WO and WF

Johan G. F. Belinfante
2004 March 14

```
In[1]:= << goedel55.14b; << tools.m;

        :Package Title: goedel55.14b          2004 March 14 at 1:30 p.m.

        It is now:  2004 Mar 15 at 12:33

        Loading Simplification Rules

        TOOLS.M                      Revised 2004 March 14

        weightlimit = 40
```

summary

It is shown that if **x** is a well−ordering, then **intersection[Di,x]** is well−founded.

review of well−founded relations

A relation **x** is well−founded if the empty set is the only set that is subvariant under **x**. This is embodied in the following rewrite rule:

```
In[2]:= equal[0, U[subvar[x]]]

Out[2]= WELLFOUNDED[composite[Id, x]]
```

Here is a new rule that is a variant of the above:

```
In[3]:= equal[singleton[0], subvar[x]] // AssertTest

Out[3]= equal[singleton[0], subvar[x]] == WELLFOUNDED[composite[Id, x]]

In[4]:= equal[singleton[0], subvar[x_]] := WELLFOUNDED[composite[Id, x]]
```

Two important examples are the membership relation, restricted to the class of regular sets, and the proper subclass relation, restricted to finite sets:

```
In[5]:= WELLFOUNDED[restrict[E, REGULAR, REGULAR]]

Out[5]= True

In[6]:= WELLFOUNDED[composite[id[FINITE], PS]]

Out[6]= True
```

The latter relation can be written another way:

```
In[7]:= composite[id[FINITE], PS, id[FINITE]] // VSNormality

Out[7]= composite[id[FINITE], PS, id[FINITE]] == composite[id[FINITE], PS]

In[8]:= composite[id[FINITE], PS, id[FINITE]] := composite[id[FINITE], PS]
```

WF

For the class **WF** of all small well−founded relations a new rewrite rule can be introduced:

```
In[9]:= image[inverse[IMAGE[id[cart[V, V]]]], WF] // Normality // Reverse

Out[9]= image[inverse[SUBVAR], singleton[singleton[0]]] ==
         image[inverse[IMAGE[id[cart[V, V]]]], WF]

In[10]:= image[inverse[SUBVAR], singleton[singleton[0]]] :=
          image[inverse[IMAGE[id[cart[V, V]]]], WF]
```

This rule replaces an existing rule, but needs to be supplemented with the following new rule to achieve normalization.

```
In[11]:= WF // Normality // Reverse

Out[11]= intersection[WF, P[cart[V, V]]] == WF

In[12]:= intersection[WF, P[cart[V, V]]] := WF
```

results about well−orderings

The first result holds for any antisymmetric relation:

```
In[13]:= SubstTest[implies, subclass[u, v], subclass[subvar[u], subvar[v]],
             {u -> intersection[Di, x], v -> complement[inverse[x]]}]

Out[13]= or[equal[0, intersection[domain[LEAST[x]], subvar[intersection[Di, x]]]],
           not[subclass[intersection[x, inverse[x]], Id]]] == True

In[14]:= (% /. x -> x_) /. Equal -> SetDelayed
```

An equality substitution rule for **disjoint** holds:

```
In[15]:= implies[and[equal[u, v], disjoint[v, w]], disjoint[u, w]] // AssertTest

Out[15]= or[equal[0, intersection[u, w]],
           not[equal[0, intersection[v, w]]], not[equal[u, v]]] == True

In[16]:= or[equal[0, intersection[u_, w_]],
           not[equal[0, intersection[v_, w_]]], not[equal[u_, v_]]] := True
```

Lemma.

```
In[17]:= SubstTest[implies, and[equal[u, v], disjoint[v, w]], disjoint[u, w],
           {u -> dif[P[fix[x]], singleton[0]],
          v -> domain[LEAST[x]], w -> subvar[intersection[Di, x]]}]

Out[17]= or[not[equal[0, intersection[domain[LEAST[x]], subvar[intersection[Di, x]]]]],
          not[equal[domain[LEAST[x]], intersection[complement[singleton[0]], P[fix[x]]]]],
          subclass[intersection[P[fix[x]], subvar[intersection[Di, x]]], singleton[0]]] == True

In[18]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Lemma.

```
In[19]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
          implies[p1, p4], implies[and[p3, p4], p5], not[implies[p1, p5]],
          {p1 -> WELLORDER[x], p2 -> subclass[intersection[x, inverse[x]], Id],
           p3 -> disjoint[domain[LEAST[x]], subvar[intersection[Di, x]]],
                 p4 -> equal[dif[P[fix[x]], singleton[0]], domain[LEAST[x]]],
                 p5 -> disjoint[dif[P[fix[x]], singleton[0]], subvar[intersection[Di, x]]]}]]

Out[19]= or[not[WELLORDER[x]],
          subclass[intersection[P[fix[x]], subvar[intersection[Di, x]]], singleton[0]]] == True

In[20]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Lemma.

```
In[21]:= SubstTest[subclass, U[subvar[w]], range[w], w -> intersection[Di, x]]

Out[21]= subclass[U[subvar[intersection[Di, x]]], fix[composite[Di, inverse[x]]]] == True

In[22]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Lemma.

```
In[23]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u -> U[subvar[intersection[Di, x]]],
          v -> range[intersection[Di, x]], w -> range[x]}]

Out[23]= subclass[U[subvar[intersection[Di, x]]], range[x]] == True

In[24]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Lemma.

```
In[25]:= implies[REFLEXIVE[x], equal[fix[x], range[x]]] // AssertTest

Out[25]= or[equal[fix[x], range[x]], not[subclass[x, cart[fix[x], fix[x]]]]] == True

In[26]:= or[equal[fix[x_], range[x_]], not[subclass[x_, cart[fix[x_], fix[x_]]]]] := True
```

Lemma.

```
In[27]:= SubstTest[implies, and[subclass[u, v], equal[v, w]], subclass[u, w],
           {u -> U[subvar[intersection[Di, x]]], v -> range[x], w -> fix[x]}]

Out[27]= or[not[equal[fix[x], range[x]]],
          subclass[U[subvar[intersection[Di, x]]], fix[x]]] == True

In[28]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Lemma.

```
In[29]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, p3], implies[p3, p4], not[implies[p1, p4]],
           {p1 -> WELLORDER[x], p2 -> REFLEXIVE[x], p3 -> equal[range[x], fix[x]],
            p4 -> subclass[U[subvar[intersection[Di, x]]], fix[x]]}]]

Out[29]= or[not[WELLORDER[x]], subclass[U[subvar[intersection[Di, x]]], fix[x]]] == True

In[30]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Lemma.

```
In[31]:= SubstTest[implies, and[equal[u, v], subclass[v, w]], subclass[u, w],
             {u -> subvar[intersection[Di, x]],
           v -> intersection[P[fix[x]], subvar[intersection[Di, x]]], w -> singleton[0]}]

Out[31]= or[not[subclass[intersection[P[fix[x]], subvar[intersection[Di, x]]], singleton[0]]],
           not[subclass[U[subvar[intersection[Di, x]]], fix[x]]],
           WELLFOUNDED[intersection[Di, x]]] == True

In[32]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Main theorem.

```
In[33]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
             {p1 -> WELLORDER[x],
            p2 -> subclass[intersection[P[fix[x]], subvar[intersection[Di, x]]], singleton[0]],
                 p3 -> subclass[U[subvar[intersection[Di, x]]], fix[x]],
            p4 -> WELLFOUNDED[intersection[Di, x]]}]]

Out[33]= or[not[WELLORDER[x]], WELLFOUNDED[intersection[Di, x]]] == True

In[34]:= or[not[WELLORDER[x_]], WELLFOUNDED[intersection[Di, x_]]] := True
```

variable−free version

For the case of small well−orderings, a variable−free version of the theorem can be derived:

```
In[35]:= Map[equal[V, #] &, SubstTest[class, x, not[member[x, y]],
            y -> dif[WO, image[inverse[IMAGE[id[Di]]], WF]]]] // Reverse

Out[35]= subclass[image[IMAGE[id[Di]], WO], WF] == True

In[36]:= subclass[image[IMAGE[id[Di]], WO], WF] := True
```

a related result

Temporary lemma.

```
In[37]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
             {u -> WO, v -> PO, w -> P[cart[V, V]]}]

Out[37]= subclass[U[WO], cart[V, V]] == True
```

```
In[38]:= % /. Equal -> SetDelayed
```

Lemma.

```
In[39]:= equal[intersection[P[cart[V, V]], WO], WO]
Out[39]= True
```

```
In[40]:= intersection[WO, P[cart[V, V]]] := WO
```

Lemma.

```
In[41]:= ImageComp[IMAGE[id[cart[V, V]]], id[P[cart[V, V]]], WO] // Reverse
Out[41]= image[IMAGE[id[cart[V, V]]], WO] == WO
```

```
In[42]:= image[IMAGE[id[cart[V, V]]], WO] := WO
```

Theorem.

```
In[43]:= SubstTest[subclass, WO, image[inverse[IMAGE[id[Di]]], x], x -> TRV] // Reverse
Out[43]= subclass[image[IMAGE[id[Di]], WO], TRV] == True
```

```
In[44]:= subclass[image[IMAGE[id[Di]], WO], TRV] := True
```

serendipity

The following additional results were discovered, but were not needed in the preceding derivations.

```
In[45]:= ImageComp[IMAGE[id[cart[V, V]]], id[P[cart[V, V]]], ANTISYM] // Reverse
Out[45]= image[IMAGE[id[cart[V, V]]], ANTISYM] == ANTISYM
```

```
In[46]:= image[IMAGE[id[cart[V, V]]], ANTISYM] := ANTISYM
```

```
In[47]:= ImageComp[IMAGE[id[cart[V, V]]], id[P[cart[V, V]]], PO] // Reverse
Out[47]= image[IMAGE[id[cart[V, V]]], PO] == PO
```

```
In[48]:= image[IMAGE[id[cart[V, V]]], PO] := PO
```

```
In[49]:= ImageComp[IMAGE[id[cart[V, V]]], id[P[cart[V, V]]], WF] // Reverse
Out[49]= image[IMAGE[id[cart[V, V]]], WF] == WF
```

```
In[50]:= image[IMAGE[id[cart[V, V]]], WF] := WF
```

```
In[51]:= SubstTest[subclass, TO, image[inverse[IMAGE[id[Di]]], x], x -> TRV] // Reverse
Out[51]= subclass[image[IMAGE[id[Di]], TO], TRV] == True
```

```
In[52]:= subclass[image[IMAGE[id[Di]], TO], TRV] := True
```