# monotone preserves greatest

Johan G. F. Belinfante
2007 June 17

```
In[1]:= SetDirectory["l:"]; << goedel94.16a; << tools.m

        :Package Title: goedel94.16a      2007 June 16 at 4:20 p.m.

        It is now:  2007 Jun 17 at 20:16

        Loading Simplification Rules

        TOOLS.M                                      Revised 2007 June 10

        weightlimit = 40
```

## summary

Monotone mappings of posets preserve greatest elements. A version of this familiar result is derived in this notebook that remains valid for arbitrary relations. A word of caution: the monotonicity condition can be formulated in various ways, which need not be equivalent in general.

## main theorem

Lemma. (Temporary rewrite rule.)

```
In[2]:= Map[composite[funpart[z], #] &, SubstTest[intersection,
          inverse[E], UB[t], t -> composite[inverse[funpart[z]], y, funpart[z]]]]

Out[2]= composite[funpart[z], GREATEST[composite[inverse[funpart[z]], y, funpart[z]]]] ==
         composite[GREATEST[y], IMAGE[funpart[z]], id[P[domain[funpart[z]]]]]

In[3]:= composite[funpart[z_], GREATEST[composite[inverse[funpart[z_]], y_, funpart[z_]]]] :=
         composite[GREATEST[y], IMAGE[funpart[z]], id[P[domain[funpart[z]]]]]
```

Lemma.

```
In[4]:= Map[or[#, subclass[composite[funpart[z], GREATEST[x]],
            composite[GREATEST[y], IMAGE[funpart[z]]]]] &,
          SubstTest[implies, subclass[u, v], subclass[composite[w, u], composite[w, v]],
           {u → GREATEST[x], v → GREATEST[t], w → funpart[z]}] // Reverse] /.
         t -> composite[inverse[funpart[z]], y, funpart[z]]

Out[4]= or[not[
           subclass[composite[id[fix[x]], x], composite[inverse[funpart[z]], y, funpart[z]]]],
          subclass[composite[funpart[z], GREATEST[x]],
           composite[GREATEST[y], IMAGE[funpart[z]]]] == True

In[5]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Main theorem.

```
In[6]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 -> subclass[x, composite[inverse[funpart[z]], y, funpart[z]]], p2 -> subclass[
              composite[id[fix[x]], x], composite[inverse[funpart[z]], y, funpart[z]]],
            p3 -> subclass[composite[funpart[z], GREATEST[x]],
              composite[GREATEST[y], IMAGE[funpart[z]]]]}]] // Reverse

Out[6]= or[not[subclass[x, composite[inverse[funpart[z]], y, funpart[z]]],
          subclass[composite[funpart[z], GREATEST[x]],
           composite[GREATEST[y], IMAGE[funpart[z]]]] == True

In[7]:= or[not[subclass[x_, composite[inverse[funpart[z_]], y_, funpart[z_]]]],
          subclass[composite[funpart[z_], GREATEST[x_]],
           composite[GREATEST[y_], IMAGE[funpart[z_]]]]] := True
```

Theorem.

```
In[8]:= Map[or[#, subclass[image[IMAGE[funpart[z]], domain[GREATEST[x]]],
             domain[GREATEST[y]]]] &, SubstTest[implies,
           subclass[u, v], subclass[domain[GREATEST[u]], domain[GREATEST[v]]],
           {u → x, v → composite[inverse[funpart[z]], y, funpart[z]]}]] // Reverse

Out[8]= or[not[subclass[x, composite[inverse[funpart[z]], y, funpart[z]]],
          subclass[image[IMAGE[funpart[z]], domain[GREATEST[x]]], domain[GREATEST[y]]]] == True

In[9]:= or[not[subclass[x_, composite[inverse[funpart[z_]], y_, funpart[z_]]]], subclass[
           image[IMAGE[funpart[z_]], domain[GREATEST[x_]]], domain[GREATEST[y_]]]] := True
```

## LEAST counterpart

Lemma.

```
In[10]:= Map[implies[subclass[x, composite[inverse[z], y, z]], #] &, SubstTest[
            subclass, inverse[x], inverse[t], t → composite[inverse[z], y, z]]] // Reverse

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

```
In[11]:= or[not[subclass[x_, composite[inverse[z_], y_, z_]]],
            subclass[inverse[x_], composite[inverse[z_], inverse[y_], z_]]] := True
```

Lemma.

```
In[12]:= SubstTest[implies, subclass[u, composite[inverse[funpart[z]], v, funpart[z]]],
            subclass[composite[funpart[z], GREATEST[u]],
             composite[GREATEST[v], IMAGE[funpart[z]]]],
            {u → inverse[x], v → inverse[y]}] // Reverse

Out[12]= or[not[subclass[inverse[x], composite[inverse[funpart[z]], inverse[y], funpart[z]]]],
            subclass[composite[funpart[z], LEAST[x]],
             composite[LEAST[y], IMAGE[funpart[z]]]]] == True

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

Theorem. (Analog of main theorem for the case of **LEAST**.)

```
In[14]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
              {p1 -> subclass[x, composite[inverse[funpart[z]], y, funpart[z]]], p2 ->
                subclass[inverse[x], composite[inverse[funpart[z]], inverse[y], funpart[z]]],
               p3 -> subclass[composite[funpart[z], LEAST[x]],
                 composite[LEAST[y], IMAGE[funpart[z]]]]}]] // Reverse

Out[14]= or[not[subclass[x, composite[inverse[funpart[z]], y, funpart[z]]], subclass[
              composite[funpart[z], LEAST[x]], composite[LEAST[y], IMAGE[funpart[z]]]]] == True

In[15]:= or[not[subclass[x_, composite[inverse[funpart[z_]], y_, funpart[z_]]]],
            subclass[composite[funpart[z_], LEAST[x_]],
             composite[LEAST[y_], IMAGE[funpart[z_]]]]] := True
```

Theorem.

```
In[16]:= Map[or[#, subclass[image[IMAGE[funpart[z]], domain[LEAST[x]]], domain[LEAST[y]]]] &,
            SubstTest[implies, subclass[u, v], subclass[domain[LEAST[u]], domain[LEAST[v]]],
             {u → x, v → composite[inverse[funpart[z]], y, funpart[z]]}]] // Reverse

Out[16]= or[not[subclass[x, composite[inverse[funpart[z]], y, funpart[z]]]],
            subclass[image[IMAGE[funpart[z]], domain[LEAST[x]]], domain[LEAST[y]]]] == True

In[17]:= or[not[subclass[x_, composite[inverse[funpart[z_]], y_, funpart[z_]]]],
            subclass[image[IMAGE[funpart[z_]], domain[LEAST[x_]]], domain[LEAST[y_]]]] := True
```

## caution

In each of the main theorems there occurs a hypothesis of the form:

```
In[18]:= hypothesis[z_, x_, y_] := subclass[x, composite[inverse[z], y, z]]
```

In the **GOEDEL** program, there are two related predicates called **monotone** and **subtwine**, neither of which by itself suffices for **hypothesis**.

```
In[19]:= monotone[z, x, y]

Out[19]= subclass[composite[z, x, inverse[z]], y]

In[20]:= subtwine[z, x, y]

Out[20]= subclass[composite[z, x], composite[y, z]]
```

The case **x = y = Id, z = 0** provides a simple couterexample to show that neither **monotone** nor **subtwine** implies **hypothesis** in general.

```
In[21]:= implies[monotone[0, Id, Id], hypothesis[0, Id, Id]]

Out[21]= False

In[22]:= implies[subtwine[0, Id, Id], hypothesis[0, Id, Id]]

Out[22]= False
```

For the case of posets, the **hypothesis** is satisfied when one adds an additional condition on the domain of **funpart[z]**.

```
In[23]:= implies[and[monotone[funpart[z], po[x], po[y]],
           subclass[fix[po[x]], domain[funpart[z]]]], hypothesis[funpart[z], po[x], po[y]]]

Out[23]= True

In[24]:= implies[and[subtwine[funpart[z], po[x], po[y]],
           subclass[fix[po[x]], domain[funpart[z]]]], hypothesis[funpart[z], po[x], po[y]]]

Out[24]= True
```