pointwise binary operations

Johan G. F. Belinfante
2006 August 6

```
In[1]:= SetDirectory["l:"]; << goedel84.05c; << tools.m

        :Package Title: goedel84.05c        2006 August 5 at 9:40 p.m.

        It is now:  2006 Aug 6 at 17:39

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 July 18

        weightlimit = 40
```

summary

Given some binary operation such as addition of numbers, one can define a corresponding pointwise binary operation on number-valued functions: if **f** and **g** are functions, then their pointwise sum **h = f + g** is defined by **h(x) = f(x) + g(x)**. If the original binary operation is associative, then this new pointwise binary operation on functions inherits the associative property. In this notebook, this idea is extended to associative relations.

some basic observations

The equation **h(x) = f(x) + g(x)** has two copies of **x** on the right side. To apply **h** to **x**, one must first duplicate **x**, then apply **f** and **g** in parallel to the two copies of **x**, and finally apply the binary operation to the result. Suppose, for example, that the binary operation **+** is addition of natural numbers. This operation is represented in the **GOEDEL** program by the function **NATADD**.

```
In[2]:= member[NATADD, map[cart[omega, omega], omega]]

Out[2]= True
```

The formula for **h** can now be expressed without explicit mention of the variable **x** as follows: **h = composite[NATADD, cross[f, g], DUP]**. The function that takes **pair[f, g]** to **h** is therefore the composite of the function **CROSS** which takes **pair[f, g]** to **w = cross[f, g]** and the function which takes **w** to **composite[NATADD, w, DUP]**. The latter function is:

```
In[3]:= lambda[w, composite[u, w, v]] /. {u → NATADD, v → DUP}

Out[3]= IMAGE[cross[inverse[DUP], NATADD]]
```

Thus pointwise addition of functions is given by the binary function **composite[IMAGE[cross[inverse[DUP], NATADD]], CROSS]**. It will be shown below that this function inherits the associative property of the function **NATADD**. More

generally, it will be shown below that if **x** is any thin associative relation, then the corresponding function **composite[-IMAGE[cross[inverse[DUP], x]], CROSS]** is associative.

sethood considerations

The functions under consideration are not sets.

```
In[4]:= member[composite[IMAGE[cross[inverse[DUP], x]], CROSS], V] // AssertTest
Out[4]= member[composite[IMAGE[cross[inverse[DUP], x]], CROSS], V] == False

In[5]:= (% /. x → x_) /. Equal → SetDelayed
```

More generally:

```
In[6]:= member[composite[IMAGE[cross[inverse[DUP], x]], CROSS], y] // AssertTest
Out[6]= member[composite[IMAGE[cross[inverse[DUP], x]], CROSS], y] == False

In[7]:= member[composite[IMAGE[cross[inverse[DUP], x_]], CROSS], y_] := False
```

Comment. If one considers only those functions whose domain is some specified set, then the binary operation of pointwise addition for such functions will be a suitable restriction of these global functions. These restrictions will be sets, but in this notebook, only the global functions will be considered.

available theorems

The derivation of the main theorem is based on two theorems about associative relations that are currently available. The first of these theorems says that the direct product of associative relations is associative:

```
In[8]:= implies[and[associative[x], associative[y]],
          associative[composite[cross[x, y], TWIST]]]
Out[8]= True
```

The second theorem says that if **x** is an associative relation, then the function **composite[IMAGE[x], CART]** is associative.

```
In[9]:= implies[and[thin[x], associative[x]], associative[composite[IMAGE[x], CART]]]
Out[9]= True
```

The thinness hypothesis is of little concern in practice because it is satisfied for functions, which is typically the case in applications. Nonetheless, the condition can not be omitted as the following counterexample shows:

```
In[10]:= (implies[associative[x], associative[composite[IMAGE[x], CART]]] /.
            x → cart[cart[V, V], V]) // assert
Out[10]= False
```

the main theorem

The theorem about direct products is applied to the case when one factor is the associative function **inverse[DUP]**.

```
In[11]:= SubstTest[implies, and[associative[w], associative[x]],
          associative[direct[w, x]], w → inverse[DUP]]

Out[11]= or[associative[composite[cross[inverse[DUP], x], TWIST]], not[associative[x]]] == True

In[12]:= (% /. x → x_) /. Equal → SetDelayed
```

The second theorem

```
In[13]:= Map[implies[thin[x], #] &, SubstTest[implies,
            and[thin[w], associative[w]], associative[composite[IMAGE[w], CART]],
            w → composite[cross[inverse[DUP], x], TWIST]]] // MapNotNot

Out[13]= or[associative[composite[IMAGE[cross[inverse[DUP], x]], CROSS]],
           not[associative[composite[cross[inverse[DUP], x], TWIST]]],
           not[equal[V, domain[VERTSECT[x]]]]] == True

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

The main theorem is derived by combining these two results:

```
In[15]:= Map[not, SubstTest[and, implies[p1, p3],
           implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
           {p1 → associative[x], p2 → thin[x],
            p3 -> associative[composite[cross[inverse[DUP], x], TWIST]],
            p4 -> associative[composite[IMAGE[cross[inverse[DUP], x]], CROSS]]}]]

Out[15]= or[associative[composite[IMAGE[cross[inverse[DUP], x]], CROSS]],
           not[associative[x]], not[equal[V, domain[VERTSECT[x]]]]] == True

In[16]:= or[associative[composite[IMAGE[cross[inverse[DUP], x_]], CROSS]],
           not[associative[x_]], not[equal[V, domain[VERTSECT[x_]]]]] := True
```

Examples:

```
In[17]:= SubstTest[implies, and[thin[x], associative[x]],
          associative[composite[IMAGE[cross[inverse[DUP], x]], CROSS]], x → NATADD]

Out[17]= associative[composite[IMAGE[cross[inverse[DUP], NATADD]], CROSS]] == True

In[18]:= associative[composite[IMAGE[cross[inverse[DUP], NATADD]], CROSS]] := True

In[19]:= SubstTest[implies, and[thin[x], associative[x]],
          associative[composite[IMAGE[cross[inverse[DUP], x]], CROSS]], x → INTADD]

Out[19]= associative[composite[IMAGE[cross[inverse[DUP], INTADD]], CROSS]] == True
```

```
In[20]:= associative[composite[IMAGE[cross[inverse[DUP], INTADD]], CROSS]] := True
```

The thinness hypothesis cannot be omitted. The same counterexample as before shows this:

```
In[21]:= (implies[associative[x],
             associative[composite[IMAGE[cross[inverse[DUP], x]], CROSS]]] /.
           x → cart[cart[V, V], V]) // assert

Out[21]= False
```

sethood of binhom[x,y]

The class of endomorphisms of any binary operation is a set.

```
In[22]:= SubstTest[implies, and[subclass[u, v], member[v, V]], member[u, V],
          {u -> binhom[x, x], v → map[fix[domain[x]], fix[domain[x]]]}]

Out[22]= member[binhom[x, x], V] == True

In[23]:= member[binhom[x_, x_], V] := True
```

For binary homomorphisms in general:

```
In[24]:= SubstTest[implies, and[subclass[u, v], member[v, V]], member[u, V],
          {u -> binhom[x, setpart[y]], v → map[fix[domain[x]], fix[domain[setpart[y]]]]}]

Out[24]= member[binhom[x, setpart[y]], V] == True

In[25]:= member[binhom[x_, setpart[y_]], V] := True
```

Corollary.

```
In[26]:= Map[implies[member[y, z], #] &,
          SubstTest[implies, equal[y, setpart[w]], member[binhom[x, y], V], w → y]]

Out[26]= or[member[binhom[x, y], V], not[member[y, z]]] == True

In[27]:= or[member[binhom[x_, y_], V], not[member[y_, z_]]] := True
```

endomorphisms of NATADD

The pointwise sum of endomorphisms of **NATADD** is an endomorphism of **NATADD**. A variable-free expression of this is:

```
In[28]:= Map[equal[0, composite[Id, complement[#]]] &,
          complement[dif[cart[binhom[NATADD, NATADD], binhom[NATADD, NATADD]],
             image[inverse[composite[IMAGE[cross[inverse[DUP], NATADD]], CROSS]],
              binhom[NATADD, NATADD]]]] // RelnNormality]

Out[28]= subclass[image[IMAGE[cross[inverse[DUP], NATADD]],
            image[CROSS, cart[binhom[NATADD, NATADD], binhom[NATADD, NATADD]]]],
           binhom[NATADD, NATADD]] == True

In[29]:= subclass[image[IMAGE[cross[inverse[DUP], NATADD]],
            image[CROSS, cart[binhom[NATADD, NATADD], binhom[NATADD, NATADD]]]],
           binhom[NATADD, NATADD]] := True
```

Corollary.

```
In[30]:= member[binhom[NATADD, NATADD],
          binclosed[composite[IMAGE[cross[inverse[DUP], NATADD]], CROSS]]]

Out[30]= True
```

endomorphisms of INTADD

The pointwise sum of endomorphisms of **INTADD** is an endomorphism of **INTADD**. A variable-free expression of this is:

```
In[31]:= Map[equal[0, composite[Id, complement[#]]] &,
          complement[dif[cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]],
             image[inverse[composite[IMAGE[cross[inverse[DUP], INTADD]], CROSS]],
              binhom[INTADD, INTADD]]]] // RelnNormality]

Out[31]= subclass[image[IMAGE[cross[inverse[DUP], INTADD]],
            image[CROSS, cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]],
           binhom[INTADD, INTADD]] == True

In[32]:= subclass[image[IMAGE[cross[inverse[DUP], INTADD]],
            image[CROSS, cart[binhom[INTADD, INTADD], binhom[INTADD, INTADD]]]],
           binhom[INTADD, INTADD]] := True
```

Corollary.

```
In[33]:= member[binhom[INTADD, INTADD],
          binclosed[composite[IMAGE[cross[inverse[DUP], INTADD]], CROSS]]]

Out[33]= True
```