lexicographic product of well-founded relations

Johan G. F. Belinfante
2004 September 23

```
In[1]:= SetDirectory["i:"]; << goedel61.23a; << tools.m

        :Package Title: goedel61.23a          2004 September 23 at 1:45 p.m.

        It is now:  2004 Sep 23 at 16:5

        Loading Simplification Rules

        TOOLS.M                      Revised 2004 September 18

        weightlimit = 40
```

summary

It is shown in this notebook that the lexicographic product of two well-founded relations is well-founded.

a basic lemma

The lemma derived in this section just says that if the domain of a relation is empty, then the relation itself is empty.

```
In[2]:= SubstTest[implies, and[equal[x, y], equal[y, z]],
         equal[x, z], {y → composite[Id, x], z → 0}]

Out[2]= or[equal[0, x], not[equal[0, domain[x]]], not[subclass[x, cart[V, V]]]] == True

In[3]:= (% /. x → x_) /. Equal → SetDelayed
```

The following is a slight generalization of this fact.

```
In[4]:= Map[not, SubstTest[and, implies[p1, p3],
          implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
          {p1 → subclass[x, cart[y, z]], p2 → equal[0, domain[x]],
           p3 → subclass[x, cart[V, V]], p4 → equal[0, x]}]]

Out[4]= or[equal[0, x], not[equal[0, domain[x]]], not[subclass[x, cart[y, z]]]] == True
```

```
In[5]:= or[equal[0, x_], not[equal[0, domain[x_]]],
          not[subclass[x_, cart[y_, z_]]]] := True
```

sethood lemmas

Lemma: If the domain of **x** is contained in **y**, then **image[x, complement[y]]** is empty, and is therefore a set.

```
In[6]:= SubstTest[implies, equal[0, z], member[z, V], z → image[x, complement[y]]]

Out[6]= or[member[image[x, complement[y]], V], not[subclass[domain[x], y]]] == True

In[7]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

A similar result:

```
In[8]:= SubstTest[member, z, singleton[0], z -> image[x, complement[y]]]

Out[8]= and[member[image[x, complement[y]], V], subclass[domain[x], y]] ==
         subclass[domain[x], y]

In[9]:= and[member[image[x_, complement[y_]], V], subclass[domain[x_], y_]] :=
         subclass[domain[x], y]
```

temporary definition

The **lexicographic product** of two relations is defined as follows:

```
In[10]:= class[pair[pair[t, u], pair[v, w]],
           or[member[pair[t, v], x], and[equal[t, v], member[pair[u, w], y]]]]

Out[10]= union[composite[inverse[FIRST], x, FIRST], cross[Id, y]]
```

The following temporary abbreviation will be introduced to increase readability:

```
In[11]:= lex[x_, y_] := union[composite[inverse[FIRST], x, FIRST], cross[Id, y]]
```

Note that the lexicographic product is a union of two cross-products:

```
In[12]:= lex[x, y] == union[cross[x, V], cross[Id, y]]

Out[12]= True
```

When **x** and **y** are well-founded, each of these two cross-products is individually well-founded, but in general the union of two well-founded relations need not be wellfounded. Nonetheless, it will be shown below that for the particular case of the lexicographic product, the union is well-founded.

---

derivation

Lemma.

```
In[13]:= SubstTest[implies, subclass[w, z], subclass[image[w, t], image[z, t]],
           {t -> complement[x], z -> union[cart[x, V], composite[y, w]]}]

Out[13]= or[not[subclass[w, union[cart[x, V], composite[y, w]]]],
            subclass[image[w, complement[x]], image[y, image[w, complement[x]]]]] == True

In[14]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

The definition of well-foundedness implies the following:

```
In[15]:= SubstTest[member, x, subvar[t], t → wf[y]] // Reverse

Out[15]= and[member[x, V], subclass[x, image[wf[y], x]]] == equal[0, x]

In[16]:= and[member[x_, V], subclass[x_, image[wf[y_], x_]]] := equal[0, x]
```

The following corollary follows because **wf[y]** is well-founded.

```
In[17]:= (implies[member[w, subvar[lex[wf[x], wf[y]]]],
             member[image[w, complement[image[wf[x], domain[w]]]], subvar[t]]] //
            NotNotTest) /. t → wf[y]

Out[17]= or[not[member[w, V]], not[
             subclass[w, union[cart[image[wf[x], domain[w]], V], composite[wf[y], w]]]],
            subclass[domain[w], image[wf[x], domain[w]]]] == True

In[18]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Using also the well-foundedness of **wf[x]**, one now deduces:

```
In[19]:= Map[not,
          SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
           not[implies[p1, p4]], {p1 → member[w, subvar[lex[wf[x], wf[y]]]],
            p2 → subvariant[wf[x], domain[w]],
            p3 → member[domain[w], V], p4 → equal[0, domain[w]]}]]

Out[19]= or[equal[0, domain[w]], not[member[w, V]], not[subclass[w,
             union[cart[image[wf[x], domain[w]], V], composite[wf[y], w]]]]] == True

In[20]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Since **w** is a relation, the conclusion **equal[0,domain[w]]** can be sharpened to **equal[0,w]**.

```
In[21]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
           implies[p1, p4], implies[and[p2, p3], p5], implies[and[p4, p5], p6],
           not[implies[p1, p6]], {p1 → member[w, subvar[lex[wf[x], wf[y]]]],
            p2 → subvariant[wf[x], domain[w]], p3 → member[domain[w], V],
            p4 → subclass[w, cart[V, V]], p5 → equal[0, domain[w]], p6 → equal[0, w]}]]

Out[21]= or[equal[0, w], not[member[w, V]], not[subclass[w,
             union[cart[image[wf[x], domain[w]], V], composite[wf[y], w]]]]] == True

In[22]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Removing the variable **w** yields the main result.

```
In[23]:= Map[equal[V, #] &, SubstTest[class, w, implies[member[w, z], equal[0, w]],
            z → subvar[lex[wf[x], wf[y]]]]] // Reverse

Out[23]= WELLFOUNDED[
           union[composite[inverse[FIRST], wf[x], FIRST], cross[Id, wf[y]]]] == True

In[24]:= WELLFOUNDED[
           union[composite[inverse[FIRST], wf[x_], FIRST], cross[Id, wf[y_]]]] := True
```

The wrappers can be removed, if desired:

```
In[25]:= SubstTest[implies, and[equal[u, wf[x]], equal[v, wf[y]]], WELLFOUNDED[
           union[composite[inverse[FIRST], u, FIRST], cross[Id, v]]], {u → x, v → y}]

Out[25]= or[not[WELLFOUNDED[x]], not[WELLFOUNDED[y]], WELLFOUNDED[
            union[composite[inverse[FIRST], x, FIRST], cross[Id, y]]]] == True

In[26]:= or[not[WELLFOUNDED[x_]], not[WELLFOUNDED[y_]], WELLFOUNDED[
            union[composite[inverse[FIRST], x_, FIRST], cross[Id, y_]]]] := True
```

some comments

When **x** and **y** are sets, the lexicographic product, as we have defined it, generally fails to be a set because of the presence of the global identity relation **Id**.

```
In[27]:= member[lex[setpart[x], setpart[y]], V] // assert

Out[27]= and[equal[0, domain[setpart[x]]], equal[0, domain[setpart[y]]]]
```

Since any subclass of a well-founded relation is well-founded, one can replace the global identity by any restriction of it, and the theorem remains true.

```
In[28]:= subclass[cross[id[z], y], union[x, cross[Id, y]]] // AssertTest

Out[28]= subclass[cross[id[z], y], union[x, cross[Id, y]]] == True

In[29]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed

In[30]:= SubstTest[implies, and[subclass[u, v], WELLFOUNDED[v]], WELLFOUNDED[u],
          {u -> union[composite[inverse[FIRST], wf[x], FIRST], cross[id[z], wf[y]]],
           v -> union[composite[inverse[FIRST], wf[x], FIRST], cross[Id, wf[y]]]}]

Out[30]= WELLFOUNDED[
           union[composite[inverse[FIRST], wf[x], FIRST], cross[id[z], wf[y]]]] == True

In[31]:= WELLFOUNDED[union[
            composite[inverse[FIRST], wf[x_], FIRST], cross[id[z_], wf[y_]]]] := True
```

Further such generalizations are possible: one could, for example, replace **composite[inverse[FIRST],wf[x],FIRST]** with **cross[wf[x], t]**, where **t** is arbitrary. The theorem itself is then the special case **t = V**.