associative[composite[IMAGE[x],CART]]

Johan G. F. Belinfante
2003 July 19

```
In[1]:= << goedel52.s56; << tools.m

        :Package Title: goedel52.s56       2003 July 17 at 5:05 p.m.

        It is now:  2003 Jul 24 at 13:12

        Loading Simplification Rules

        TOOLS.M                      Revised 2003 July 8

        weightlimit = 40
```

■ summary

In this notebook it is shown that if **x** is a thin associative relation, then **composite[IMAGE[x],CART]** is associative. The latter is a function.

■ lemma about IMAGE[ASSOC]

```
In[2]:= Map[composite[VERTSECT[#], id[cart[cart[V, V], V]]] &,
          SubstTest[composite, ASSOC, cross[cross[x, x], x], x -> inverse[E]]]

Out[2]= composite[IMAGE[ASSOC], CART, cross[CART, Id]] == composite[CART, cross[Id, CART], ASSOC]

In[3]:= composite[IMAGE[ASSOC], CART, cross[CART, Id]] := composite[CART, cross[Id, CART], ASSOC]
```

■ thinpart

The old definition of **thinpart** will be removed:

```
In[4]:= thinpart[x]

Out[4]= composite[x, id[domain[VERTSECT[x]]]]

In[5]:= thinpart[x_] =.
```

discovering a replacement:

```
In[6]:= member[x, composite[y, id[domain[VERTSECT[y]]]]] // AssertTest

Out[6]= member[x, composite[y, id[domain[VERTSECT[y]]]]] == and[member[first[x], V],
          member[image[y, singleton[first[x]]], V], member[pair[first[x], second[x]], y]]
```

The new definition:

```
In[7]:= member[x_, thinpart[y_]] := and[member[first[x], V],
          member[image[y, singleton[first[x]]], V], member[pair[first[x], second[x]], y]]
```

The old definition is deduced, but is turned around:

```
In[8]:= thinpart[x] // Normality // Reverse

Out[8]= composite[x, id[domain[VERTSECT[x]]]] == thinpart[x]

In[9]:= composite[x_, id[domain[VERTSECT[x_]]]] := thinpart[x]

In[10]:= SubstTest[VERTSECT, composite[x, id[y]], y -> domain[VERTSECT[x]]]

Out[10]= VERTSECT[thinpart[x]] ==
          union[cart[complement[domain[VERTSECT[x]]], singleton[0]], VERTSECT[x]]

In[11]:= VERTSECT[thinpart[x_]] :=
          union[cart[complement[domain[VERTSECT[x]]], singleton[0]], VERTSECT[x]]

In[12]:= domain[VERTSECT[thinpart[x]]]

Out[12]= V

In[13]:= composite[thinpart[x], inverse[E]]

Out[13]= composite[inverse[E], IMAGE[thinpart[x]]]

In[14]:= equal[x, thinpart[x]] // AssertTest

Out[14]= equal[x, thinpart[x]] == and[equal[V, domain[VERTSECT[x]]], subclass[x, cart[V, V]]]

In[15]:= equal[x_, thinpart[x_]] := and[equal[V, domain[VERTSECT[x]]], subclass[x, cart[V, V]]]

In[16]:= equal[composite[Id, x], thinpart[x]] // AssertTest

Out[16]= equal[composite[Id, x], thinpart[x]] == equal[V, domain[VERTSECT[x]]]

In[17]:= equal[composite[Id, x_], thinpart[x_]] := equal[V, domain[VERTSECT[x]]]
```

The following is true, but will not be needed:

```
In[18]:= Map[VERTSECT, Assoc[x, id[domain[VERTSECT[x]]], inverse[E]]]

Out[18]= composite[IMAGE[x], IMAGE[id[domain[VERTSECT[x]]]]] == IMAGE[thinpart[x]]
```

This is more useful:

```
In[19]:= SubstTest[implies, equal[y, z],
          equal[IMAGE[y], IMAGE[z]], {y -> composite[Id, x], z -> thinpart[x]}]

Out[19]= or[equal[IMAGE[x], IMAGE[thinpart[x]]], not[equal[V, domain[VERTSECT[x]]]]] == True

In[20]:= or[equal[IMAGE[x_], IMAGE[thinpart[x_]]], not[equal[V, domain[VERTSECT[x_]]]]] := True
```

■ derivation using thinpart

```
In[21]:= SubstTest[implies, equal[u, v], equal[composite[u, w], composite[v, w]],
           {u → composite[thinpart[x], cross[thinpart[x], Id]],
          v → composite[thinpart[x], cross[Id, thinpart[x]], ASSOC],
          w → composite[inverse[E], CART, cross[CART, Id]]}]

Out[21]= or[equal[composite[inverse[E], IMAGE[thinpart[x]], CART,
            cross[composite[IMAGE[thinpart[x]], CART], Id]], composite[inverse[E],
            IMAGE[thinpart[x]], CART, cross[Id, composite[IMAGE[thinpart[x]], CART]], ASSOC]],
          not[equal[composite[thinpart[x], cross[thinpart[x], Id]],
            composite[thinpart[x], cross[Id, thinpart[x]], ASSOC]]]] == True

In[22]:= (% /. x -> x_) /. Equal -> SetDelayed

In[23]:= implies[equal[u, v],
           equal[composite[VERTSECT[u], id[w]], composite[VERTSECT[v], id[w]]]]

Out[23]= or[equal[composite[VERTSECT[u], id[w]], composite[VERTSECT[v], id[w]]],
           not[equal[u, v]]]

In[24]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
                {p1 -> equal[x, y], p2 -> equal[VERTSECT[x], VERTSECT[y]],
             p3 -> equal[composite[VERTSECT[x], id[z]], composite[VERTSECT[y], id[z]]]}]]

Out[24]= or[equal[composite[VERTSECT[x], id[z]], composite[VERTSECT[y], id[z]]],
            not[equal[x, y]]] == True

In[25]:= or[equal[composite[VERTSECT[x_], id[z_]], composite[VERTSECT[y_], id[z_]]],
            not[equal[x_, y_]]] := True

In[26]:= SubstTest[implies, equal[u, v],
           equal[composite[VERTSECT[u], id[w]], composite[VERTSECT[v], id[w]]],
             {u -> composite[inverse[E], IMAGE[thinpart[x]],
             CART, cross[composite[IMAGE[thinpart[x]], CART], Id]],
            v -> composite[inverse[E], IMAGE[thinpart[x]], CART,
              cross[Id, composite[IMAGE[thinpart[x]], CART]], ASSOC],
                w -> cart[cart[V, V], V]}]

Out[26]= or[equal[composite[IMAGE[thinpart[x]], CART,
            cross[composite[IMAGE[thinpart[x]], CART], Id]], composite[IMAGE[thinpart[x]],
            CART, cross[Id, composite[IMAGE[thinpart[x]], CART]], ASSOC]],
          not[equal[composite[inverse[E], IMAGE[thinpart[x]], CART,
             cross[composite[IMAGE[thinpart[x]], CART], Id]],
            composite[inverse[E], IMAGE[thinpart[x]], CART,
             cross[Id, composite[IMAGE[thinpart[x]], CART]], ASSOC]]] == True

In[27]:= (% /. x -> x_) /. Equal -> SetDelayed

In[28]:= SubstTest[implies, and[subclass[z, cart[cart[V, V], V]], equal[
            composite[z, cross[z, Id]], composite[z, cross[Id, z], ASSOC]]], associative[z],
             z -> composite[IMAGE[x], CART]]

Out[28]= or[associative[composite[IMAGE[x], CART]],
            not[equal[composite[IMAGE[x], CART, cross[composite[IMAGE[x], CART], Id]],
              composite[IMAGE[x], CART, cross[Id, composite[IMAGE[x], CART]], ASSOC]]]] == True

In[29]:= (% /. x -> x_) /. Equal -> SetDelayed
```

```
In[30]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           implies[p3, p4], implies[p4, p5], not[implies[p1, p5]],
                 {p1 -> associative[thinpart[x]],
            p2 -> equal[composite[thinpart[x], cross[thinpart[x], Id]],
              composite[thinpart[x], cross[Id, thinpart[x]], ASSOC]],
                    p3 -> equal[composite[inverse[E], IMAGE[thinpart[x]],
             CART, cross[composite[IMAGE[thinpart[x]], CART], Id]],
              composite[inverse[E], IMAGE[thinpart[x]], CART,
               cross[Id, composite[IMAGE[thinpart[x]], CART]], ASSOC]],
                    p4 -> equal[composite[IMAGE[thinpart[x]], CART,
               cross[composite[IMAGE[thinpart[x]], CART], Id]], composite[IMAGE[thinpart[x]],
               CART, cross[Id, composite[IMAGE[thinpart[x]], CART]], ASSOC]],
                    p5 -> associative[composite[IMAGE[thinpart[x]], CART]]}]]

Out[30]= or[associative[composite[IMAGE[thinpart[x]], CART]],
           not[associative[thinpart[x]]]] == True

In[31]:= or[associative[composite[IMAGE[thinpart[x_]], CART]],
           not[associative[thinpart[x_]]]] := True
```

■ eliminating thinpart

```
In[32]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
                 {p1 -> associative[x],
            p2 -> subclass[x, cart[cart[V, V], V]], p3 -> subclass[x, cart[V, V]]}]]

Out[32]= or[not[associative[x]], subclass[x, cart[V, V]]] == True

In[33]:= or[not[associative[x_]], subclass[x_, cart[V, V]]] := True

In[34]:= equiv[or[not[associative[x]], not[subclass[x, cart[V, V]]]], not[associative[x]]]

Out[34]= True

In[35]:= or[not[associative[x_]], not[subclass[x_, cart[V, V]]]] := not[associative[x]]

In[36]:= SubstTest[implies, and[equal[x, y], associative[x]], associative[y], y -> thinpart[x]]

Out[36]= or[associative[thinpart[x]],
           not[associative[x]], not[equal[V, domain[VERTSECT[x]]]]] == True

In[37]:= or[associative[thinpart[x_]],
           not[associative[x_]], not[equal[V, domain[VERTSECT[x_]]]]] := True

In[38]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[p3, p4],
           implies[p2, p5], implies[p5, p6], implies[and[p4, p6], p7],
                 not[implies[and[p1, p2], p7]],
              {p1 -> associative[x], p2 -> thin[x], p3 -> associative[thinpart[x]],
                    p4 -> associative[composite[IMAGE[thinpart[x]], CART]],
                    p5 -> equal[IMAGE[x], IMAGE[thinpart[x]]],
                 p6 ->
               equal[composite[IMAGE[x], CART], composite[IMAGE[thinpart[x]], CART]],
                    p7 -> associative[composite[IMAGE[x], CART]]}]]

Out[38]= or[associative[composite[IMAGE[x], CART]],
           not[associative[x]], not[equal[V, domain[VERTSECT[x]]]]] == True

In[39]:= or[associative[composite[IMAGE[x_], CART]],
           not[associative[x_]], not[equal[V, domain[VERTSECT[x_]]]]] := True
```

■ two examples

Example 1.

```
In[40]:= associative[composite[SWAP, RIF]]
Out[40]= True

In[41]:= thin[composite[SWAP, RIF]]
Out[41]= True

In[42]:= composite[IMAGE[composite[SWAP, RIF]], CART]
Out[42]= COMPOSE

In[43]:= associative[COMPOSE]
Out[43]= True
```

Example 2.

```
In[44]:= associative[inverse[DUP]]
Out[44]= True

In[45]:= thin[inverse[DUP]]
Out[45]= True

In[46]:= composite[IMAGE[inverse[DUP]], CART]
Out[46]= CAP

In[47]:= associative[CAP]
Out[47]= True
```