a characterization of well-ordering

Johan G. F. Belinfante
2005 April 7

```
In[1]:= SetDirectory["i:"]; << goedel67.29c; << tools.m

        :Package Title: goedel67.29c          2005 March 29 at 11:30 a.m.

        It is now:  2005 Apr 7 at 10:35

        Loading Simplification Rules

        TOOLS.M                       Revised 2005 February 22

        weightlimit = 40
```

summary

It is shown that **x** is a well-ordering if and only if **x** is a total order and **intersection[-Di,x]** is well-founded.

normalization rules

The derivations in this notebook are speeded up by clearing two flags that control certain conditional rewrite rules.

```
In[2]:= simplify = False; cond = False;
```

The first step will be to add some normalization rules for the predicates **PARTIALORDER** and **TOTALORDER**.

```
In[3]:= TOTALORDER[x] // AssertTest // Reverse

Out[3]= and[PARTIALORDER[x], subclass[cart[fix[x], fix[x]], union[x, inverse[x]]]] ==
         TOTALORDER[x]

In[4]:= and[PARTIALORDER[x_],
          subclass[cart[fix[x_], fix[x_]], union[x_, inverse[x_]]]] := TOTALORDER[x]
```

The remaining rules are negative forms of existing rules.

```
In[14]:= or[not[subclass[domain[x], fix[x]]],
           not[subclass[range[x], fix[x]]]] // NotNotTest

Out[14]= or[not[subclass[domain[x], fix[x]]], not[subclass[range[x], fix[x]]]] ==
          not[REFLEXIVE[composite[Id, x]]]

In[15]:= or[not[subclass[domain[x_], fix[x_]]], not[subclass[range[x_], fix[x_]]]] :=
          not[REFLEXIVE[composite[Id, x]]]

In[17]:= or[not[REFLEXIVE[composite[Id, x]]], not[TRANSITIVE[x]]] // NotNotTest

Out[17]= or[not[REFLEXIVE[composite[Id, x]]], not[TRANSITIVE[x]]] ==
          or[not[REFLEXIVE[x]], not[TRANSITIVE[x]]]

In[18]:= or[not[REFLEXIVE[composite[Id, x_]]], not[TRANSITIVE[x_]]] :=
          or[not[REFLEXIVE[x]], not[TRANSITIVE[x]]]

In[21]:= or[not[PARTIALORDER[x]],
           not[subclass[cart[fix[x], fix[x]], union[x, inverse[x]]]]] // NotNotTest

Out[21]= or[not[PARTIALORDER[x]], not[
            subclass[cart[fix[x], fix[x]], union[x, inverse[x]]]]] == not[TOTALORDER[x]]

In[22]:= or[not[PARTIALORDER[x_]],
           not[subclass[cart[fix[x_], fix[x_]], union[x_, inverse[x_]]]]] := not[
           TOTALORDER[x]]
```

---

a theorem

Lemma.

```
In[23]:= equal[composite[id[fix[x]], intersection[Di, complement[inverse[x]]]],
           composite[id[fix[x]], complement[inverse[x]]]] // AssertTest

Out[23]= equal[composite[id[fix[x]], complement[inverse[x]]],
           composite[id[fix[x]], intersection[Di, complement[inverse[x]]]]] == True

In[24]:= composite[id[fix[x_]], intersection[Di, complement[inverse[x_]]]] :=
          composite[id[fix[x]], complement[inverse[x]]]
```

Theorem.

```
In[25]:= SubstTest[implies, and[equal[intersection[x, y], w], equal[union[x, y], z]],
          equal[intersection[v, x], intersection[v, z, union[w, complement[y]]]],
          {v → Di, y → inverse[x], w → id[fix[x]], z → cart[fix[x], fix[x]]}]

Out[25]= or[equal[composite[id[fix[x]], complement[inverse[x]], id[fix[x]]],
            intersection[Di, x]],
           not[equal[cart[fix[x], fix[x]], union[x, inverse[x]]]],
           not[subclass[intersection[x, inverse[x]], Id]]] == True

In[26]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[27]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
           {p1 → TOTALORDER[x], p2 → equal[intersection[x, inverse[x]], id[fix[x]]],
            p3 → equal[union[x, inverse[x]], cart[fix[x], fix[x]]],
            p4 → equal[intersection[Di, x], composite[id[fix[x]],
               intersection[Di, complement[inverse[x]]], id[fix[x]]]]}]]

Out[27]= or[equal[composite[id[fix[x]], complement[inverse[x]], id[fix[x]]],
            intersection[Di, x]], not[TOTALORDER[x]]] == True

In[28]:= or[equal[composite[id[fix[x_]], complement[inverse[x_]], id[fix[x_]]],
            intersection[Di, x_]], not[TOTALORDER[x_]]] := True
```

---

not quite a proof

Lemma.

```
In[29]:= subvar[composite[complement[inverse[x]], id[fix[x]]]] // Normality

Out[29]= subvar[composite[complement[inverse[x]], id[fix[x]]]] ==
          complement[domain[LEAST[union[x, cart[V, complement[fix[x]]]]]]]

In[30]:= subvar[composite[complement[inverse[x_]], id[fix[x_]]]] :=
          complement[domain[LEAST[union[x, cart[V, complement[fix[x]]]]]]]
```

Lemma.

```
In[31]:= equal[intersection[complement[y], P[fix[x]]], set[0]] // AssertTest

Out[31]= equal[intersection[complement[y], P[fix[x]]], set[0]] ==
          and[not[member[0, y]], subclass[P[fix[x]], union[y, set[0]]]]

In[32]:= equal[intersection[complement[y_], P[fix[x_]]], set[0]] :=
          and[not[member[0, y]], subclass[P[fix[x]], union[y, set[0]]]]
```

```
In[33]:= SubstTest[implies,
          and[equal[u, v], equal[w, subvar[u]]], equal[w, subvar[v]],
          {u → intersection[Di, x],
           v -> composite[id[fix[x]], complement[inverse[x]], id[fix[x]]], w → set[0]}]

Out[33]= or[not[equal[composite[id[fix[x]], complement[inverse[x]], id[fix[x]]],
             intersection[Di, x]]],
           not[WELLFOUNDED[intersection[Di, x]]], subclass[P[fix[x]], union[
             domain[LEAST[union[x, cart[V, complement[fix[x]]]]]], set[0]]]] == True

In[34]:= (% /. x → x_) /. Equal → SetDelayed
```

This result is not quite what we want, but it is in the right direction.

```
In[35]:= Map[not, SubstTest[and, implies[p1, p3],
           implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
           {p1 → TOTALORDER[x], p2 → WELLFOUNDED[intersection[Di, x]],
            p3 -> equal[composite[id[fix[x]], complement[inverse[x]], id[fix[x]]],
              intersection[Di, x]],
            p4 -> subclass[P[fix[x]], union[domain[
                LEAST[union[x, cart[V, complement[fix[x]]]]]], set[0]]]}]]

Out[35]= or[not[TOTALORDER[x]],
           not[WELLFOUNDED[intersection[Di, x]]], subclass[P[fix[x]], union[
             domain[LEAST[union[x, cart[V, complement[fix[x]]]]]], set[0]]]] == True

In[36]:= or[not[TOTALORDER[x_]],
           not[WELLFOUNDED[intersection[Di, x_]]], subclass[P[fix[x_]], union[
             domain[LEAST[union[x_, cart[V, complement[fix[x_]]]]]], set[0]]]] := True
```

The problem is that what one needs is just **LEAST[x],** and not **LEAST[union[x, cart[-V,complement[fix[x]]]]].**

---

lemmas

Introducing an extra variable **y** led to key simplifications needed to complete the proof started above.

```
In[37]:= lb[union[x, cart[V, complement[fix[x]]]], y] // Normality

Out[37]= lb[union[x, cart[V, complement[fix[x]]]], y] == lb[x, intersection[y, fix[x]]]

In[38]:= lb[union[x_, cart[V, complement[fix[x_]]]], y_] :=
          lb[x, intersection[y, fix[x]]]
```

```
In[39]:= SubstTest[implies, and[member[y, u], subclass[u, v]], member[y, v],
          {u → P[fix[x]],
           v -> union[domain[LEAST[union[x, cart[V, complement[fix[x]]]]]], set[0]]}]

Out[39]= or[equal[0, y],
           not[equal[0, intersection[y, lb[x, intersection[y, fix[x]]]]]],
           not[member[y, V]], not[subclass[y, fix[x]]], not[subclass[P[fix[x]], union[
              domain[LEAST[union[x, cart[V, complement[fix[x]]]]], set[0]]]]] == True

In[40]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed

In[41]:= SubstTest[implies, and[equal[z, y], disjoint[y, lb[x, y]]],
          disjoint[y, lb[x, z]], z → intersection[y, fix[x]]]

Out[41]= or[equal[0, intersection[y, lb[x, intersection[y, fix[x]]]]],
           not[equal[0, intersection[y, lb[x, y]]]], not[subclass[y, fix[x]]]] == True

In[42]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed

In[43]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
           implies[and[p3, p4], p5], implies[and[p4, p5], p6],
           not[implies[and[p1, p2, p4], p6]], {p1 → TOTALORDER[x],
            p2 → WELLFOUNDED[intersection[Di, x]], p3 -> subclass[P[fix[x]],
              union[domain[LEAST[union[x, cart[V, complement[fix[x]]]]], set[0]]],
            p4 → member[y, dif[P[fix[x]], set[0]]],
            p5 → not[equal[0, intersection[y, lb[x, intersection[y, fix[x]]]]]],
            p6 → not[equal[0, least[x, y]]]}]]

Out[43]= or[equal[0, y], not[equal[0, intersection[y, lb[x, y]]]],
           not[member[y, V]], not[subclass[y, fix[x]]], not[TOTALORDER[x]],
           not[WELLFOUNDED[intersection[Di, x]]]] == True

In[44]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

---

removing the variable y

```
In[45]:= class[y,
          and[equal[0, intersection[y, lb[x, y]]], member[y, V], subclass[y, fix[x]]]]

Out[45]= intersection[complement[domain[LEAST[x]]], P[fix[x]]]

In[46]:= and[equal[0, intersection[y, lb[x, y]]],
           member[y, V], not[equal[0, y]], subclass[y, fix[x]],
           TOTALORDER[x], WELLFOUNDED[intersection[Di, x]]] // NotNotTest

Out[46]= and[equal[0, intersection[y, lb[x, y]]],
           member[y, V], not[equal[0, y]], subclass[y, fix[x]],
           TOTALORDER[x], WELLFOUNDED[intersection[Di, x]]] == False
```

```
In[47]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The following takes a long time, even with flags off.

```
In[48]:= Map[equal[0, #] &,
           SubstTest[class, y, not[implies[and[member[y, u], TOTALORDER[x],
                subclass[P[intersection[Di, x]], w]], equal[0, y]]],
            {u -> intersection[complement[domain[LEAST[x]]], P[fix[x]]],
             w → WF}]] // Reverse

Out[48]= or[not[TOTALORDER[x]], not[WELLFOUNDED[intersection[Di, x]]],
           subclass[P[fix[x]], union[domain[LEAST[x]], set[0]]]] == True

In[49]:= (% /. x → x_) /. Equal → SetDelayed

In[52]:= Map[not, SubstTest[and, implies[p1, p3], implies[and[p1, p2], p4],
           implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]], {p1 → TOTALORDER[x],
            p2 → WELLFOUNDED[intersection[Di, x]], p3 → PARTIALORDER[x], p4 ->
             subclass[P[fix[x]], union[domain[LEAST[x]], set[0]]], p5 → WELLORDER[x]}]]

Out[52]= or[not[TOTALORDER[x]],
           not[WELLFOUNDED[intersection[Di, x]]], WELLORDER[x]] == True

In[53]:= (% /. x → x_) /. Equal → SetDelayed

In[55]:= equiv[and[TOTALORDER[x], WELLFOUNDED[intersection[Di, x]]], WELLORDER[x]] //
           not // not

Out[55]= True

In[57]:= and[TOTALORDER[x_], WELLFOUNDED[intersection[Di, x_]]] := WELLORDER[x]

In[58]:= or[not[TOTALORDER[x]], not[WELLFOUNDED[intersection[Di, x]]]] // NotNotTest

Out[58]= or[not[TOTALORDER[x]], not[WELLFOUNDED[intersection[Di, x]]]] ==
          not[WELLORDER[x]]

In[59]:= or[not[TOTALORDER[x_]], not[WELLFOUNDED[intersection[Di, x_]]]] :=
          not[WELLORDER[x]]
```

---

variable-free corollary

The theorem proved above has this corollary.

```
In[67]:= Map[equal[V, #] &, SubstTest[class, x, implies[member[x, u], member[x, v]], {u →
              intersection[TO, image[inverse[IMAGE[id[Di]]], WF]], v -> WO}]] // Reverse

Out[67]= subclass[intersection[TO, image[inverse[IMAGE[id[Di]]], WF]], WO] == True
```

```
In[68]:= subclass[intersection[TO, image[inverse[IMAGE[id[Di]]], WF]], WO] := True
```

The reverse inclusion is already known, so one can strengthen this to an equation.

```
In[70]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> intersection[TO, image[inverse[IMAGE[id[Di]]], WF]], v -> WO}]

Out[70]= True == equal[WO, intersection[TO, image[inverse[IMAGE[id[Di]]], WF]]]

In[72]:= intersection[TO, image[inverse[IMAGE[id[Di]]], WF]] := WO
```