X[composite[x,y]]

Johan G. F. Belinfante
2004 November 12

```
In[1]:= SetDirectory["i:"]; << goedel63.11a; << tools.m

        :Package Title: goedel63.10b        2004 November 11 at 1:00 p.m.

        It is now:  2004 Nov 12 at 8:9

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 November 2

        weightlimit = 40
```

summary

It is shown that if **x** and **y** admit cross-sections, and if **range[y]** is contained in **domain[x]**, then **composite[x, y]** admits a cross-section. This result is useful for deriving the equivalence of various versions of the axiom of choice.

derivation

Lemma.

```
In[2]:= Map[not, SubstTest[and, implies[p3, p5], implies[p4, p6],
          implies[and[p1, p3, p6], p7], implies[p7, p8], implies[and[p2, p5, p8], p9],
          not[implies[and[p1, p2, p3, p4], p9]], {p1 -> equal[domain[u], domain[x]],
           p2 -> equal[domain[v], domain[y]], p3 -> subclass[range[y], domain[x]],
           p4 → subclass[v, y], p5 -> equal[image[inverse[y], domain[x]], domain[y]],
           p6 → subclass[range[v], range[y]], p7 → subclass[range[v], domain[u]],
           p8 -> equal[image[inverse[v], domain[u]], domain[v]],
           p9 -> equal[image[inverse[v], domain[u]], image[inverse[y], domain[x]]]}]]

Out[2]= or[equal[image[inverse[v], domain[u]], image[inverse[y], domain[x]]],
          not[equal[domain[u], domain[x]]], not[equal[domain[v], domain[y]]],
          not[subclass[v, y]], not[subclass[range[y], domain[x]]]] == True

In[3]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

For the next step one needs to turn off the **equality** flag.

```
In[4]:= equality = False;

In[5]:= implies[and[subclass[range[y], domain[x]], member[u, X[x]], member[v, X[y]]],
          member[composite[u, v], X[composite[x, y]]]] // NotNotTest

Out[5]= or[and[equal[image[inverse[v], domain[u]], image[inverse[y], domain[x]]],
           FUNCTION[composite[u, v]], member[composite[u, v], V],
           subclass[composite[u, v], composite[x, y]]],
          not[equal[domain[u], domain[x]]], not[equal[domain[v], domain[y]]],
          not[FUNCTION[u]], not[FUNCTION[v]], not[member[u, V]],
          not[member[v, V]], not[subclass[u, x]], not[subclass[v, y]],
          not[subclass[range[y], domain[x]]]] == True

In[6]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

The variables **u** and **v** are eliminated in the standard way.

```
In[7]:= Map[equal[0, composite[Id, complement[#]]] &,
          SubstTest[class, pair[u, v], implies[and[subclass[range[y], domain[x]],
             member[u, r], member[v, s]], member[composite[u, v], t]],
           {r → X[x], s → X[y], t → X[composite[x, y]]}]] // Reverse

Out[7]= or[not[subclass[range[y], domain[x]]],
          subclass[image[COMPOSE, cart[X[x], X[y]]], X[composite[x, y]]]] == True

In[8]:= or[not[subclass[range[y_], domain[x_]]],
          subclass[image[COMPOSE, cart[X[x_], X[y_]]], X[composite[x_, y_]]]] := True
```

Corollary. If **composite[x,y]** has no cross-section, and if **range[y]** is contained in **domain[x]**, then at least one of the classes **x** and **y** has no cross-section.

```
In[9]:= Map[implies[equal[0, X[composite[x, y]]], #] &, SubstTest[and,
           implies[p, subclass[u, v]], equal[0, v], {p → subclass[range[y], domain[x]],
            u -> image[COMPOSE, cart[X[x], X[y]]], v -> X[composite[x, y]]}]] // Reverse

Out[9]= or[equal[0, X[x]], equal[0, X[y]], not[equal[0, X[composite[x, y]]]],
          not[subclass[range[y], domain[x]]]] == True

In[10]:= or[equal[0, X[x_]], equal[0, X[y_]], not[equal[0, X[composite[x_, y_]]]],
           not[subclass[range[y_], domain[x_]]]] := True
```

---

an application

To demonstrate the usefulness of the corollary derived in the preceding section, consider the argument that led to the equivalence between two versions of the axiom of choice, **ac1** and **ac2**. One lemma is needed:

```
In[11]:= SubstTest[implies, and[subclass[u, v], member[v, V]], member[u, V],
          {u -> range[VERTSECT[setpart[x]]], v → P[range[setpart[x]]]}]

Out[11]= member[range[VERTSECT[setpart[x]]], V] == True

In[12]:= member[range[VERTSECT[setpart[x_]]], V] := True
```

The essential step in the proof of **ac1 <=> ac2** can now be deduced from the corollary in the preceding section as follows:

```
In[13]:= SubstTest[implies,
           and[subclass[range[v], domain[u]], equal[0, X[composite[u, v]]]],
           or[equal[0, X[u]], equal[0, X[v]]], {u → composite[inverse[E], id[t]],
            v → composite[id[t], VERTSECT[setpart[x]]]}] /.
          t -> dif[P[range[setpart[x]]], singleton[0]]

Out[13]= or[equal[0, X[composite[inverse[E], id[P[range[setpart[x]]]]]]],
           not[equal[0, X[setpart[x]]]]] == True

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

To remove the **setpart** wrappers, the **equality** flag needs to be set.

```
In[15]:= equality = True;

In[16]:= Map[implies[member[x, y], #] &,
          SubstTest[implies, equal[w, setpart[x]], or[equal[0,
             X[composite[inverse[E], id[P[range[w]]]]]], not[equal[0, X[w]]], w → x]]

Out[16]= or[equal[0, X[composite[inverse[E], id[P[range[x]]]]]],
           not[equal[0, X[x]]], not[member[x, y]]] == True

In[17]:= or[equal[0, X[composite[inverse[E], id[P[range[x_]]]]]],
           not[equal[0, X[x_]]], not[member[x_, y_]]] := True
```

The equivalence of the **ac2** and **ac3** versions of the axiom of choice can be derived in a similar fashion. To demonstrate this, a key step of that proof will be rederived. But first one needs to remove the rewrite rule in question:

```
In[18]:= or[equal[0, X[composite[id[x_], inverse[FIRST]]]], not[equal[0, X[x_]]]] =.
```

The following result is weaker, but still suffices to prove that **ac2 <=> ac3**. because for that one only needs to consider the case that **x** is a set.

In[19]:=
```
SubstTest[implies,
 and[subclass[range[v], domain[u]], equal[0, X[composite[u, v]]]],
 or[equal[0, X[u]], equal[0, X[v]]],
 {u → composite[SECOND, id[x]], v → composite[id[x], inverse[FIRST]]}]
```

Out[19]=
```
or[equal[0, X[composite[id[x], inverse[FIRST]]]], not[equal[0, X[x]]],
  not[member[domain[x], V]], not[member[range[x], V]]] == True
```