union of well-founded relations

Johan G. F. Belinfante
2004 September 23

```
In[1]:= SetDirectory["i:"]; << goedel61.21a; << tools.m

        :Package Title: goedel61.21a          2004 September 21 at 4:45 p.m.

        It is now:  2004 Sep 23 at 22:2

        Loading Simplification Rules

        TOOLS.M                      Revised 2004 September 18

        weightlimit = 40
```

summary

In general, the union of well-founded relations need not be well-founded. In this notebook it is shown that the union of two well-founded relations **x** and **y** is well-founded when **composite[x,y]** is contained in **y**. This fact implies a theorem about the lexicographic product of well-founded relations, and can therefore be viewed as a generalization of that theorem. The main theorem is derived first with **wf** wrappers for the well-founded relations. These wrappers are easily removed to obtain an equivalent formulation of the theorem in terms of the **WELLFOUNDED** predicate.

a counter-example

Technical lemma.

```
In[2]:= subclass[V, union[P[complement[singleton[0]]],
           P[complement[singleton[singleton[0]]]]]] // AssertTest

Out[2]= subclass[V, union[P[complement[singleton[0]]],
           P[complement[singleton[singleton[0]]]]]] == False

In[3]:= subclass[V, union[P[complement[singleton[0]]],
           P[complement[singleton[singleton[0]]]]]] := False
```

There is a simple criterion for a cartesian product to be well-founded:

```
In[4]:= WELLFOUNDED[cart[x, y]]

Out[4]= equal[0, intersection[x, y]]
```

Here is an example of a union of two well-founded relations that is not well-founded:

```
In[5]:= (WELLFOUNDED[union[cart[x, complement[x]], cart[y, complement[y]]]] //
           AssertTest) /. {x → singleton[0], y → singleton[singleton[0]]}

Out[5]= WELLFOUNDED[union[cart[singleton[0], complement[singleton[0]]], cart[
            singleton[singleton[0]], complement[singleton[singleton[0]]]]]] == False
```

---

lemma

The following lemma echos the definition of well-foundedness, but introduces an extra variable **x**, which will be convenient for the reasoning used in this notebook.

```
In[6]:= Map[implies[and[member[x, y], #], equal[0, x]] &,
         SubstTest[member, x, subvar[t], t → wf[z]] // Reverse]

Out[6]= or[equal[0, x], not[member[x, y]], not[subclass[x, image[wf[z], x]]]] == True

In[7]:= or[equal[0, x_], not[member[x_, y_]],
           not[subclass[x_, image[wf[z_], x_]]]] := True
```

The **wf** wrapper can be removed, if desired:

```
In[8]:= SubstTest[implies,
         and[equal[w, wf[z]], member[x, y], subvariant[w, x]], equal[0, x], w → z]

Out[8]= or[equal[0, x], not[member[x, y]],
           not[subclass[x, image[z, x]]], not[WELLFOUNDED[z]]] == True

In[9]:= or[equal[0, x_], not[member[x_, y_]],
           not[subclass[x_, image[z_, x_]]], not[WELLFOUNDED[z_]]] := True
```

---

a consequence of the special hypothesis: subclass[composite[x,y], y]

Lemma 1.

```
In[10]:= SubstTest[implies, subclass[u, v],
          subclass[image[u, z], image[v, z]], {u → composite[x, y], v → y}]

Out[10]= or[not[subclass[composite[x, y], y]],
            subclass[image[x, image[y, z]], image[y, z]]] == True
```

```
In[11]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Key lemma.

```
In[12]:= SubstTest[implies, equal[t, union[s, image[x, intersection[z, image[y, z]]]]],
           subclass[image[x, z], union[t, image[x, dif[z, image[y, z]]]]],
           {s → t}] /. t → image[y, z]

Out[12]= or[not[subclass[image[x, intersection[z, image[y, z]]], image[y, z]]],
           subclass[image[x, z], union[
             image[x, intersection[z, complement[image[y, z]]]], image[y, z]]]] == True
```

```
In[13]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Technical lemma.

```
In[14]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → dif[z, image[y, z]], v → dif[image[x, z], image[y, z]],
           w -> image[x, intersection[z, complement[image[y, z]]]]}]

Out[14]= or[not[subclass[z, union[image[x, z], image[y, z]]]],
           not[subclass[image[x, z],
             union[image[x, intersection[z, complement[image[y, z]]]], image[y, z]]]],
           subclass[z, union[image[x, intersection[z, complement[image[y, z]]]],
             image[y, z]]]] == True
```

```
In[15]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Putting these facts together yields:

```
In[16]:= Map[not, SubstTest[and, implies[p2, p3], implies[p3, p4],
           implies[p4, p5], implies[and[p1, p5], p6], not[implies[and[p1, p2], p6]],
           {p1 → subvariant[union[x, y], z], p2 → subclass[composite[x, y], y],
            p3 -> subclass[image[x, image[y, z]], image[y, z]],
            p4 -> subclass[image[x, intersection[z, image[y, z]]], image[y, z]],
            p5 → subclass[image[x, z],
              union[image[y, z], image[x, intersection[z, complement[image[y, z]]]]]],
            p6 → subvariant[x, dif[z, image[y, z]]]}]]

Out[16]= or[not[subclass[z, union[image[x, z], image[y, z]]]],
           not[subclass[composite[x, y], y]], subclass[z, union[
             image[x, intersection[z, complement[image[y, z]]]], image[y, z]]]] == True
```

```
In[17]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

derivation of the main theorem

First the condition that **x** is well-founded will be used. The following technical lemma is used to clean up an expression.

```
In[18]:= SubstTest[member, u, subvar[v], {u -> dif[z, image[y, z]], v → wf[x]}] // Reverse

Out[18]= and[member[intersection[z, complement[image[y, z]]], V], subclass[z, union[
           image[y, z], image[wf[x], intersection[z, complement[image[y, z]]]]]]] ==
         and[member[intersection[z, complement[image[y, z]]], V],
          subclass[z, image[y, z]]]

In[19]:= and[member[intersection[z_, complement[image[y_, z_]]], V],
           subclass[z_, union[image[y_, z_],
             image[wf[x_], intersection[z_, complement[image[y_, z_]]]]]]] :=
         and[member[intersection[z, complement[image[y, z]]], V],
          subclass[z, image[y, z]]]
```

The following lemma adds sethood literals.

```
In[20]:= ((implies[and[member[z, subvar[union[t, y]]], subclass[composite[t, y], y]],
              member[dif[z, image[y, z]], subvar[t]]] //
            NotNotTest) /. t → wf[x]) // MapNotNot

Out[20]= or[not[member[z, V]], not[subclass[z, union[image[y, z], image[wf[x], z]]]],
           not[subclass[composite[wf[x], y], y]], subclass[z, image[y, z]]] == True

In[21]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Restatement.

```
In[22]:= implies[and[member[z, subvar[union[wf[x], y]]],
           subclass[composite[wf[x], y], y]], member[z, subvar[y]]]

Out[22]= True
```

The fact that **y** is well-founded is used now.

```
In[23]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p4],
           implies[and[p1, p4], p5], not[implies[and[p1, p2, p3], p5]],
           {p1 → member[z, V], p2 → subvariant[union[wf[x], wf[y]], z],
            p3 → subclass[composite[wf[x], wf[y]], wf[y]],
            p4 → subvariant[wf[y], z], p5 → equal[0, z]}]]

Out[23]= or[equal[0, z], not[member[z, V]],
           not[subclass[z, union[image[wf[x], z], image[wf[y], z]]]],
           not[subclass[composite[wf[x], wf[y]], wf[y]]] == True

In[24]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Eliminating the variable **z**, one obtains the main theorem:

```
In[25]:= Map[equal[V, #] &,
          SubstTest[class, z, implies[and[subclass[u, v], member[z, w]], equal[0, z]],
            {u -> composite[wf[x], wf[y]], v → wf[y],
             w -> subvar[union[wf[x], wf[y]]]}] // Reverse]

Out[25]= or[not[subclass[composite[wf[x], wf[y]], wf[y]]],
           WELLFOUNDED[union[wf[x], wf[y]]]] == True

In[26]:= or[not[subclass[composite[wf[x_], wf[y_]], wf[y_]]],
           WELLFOUNDED[union[wf[x_], wf[y_]]]] := True
```

Removing the **wf** wrappers yields an alternative statement of the main theorem.

```
In[27]:= SubstTest[implies,
          and[equal[u, wf[x]], equal[v, wf[y]], subclass[composite[u, v], v]],
          WELLFOUNDED[union[u, v]], {u → x, v → y}]

Out[27]= or[not[subclass[composite[x, y], y]], not[WELLFOUNDED[x]],
           not[WELLFOUNDED[y]], WELLFOUNDED[union[x, y]]] == True

In[28]:= or[not[subclass[composite[x_, y_], y_]], not[WELLFOUNDED[x_]],
           not[WELLFOUNDED[y_]], WELLFOUNDED[union[x_, y_]]] := True
```

---

comment

This theorem can be used to show that the lexicographic product of two well-founded relations **x** and **y** is well-founded. To do so, one first shows that the relations **cross[x,V]** and **cross[Id,y]** are well-founded, and then one notices that the special condition studied in this notebook holds for these two cross products.

```
In[29]:= subclass[composite[cross[Id, y], cross[x, V]], cross[x, V]]
Out[29]= True
```

An independent derivation of this special case is given in the notebook **wf-lex.nb**.