associative relations

Johan G. F. Belinfante
2003 July 2

```
In[1]:= << goedel52.s31; << tools.m

        :Package Title: goedel52.s31       2003 July 2 at 9:50 a.m.

        It is now:  2003 Jul 6 at 22:35

        Loading Simplification Rules

        TOOLS.M                     Revised 2003 July 2

        weightlimit = 40
```

■ summary

A formula for the class of all associative relations is verified.

■ preliminaries

To speed up the computation, some flags are turned off.

```
In[2]:= cond = False; simplify = False;
```

The class of associative relations is defined by this membership rule:

```
In[3]:= member[x, ASSOCIATIVE]

Out[3]= and[equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]],
         member[x, V], subclass[x, cart[cart[V, V], V]]]
```

■ lemmas

To clean up some formulas, it helps to add this rule:

```
In[4]:= equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]] // AssertTest //
         Reverse

Out[4]= and[subclass[composite[x, cross[Id, x], ASSOC], composite[x, cross[x, Id]]],
          subclass[composite[x, cross[x, Id], inverse[ASSOC]], composite[x, cross[Id, x]]],
          subclass[image[inverse[x], domain[domain[x]]], cart[V, V]]] ==
         equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]
```

```
In[5]:= and[subclass[
           composite[x_, cross[Id, x_], ASSOC], composite[x_, cross[x_, Id]]], subclass[
           composite[x_, cross[x_, Id], inverse[ASSOC]], composite[x_, cross[Id, x_]]],
          subclass[image[inverse[x_], domain[domain[x_]]], cart[V, V]]] :=
         equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]
```

A formula discovered earlier this morning can almost be verified at this point:

```
In[6]:= member[x, intersection[P[cart[cart[V, V], V]],
        fix[image[inverse[CART], fix[composite[inverse[IMAGE[
                 composite[SWAP, RIF, cross[Id, composite[cross[SWAP, Id], inverse[RIF]]]]]],
               IMAGE[composite[SWAP, RIF,
                 cross[Id, composite[cross[inverse[ASSOC], SWAP], inverse[RIF]]]]]]]]]]] //
         assert

Out[6]= and[equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]],
         member[x, V], member[image[domain[x], range[x]], V],
         subclass[x, cart[cart[V, V], V]]]
```

The following theorem cleans this up further:

```
In[7]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
          {p1 -> member[x, V], p2 -> member[domain[x], V],
           p3 -> member[image[domain[x], y], V]}]]

Out[7]= or[member[image[domain[x], y], V], not[member[x, V]]] == True

In[8]:= or[member[image[domain[x_], y_], V], not[member[x_, V]]] := True
```

Thus:

```
In[9]:= equiv[and[member[x, V], member[image[domain[x], y], V]], member[x, V]]

Out[9]= True
```

The simpliest procedure is to make this a temporary rewrite rule:

```
In[10]:= and[member[x_, V], member[image[domain[x_], y_], V]] := member[x, V]
```

■ verification

```
In[11]:= (ASSOCIATIVE // Normality // Reverse) /. Equal -> SetDelayed
```

```
In[12]:= intersection[P[cart[cart[V, V], V]],
          fix[image[inverse[CART], fix[composite[inverse[IMAGE[
                composite[SWAP, RIF, cross[Id, composite[cross[SWAP, Id], inverse[RIF]]]]]],
              IMAGE[composite[SWAP, RIF,
                cross[Id, composite[cross[inverse[ASSOC], SWAP], inverse[RIF]]]]]]]]]] //
         Renormality

Out[12]= intersection[fix[image[inverse[CART], fix[composite[inverse[IMAGE[
                composite[SWAP, RIF, cross[Id, composite[cross[SWAP, Id], inverse[RIF]]]]]],
              IMAGE[composite[SWAP, RIF,
                cross[Id, composite[cross[inverse[ASSOC], SWAP], inverse[RIF]]]]]]]]],
          P[cart[cart[V, V], V]]] ==
         ASSOCIATIVE

In[13]:= intersection[fix[image[inverse[CART], fix[composite[inverse[IMAGE[
                composite[SWAP, RIF, cross[Id, composite[cross[SWAP, Id], inverse[RIF]]]]]],
              IMAGE[composite[SWAP, RIF,
                cross[Id, composite[cross[inverse[ASSOC], SWAP], inverse[RIF]]]]]]]]],
          P[cart[cart[V, V], V]]] :=
         ASSOCIATIVE
```

■ some clues about how the formula was discovered

The following results were discovered using reification. These formulas formed the basis of the discovery of the formula verified above. The two functions inside **IMAGE** in the formula above convert **cart[x,x]** to the left and right sides of the associativity condition.

```
In[14]:= image[composite[SWAP, RIF, cross[Id, composite[cross[SWAP, Id], inverse[RIF]]]],
         cart[x, x]]

Out[14]= composite[x, cross[x, Id]]

In[15]:= image[composite[SWAP, RIF,
          cross[Id, composite[cross[inverse[ASSOC], SWAP], inverse[RIF]]]],
         cart[x, x]]

Out[15]= composite[x, cross[Id, x], ASSOC]
```

This observation allows one to consider the associative law as a condition satisfied by **cart[x,x]**, thereby effectively replacing four occurrences of the variable **x** in the associative law by two occurrences of the cartesian square **cart[x,x]**. The formula says that x is associative if its cartesian square is a fixed point of some relation.

■ thin-ness results

The two relations involved in the formula for **ASSOCIATIVE** are thin:

```
In[16]:= domain[VERTSECT[
            composite[SWAP, RIF, cross[Id, composite[cross[SWAP, Id], inverse[RIF]]]]]] //
          Normality

Out[16]= domain[VERTSECT[
            composite[SWAP, RIF, cross[Id, composite[cross[SWAP, Id], inverse[RIF]]]]]] ==
          V

In[17]:= domain[VERTSECT[composite[SWAP, RIF,
             cross[Id, composite[cross[inverse[ASSOC], SWAP], inverse[RIF]]]]]] //
          Normality

Out[17]= domain[VERTSECT[composite[SWAP, RIF,
             cross[Id, composite[cross[inverse[ASSOC], SWAP], inverse[RIF]]]]]] ==
          V
```