IMAGE[x] = IMAGE[y]

Johan G. F. Belinfante
2004 February 28

In[1]:= **<< goedel54.25a; << tools.m**

```
:Package Title: goedel54.25a     2004 February 25 at 8:11 p.m.

It is now:  2004 Mar 1 at 12:13

Loading Simplification Rules

TOOLS.M                       Revised 2004 February 21

weightlimit = 40
```

summary

The constructor **VERTSECT** is not one−to−one. For example **VERTSECT[E] = VERTSECT[S] = 0**, but **E** is obviously not equal to **S**. If **VERTSECT[x] = VERTSECT[y]**, then **thinpart[x] = thinpart[y]**, but a counterexample shows that the converse does not hold. On the other hand, **VERTSECT[x]** and **IMAGE[x]** determine one another, and therefore **IMAGE[x] = IMAGE[y]** is equivalent to **VERTSECT[x] = VERTSECT[y]**.

a thinpart counterexample

The **thinpart** of **x** is defined as follows:

In[2]:= **thinpart[x]**

Out[2]= composite[x, id[domain[VERTSECT[x]]]]

This is identical:

In[3]:= **composite[inverse[E], VERTSECT[x]]**

Out[3]= composite[x, id[domain[VERTSECT[x]]]]

Equality of **VERTSECT**'s implies equality of **thinpart**'s:

In[4]:= **SubstTest[implies, equal[u, v], equal[composite[w, u], composite[w, v]],
{u -> VERTSECT[x], v -> VERTSECT[y], w -> inverse[E]}]**

Out[4]= or[equal[composite[x, id[domain[VERTSECT[x]]]], composite[y, id[domain[VERTSECT[y]]]]],
not[equal[VERTSECT[x], VERTSECT[y]]]] == True

The converse is not true; the following counterexample shows that the equality of **thinpart**'s of a relation does not imply equality of their **VERTSECT**'s

```
In[5]:= implies[equal[thinpart[x], thinpart[y]], equal[VERTSECT[x], VERTSECT[y]]] /.
          {x ->
         union[cart[complement[succ[singleton[0]]], V], cart[singleton[0], singleton[0]]],
              y -> union[cart[complement[singleton[0]], V], cart[singleton[0], singleton[0]]]}

Out[5]= False
```

IMAGE equality

Equality of **IMAGE**'s implies equality of **VERTSECT**'s:

```
In[6]:= SubstTest[implies, equal[u, v], equal[composite[u, w], composite[v, w]],
          {u -> IMAGE[x], v -> IMAGE[y], w -> SINGLETON}]

Out[6]= or[equal[VERTSECT[x], VERTSECT[y]], not[equal[IMAGE[x], IMAGE[y]]]] == True

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

The converse also holds because one construct **IMAGE[x]** out of **VERTSECT[x]** as follows:

```
In[8]:= composite[BIGCUP, IMAGE[VERTSECT[x]], id[P[domain[VERTSECT[x]]]]]

Out[8]= IMAGE[x]
```

details of the proof

The somewhat tedious details of the proof of the assertion made in the preceding section are taken care of in this section.

```
In[9]:= SubstTest[implies, equal[u, v], equal[composite[w, u], composite[w, v]],
          {u -> id[P[x]], v -> id[P[y]]}]

Out[9]= or[equal[composite[w, id[P[x]]], composite[w, id[P[y]]]], not[equal[x, y]]] == True

In[10]:= (% /. {w -> w_, x -> x_, y -> y_}) /. Equal -> SetDelayed

In[11]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[p2, p4],
            implies[p3, p5], implies[and[p4, p5], p6], not[implies[p1, p6]],
             {p1 -> equal[VERTSECT[x], VERTSECT[y]],
             p2 -> equal[domain[VERTSECT[x]], domain[VERTSECT[y]]],
                 p3 -> equal[IMAGE[VERTSECT[x]], IMAGE[VERTSECT[y]]],
                 p4 -> equal[composite[w, IMAGE[VERTSECT[x]], id[P[domain[VERTSECT[x]]]]],
                        composite[w, IMAGE[VERTSECT[x]], id[P[domain[VERTSECT[y]]]]]],
                 p5 -> equal[composite[w, IMAGE[VERTSECT[x]], id[P[domain[VERTSECT[y]]]]],
                        composite[w, IMAGE[VERTSECT[y]], id[P[domain[VERTSECT[y]]]]]],
                 p6 -> equal[composite[w, IMAGE[VERTSECT[x]], id[P[domain[VERTSECT[x]]]]],
                        composite[w, IMAGE[VERTSECT[y]], id[P[domain[VERTSECT[y]]]]]]}]] /.
          w -> BIGCUP

Out[11]= or[equal[IMAGE[x], IMAGE[y]], not[equal[VERTSECT[x], VERTSECT[y]]]] == True

In[12]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

a new rewrite rule

The two implications derived above can be combined into a new rewrite rule.

```
In[13]:= equiv[equal[IMAGE[x], IMAGE[y]], equal[VERTSECT[x], VERTSECT[y]]]

Out[13]= True
```

The orientation of the rewrite rule is tentatively chosen as follows:

```
In[14]:= equal[VERTSECT[x_], VERTSECT[y_]] := equal[IMAGE[x], IMAGE[y]]
```

This is done to preserve an existing rewrite rule that deals with this special case:

```
In[15]:= equal[IMAGE[id[x]], IMAGE[id[y]]]

Out[15]= equal[x, y]
```

The equality of **thinpart**'s derived in an earlier section is transformed by the new rule to the following:

```
In[16]:= SubstTest[implies, equal[u, v], equal[composite[w, u], composite[w, v]],
             {u -> VERTSECT[x], v -> VERTSECT[y], w -> inverse[E]}]

Out[16]= or[equal[composite[x, id[domain[VERTSECT[x]]]],
            composite[y, id[domain[VERTSECT[y]]]]], not[equal[IMAGE[x], IMAGE[y]]]] == True

In[17]:= or[equal[composite[x_, id[domain[VERTSECT[x_]]]],
            composite[y_, id[domain[VERTSECT[y_]]]]], not[equal[IMAGE[x_], IMAGE[y_]]]] := True
```

The equality substitution rule for **VERTSECT** now follows from that for **IMAGE**, and will therefore be removed from the **GOEDEL** program.