] == True

In[47]:= % /. Equal → SetDelayed
```

Main Theorem. The class of fixed point sets of well-orderings is equal to the class of sets equipollent to an ordinal.

```
In[48]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[IMAGE[inverse[DUP]], WO], v -> image[Q, OMEGA]}]

Out[48]= equal[image[Q, OMEGA], image[IMAGE[inverse[DUP]], WO]] == True

In[49]:= image[IMAGE[inverse[DUP]], WO] := image[Q, OMEGA]
```

It is to be emphasized that everything up to here has been derived without using the axiom of choice.

the well-ordering theorem

The usual statement of the well-ordering theorem follows as a corollary of the main theorem derived in the preceding section. The axiom of choice is equivalent to the statement that every set can be well-ordered. No new rewrite rule is needed here.

```
In[50]:= equal[V, image[IMAGE[inverse[DUP]], WO]]

Out[50]= axch
```