subvariance under cross products

Johan G. F. Belinfante
2004 September 21

```
In[1]:= SetDirectory["i:"]; << goedel61.18b; << tools.m

        :Package Title: goedel61.18b          2004 September 18 at 6:30 p.m.

        It is now:  2004 Sep 21 at 15:8

        Loading Simplification Rules

        TOOLS.M                      Revised 2004 September 18

        weightlimit = 40
```

---

summary

It is is shown in this notebook that **cross[x,y]** is well-founded if either **x** or **y** is well-founded.

---

a theorem about subvar[cross[x,y]]

It will be shown in this section that if **z** is subvariant under **cross[x,y]**, then **domain[z]** is subvariant under **x**. One lemma is used:

```
In[2]:= SubstTest[implies, subclass[w, z],
         subclass[domain[w], domain[z]], z → composite[y, w, inverse[x]]]

Out[2]= or[not[subclass[w, composite[y, w, inverse[x]]]],
          subclass[domain[w], image[x, image[inverse[w], domain[y]]]]] == True

In[3]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

This is the main result:

```
In[4]:= Map[not,
         SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
          {p1 → subvariant[cross[x, y], w],
           p2 -> subclass[domain[w], image[x, image[inverse[w], domain[y]]]],
           p3 → subvariant[x, domain[w]]}]]

Out[4]= or[not[subclass[w, composite[y, w, inverse[x]]]],
          subclass[domain[w], image[x, domain[w]]]] == True

In[5]:= or[not[subclass[w_, composite[y_, w_, inverse[x_]]]],
          subclass[domain[w_], image[x_, domain[w_]]]] := True
```

A corresponding property of subvar[cross[x,y]] follows from this. One more lemma is needed.

```
In[8]:= implies[member[w, subvar[cross[x, y]]],
          member[domain[w], subvar[x]]] // NotNotTest

Out[8]= or[and[member[domain[w], V], subclass[domain[w], image[x, domain[w]]]],
          not[member[w, V]], not[subclass[w, composite[y, w, inverse[x]]]]] == True

In[9]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Corollary.

```
In[10]:= Map[equal[V, #] &,
           SubstTest[class, w, implies[member[w, u], member[domain[w], v]],
            {u → subvar[cross[x, y]], v → subvar[x]}]] // Reverse

Out[10]= subclass[image[IMAGE[FIRST], subvar[cross[x, y]]], subvar[x]] == True

In[11]:= subclass[image[IMAGE[FIRST], subvar[cross[x_, y_]]], subvar[x_]] := True
```

---

an upper bound for subvar[cross[x,y]]

The power class of the range of a relation **z** is an upper bound for **subvar[z]**. For the special case **z = cross[x,y]**, this yields a number of useful rewrite rules.

```
In[12]:= SubstTest[subclass, U[subvar[w]], range[w], w → cross[x, y]]

Out[12]= subclass[U[subvar[cross[x, y]]], cart[range[x], range[y]]] == True

In[13]:= subclass[U[subvar[cross[x_, y_]]], cart[range[x_], range[y_]]] := True
```

Corollary 1.

```
In[14]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u -> U[subvar[cross[x, y]]], v → cart[range[x], range[y]], w -> cart[V, V]}]
Out[14]= subclass[U[subvar[cross[x, y]]], cart[V, V]] == True

In[15]:= subclass[U[subvar[cross[x_, y_]]], cart[V, V]] := True
```

Corollary 2.

```
In[16]:= equal[composite[Id, U[subvar[cross[x, y]]]], U[subvar[cross[x, y]]]]
Out[16]= True

In[17]:= composite[Id, U[subvar[cross[x_, y_]]]] := U[subvar[cross[x, y]]]
```

Corollary 3.

```
In[18]:= equal[intersection[P[cart[V, V]], subvar[cross[x, y]]], subvar[cross[x, y]]]
Out[18]= True

In[19]:= intersection[P[cart[V, V]], subvar[cross[x_, y_]]] := subvar[cross[x, y]]
```

Corollary 4.

```
In[20]:= ImageComp[IMAGE[id[cart[V, V]]],
           id[P[cart[V, V]]], subvar[cross[x, y]]] // Reverse
Out[20]= image[IMAGE[id[cart[V, V]]], subvar[cross[x, y]]] == subvar[cross[x, y]]

In[21]:= image[IMAGE[id[cart[V, V]]], subvar[cross[x_, y_]]] := subvar[cross[x, y]]
```

---

relation between subvar[cross[x,y]] and subvar[cross[y,x]]

A formula relating **subvar[cross[x,y]]** to **subvar[cross[y,x]]** is derived in this section. Two lemmas are needed:

```
In[22]:= Map[implies[subvariant[cross[x, y], w], #] &, SubstTest[subclass,
           inverse[u], inverse[v], {u → w, v -> composite[y, w, inverse[x]]}]]
Out[22]= or[not[subclass[w, composite[y, w, inverse[x]]]],
           subclass[inverse[w], composite[x, inverse[w], inverse[y]]]] == True

In[23]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

```
In[24]:= implies[member[w, subvar[cross[x, y]]],
           member[inverse[w], subvar[cross[y, x]]]] // NotNotTest

Out[24]= or[and[member[domain[w], V], member[range[w], V],
            subclass[inverse[w], composite[x, inverse[w], inverse[y]]]],
           not[member[w, V]], not[subclass[w, composite[y, w, inverse[x]]]]] == True

In[25]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

The following inclusion will be sharpened to an equation momentarily:

```
In[26]:= Map[equal[V, #] &,
           SubstTest[class, w, implies[member[w, u], member[inverse[w], v]],
            {u → subvar[cross[x, y]], v → subvar[cross[y, x]]}]] // Reverse

Out[26]= subclass[image[IMAGE[SWAP], subvar[cross[x, y]]], subvar[cross[y, x]]] == True

In[27]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The reverse inclusion follows, making use of one of the rewrite rules derived in the preceding section.

```
In[28]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u -> image[IMAGE[SWAP], subvar[cross[x, y]]],
           v -> subvar[cross[y, x]], w → IMAGE[SWAP]}]

Out[28]= subclass[subvar[cross[x, y]], image[IMAGE[SWAP], subvar[cross[y, x]]]] == True

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

These inclusions are combined into an equation:

```
In[30]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[IMAGE[SWAP], subvar[cross[x, y]]], v -> subvar[cross[y, x]]}]

Out[30]= True == equal[image[IMAGE[SWAP], subvar[cross[x, y]]], subvar[cross[y, x]]]
```

The following rewrite rule is justified by this equation:

```
In[31]:= image[IMAGE[SWAP], subvar[cross[x_, y_]]] := subvar[cross[y, x]]
```

Corollary.

```
In[34]:= Map[subclass[#, subvar[y]] &,
          ImageComp[IMAGE[FIRST], IMAGE[SWAP], subvar[cross[x, y]]]]

Out[34]= subclass[image[IMAGE[SECOND], subvar[cross[x, y]]], subvar[y]] == True

In[35]:= subclass[image[IMAGE[SECOND], subvar[cross[x_, y_]]], subvar[y_]] := True
```

Corollary. (No new rewrite rule is needed here.)

```
In[46]:= subclass[image[DORA, subvar[cross[x, y]]], cart[subvar[x], subvar[y]]]

Out[46]= True
```

the main theorem

The following lemmas are needed to derive the main theorem.

```
In[48]:= SubstTest[subclass, u,
           image[inverse[IMAGE[FIRST]], v], v → singleton[0]] // Reverse

Out[48]= subclass[image[IMAGE[FIRST], u], singleton[0]] == equal[0, domain[U[u]]]

In[49]:= subclass[image[IMAGE[FIRST], u_], singleton[0]] := equal[0, domain[U[u]]]

In[50]:= SubstTest[equal, 0, composite[Id, u], u -> U[subvar[cross[x, y]]]] // Reverse

Out[50]= equal[0, domain[U[subvar[cross[x, y]]]]] == WELLFOUNDED[cross[x, y]]

In[51]:= equal[0, domain[U[subvar[cross[x_, y_]]]]] := WELLFOUNDED[cross[x, y]]

In[52]:= SubstTest[implies, and[subclass[u, v], equal[v, w]],
           subclass[u, w], {u → image[IMAGE[FIRST], subvar[cross[x, y]]],
            v → subvar[x], w -> singleton[0]}]

Out[52]= or[not[WELLFOUNDED[composite[Id, x]]], WELLFOUNDED[cross[x, y]]] == True

In[53]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Main theorem:

```
In[54]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, p3], not[implies[p1, p3]], {p1 → WELLFOUNDED[x],
            p2 → WELLFOUNDED[composite[Id, x]], p3 → WELLFOUNDED[cross[x, y]]}]]

Out[54]= or[not[WELLFOUNDED[x]], WELLFOUNDED[cross[x, y]]] == True

In[55]:= or[not[WELLFOUNDED[x_]], WELLFOUNDED[cross[x_, y_]]] := True
```

corollaries

```
In[56]:= SubstTest[implies, equal[u, v], equal[image[w, u], image[w, v]],
           {u → subvar[cross[x, y]], v → singleton[0], w → IMAGE[SWAP]}]

Out[56]= or[not[WELLFOUNDED[cross[x, y]]], WELLFOUNDED[cross[y, x]]] == True
```

```
In[57]:= or[not[WELLFOUNDED[cross[x_, y_]]], WELLFOUNDED[cross[y_, x_]]] := True

In[58]:= Map[not,
           SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
            {p1 → WELLFOUNDED[y], p2 → WELLFOUNDED[cross[y, x]],
             p3 → WELLFOUNDED[cross[x, y]]}]]

Out[58]= or[not[WELLFOUNDED[y]], WELLFOUNDED[cross[x, y]]] == True

In[59]:= or[not[WELLFOUNDED[y_]], WELLFOUNDED[cross[x_, y_]]] := True
```

---

variable-free versions

Lemma.

```
In[60]:= implies[and[member[x, WF], member[y, V]],
           member[cross[x, y], WF]] // NotNotTest

Out[60]= or[and[member[cross[x, y], V], WELLFOUNDED[cross[x, y]]],
           not[member[x, V]], not[member[y, V]], not[WELLFOUNDED[x]]] == True

In[61]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed

In[62]:= Map[equal[0, composite[Id, complement[#]]] &,
           SubstTest[class, pair[x, y], not[member[pair[x, y], z]],
            z -> dif[cart[WF, V], image[inverse[CROSS], WF]]]] // Reverse

Out[62]= subclass[image[CROSS, cart[WF, V]], WF] == True

In[63]:= subclass[image[CROSS, cart[WF, V]], WF] := True

In[64]:= Map[equal[0, composite[Id, complement[#]]] &,
           SubstTest[class, pair[x, y], not[member[pair[x, y], z]],
            z -> dif[cart[V, WF], image[inverse[CROSS], WF]]]] // Reverse

Out[64]= subclass[image[CROSS, cart[V, WF]], WF] == True

In[65]:= subclass[image[CROSS, cart[V, WF]], WF] := True
```