binary operations on power sets

Johan G. F. Belinfante
2006 September 3

```
In[1]:= SetDirectory["l:"]; << goedel85.02a; << tools.m

        :Package Title: goedel85.02a              2006 September 2 at 11:00 p.m.

        It is now:  2006 Sep 3 at 18:23

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 August 22

        weightlimit = 40
```

summary

For any binary operation **t** on a set **w**, a restriction of the function **composite[IMAGE[t], CART]** is a binary operation on the set of subsets of **w**. If the original binary operation is associative, then so is the corresponding operation on sets.

binary operations on sets

If **t** is a binary operation on **w**, then **t** belongs to **map[cart[w,w], w]**. If **u** and **v** are subsets of **w**, then **image[t, cart[u, v]]** is the set of all elements obtained by applying **t** to pairs of elements, one from **u** and one from **v**.

```
In[2]:= class[z, exists[x, y, and[member[x, u], member[y, v], member[PAIR[PAIR[x, y], z], t]]]]

Out[2]= image[t, cart[u, v]]
```

It will be shown that for any binary operation **t** there is a corresponding binary operation on the subsets of **w** which is a restriction of the function **composite[IMAGE[t], CART]**.

a sethood rule

Lemma.

```
In[3]:= equiv[or[member[x, V], subclass[x, set[0]]], member[x, V]]

Out[3]= True

In[4]:= or[member[x_, V], subclass[x_, set[0]]] := member[x, V]
```

Sethood theorem.

```
In[5]:= SubstTest[member, U[w], V, w -> image[CART, cart[x, x]]] // Reverse

Out[5]= member[image[CART, cart[x, x]], V] == member[x, V]

In[6]:= member[image[CART, cart[x_, x_]], V] := member[x, V]
```

Corollary.

```
In[7]:= Map[implies[member[y, v], #] &, SubstTest[implies, and[member[w, V], member[x, u]],
          member[image[IMAGE[x], w], V], w -> image[CART, cart[P[y], P[y]]]]]

Out[7]= or[member[image[IMAGE[x], image[CART, cart[P[y], P[y]]]], V],
          not[member[x, u]], not[member[y, v]]] == True

In[8]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

---

main theorem

Lemma.

```
In[9]:= SubstTest[implies, member[t, z], not[empty[z]], z → map[cart[w, w], w]]

Out[9]= or[member[w, V], not[member[t, map[cart[w, w], w]]]] == True

In[10]:= or[member[w_, V], not[member[t_, map[cart[w_, w_], w_]]]] := True
```

Lemma.

```
In[11]:= Map[implies[member[t, v], #] &, SubstTest[implies,
           and[equal[s, composite[IMAGE[t], CART, id[cart[P[w], P[w]]]]], equal[t, setpart[r]]],
           equal[domain[s], cart[P[w], P[w]]], r → t]]

Out[11]= or[equal[cart[P[w], P[w]], domain[s]],
           not[equal[s, composite[IMAGE[t], CART, id[cart[P[w], P[w]]]]]],
           not[member[t, v]]] == True

In[12]:= (% /. {s → s_, t → t_, v → v_, w → w_}) /. Equal → SetDelayed
```

Lemma.

```
In[13]:= Map[implies[member[t, v], #] &, SubstTest[implies,
           and[equal[s, composite[IMAGE[t], CART, id[cart[P[w], P[w]]]]], equal[t, setpart[r]]],
           equal[U[range[s]], image[t, cart[w, w]]], r → t]]

Out[13]= or[equal[image[t, cart[w, w]], U[range[s]]],
           not[equal[s, composite[IMAGE[t], CART, id[cart[P[w], P[w]]]]]],
           not[member[t, v]]] == True

In[14]:= (% /. {s → s_, t → t_, v → v_, w → w_}) /. Equal → SetDelayed
```

Lemma.

```
In[15]:= SubstTest[implies, and[member[s, t], FUNCTION[s],
           equal[domain[s], u], subclass[range[s], v]], member[s, map[u, v]], v → P[w]]

Out[15]= or[member[s, map[u, P[w]]], not[equal[u, domain[s]]],
           not[FUNCTION[s]], not[member[s, t]], not[subclass[U[range[s]], w]]] == True

In[16]:= (% /. {s → s_, t → t_, u → u_, w → w_}) /. Equal → SetDelayed
```

Theorem. If **t** is a binary operation on **w,** then a restriction of the function **composite[IMAGE[t], CART]** is a binary operation on the power set **P[w]**. Comments. This derivation takes a few seconds. One step of the proof has been deliberately omitted: **implies[and[p1, p3, p4, p5, p7], p8]**, relying on the rewrite rules of the **GOEDEL** program to supply this missing step. Had this step not been omitted, the derivation could not be completed within three minutes.

```
In[17]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[and[p0, p1, p2], p4],
            implies[p2, p0], implies[and[p1, p2], p5], implies[and[p1, p2], p6],
            implies[and[p2, p6], p7], not[implies[and[p1, p2], p8]],
            {p0 → member[w, V], p1 → equal[s, composite[IMAGE[t], CART, id[cart[P[w], P[w]]]]],
             p2 -> member[t, map[cart[w, w], w]], p3 → FUNCTION[s], p4 -> member[s, V], p5 →
              equal[domain[s], cart[P[w], P[w]]], p6 → equal[U[range[s]], image[t, cart[w, w]]],
             p7 → subclass[U[range[s]], w], p8 -> member[s, map[cart[P[w], P[w]], P[w]]]}]] /.
          s -> composite[IMAGE[t], CART, id[cart[P[w], P[w]]]]

Out[17]= or[member[composite[IMAGE[t], CART, id[cart[P[w], P[w]]]],
           map[cart[P[w], P[w]], P[w]]], not[member[t, map[cart[w, w], w]]]] == True

In[18]:= or[member[composite[IMAGE[t_], CART, id[cart[P[w_], P[w_]]]],
           map[cart[P[w_], P[w_]], P[w_]]], not[member[t_, map[cart[w_, w_], w_]]]] := True
```

The variable **w** can be eliminated:

```
In[19]:= SubstTest[or,
          member[composite[IMAGE[t], CART, id[cart[P[w], P[w]]]], map[cart[P[w], P[w]], P[w]]],
          not[member[t, map[cart[w, w], w]]], w → fix[domain[t]]]

Out[19]= or[member[composite[IMAGE[t], CART, id[cart[P[fix[domain[t]]], P[fix[domain[t]]]]]],
           map[cart[P[fix[domain[t]]], P[fix[domain[t]]]], P[fix[domain[t]]]],
          not[member[t, BINOPS]]] == True

In[20]:= (% /. t → t_) /. Equal → SetDelayed
```

Corollary.

```
In[21]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 → member[x, BINOPS], p2 ->
             member[composite[IMAGE[x], CART, id[cart[P[fix[domain[x]]], P[fix[domain[x]]]]]],
              map[cart[P[fix[domain[x]]], P[fix[domain[x]]]], P[fix[domain[x]]]],
            p3 -> member[composite[IMAGE[x], CART,
               id[cart[P[fix[domain[x]]], P[fix[domain[x]]]]], BINOPS]}]]

Out[21]= or[member[composite[IMAGE[x], CART, id[cart[P[fix[domain[x]]], P[fix[domain[x]]]]],
          BINOPS], not[member[x, BINOPS]]] == True

In[22]:= or[
           member[composite[IMAGE[x_], CART, id[cart[P[fix[domain[x_]]], P[fix[domain[x_]]]]],
            BINOPS], not[member[x_, BINOPS]]] := True
```

commutativity

If a binary operation is commutative, then so is the corresponding operation on sets.

```
In[23]:= SubstTest[implies, equal[x, y],
          equal[composite[IMAGE[x], CART], composite[IMAGE[y], CART]], y → flip[x]]

Out[23]= or[equal[composite[IMAGE[x], CART], composite[IMAGE[x], CART, SWAP]],
           not[equal[x, composite[x, SWAP]]]] == True

In[24]:= or[equal[composite[IMAGE[x_], CART], composite[IMAGE[x_], CART, SWAP]],
           not[equal[x_, composite[x_, SWAP]]]] := True
```

associativity

Lemma.

```
In[25]:= Map[or[#, subclass[image[x, cart[y, y]], y]] &, SubstTest[implies, member[w, BINOPS],
           subclass[range[w], fix[domain[w]]], w -> composite[x, id[cart[y, y]]]]]

Out[25]= or[not[member[composite[x, id[cart[y, y]]], BINOPS]],
           subclass[image[x, cart[y, y]], y]] == True

In[26]:= or[not[member[composite[x_, id[cart[y_, y_]]], BINOPS]],
           subclass[image[x_, cart[y_, y_]], y_]] := True
```

Theorem.

```
In[27]:= Map[not, SubstTest[and, implies[p2, p3], implies[and[p1, p3], p4],
           implies[and[p2, p4], p5], not[implies[and[p1, p2], p5]],
           {p1 → associative[x], p2 → member[composite[x, id[cart[y, y]]], BINOPS],
            p3 -> subclass[image[x, cart[y, y]], y],
            p4 → associative[composite[x, id[cart[y, y]]]],
            p5 -> member[composite[x, id[cart[y, y]]], SEMIGPS]}]]

Out[27]= or[member[composite[x, id[cart[y, y]]], SEMIGPS],
           not[associative[x]], not[member[composite[x, id[cart[y, y]]], BINOPS]]] == True

In[28]:= or[member[composite[x_, id[cart[y_, y_]]], SEMIGPS], not[associative[x_]],
           not[member[composite[x_, id[cart[y_, y_]]], BINOPS]]] := True
```

Corollary. If a binary operation is associative, then so is the corresponding binary operation on sets.

```
In[29]:= Map[not, SubstTest[and, implies[and[p3, p4], p6], implies[p2, p5],
           not[implies[p1, p7]], {p1 → member[x, SEMIGPS], p2 → member[x, BINOPS],
            p3 → associative[x], p4 → equal[V, domain[VERTSECT[x]]], p5 → member[composite[
               IMAGE[x], CART, id[cart[P[fix[domain[x]]], P[fix[domain[x]]]]]], BINOPS],
            p6 → associative[composite[IMAGE[x], CART]], p7 → member[composite[IMAGE[x],
               CART, id[cart[P[fix[domain[x]]], P[fix[domain[x]]]]]], SEMIGPS]}]]

Out[29]= or[member[composite[IMAGE[x], CART, id[cart[P[fix[domain[x]]], P[fix[domain[x]]]]],
            SEMIGPS], not[member[x, SEMIGPS]]] == True

In[30]:= or[
           member[composite[IMAGE[x_], CART, id[cart[P[fix[domain[x_]]], P[fix[domain[x_]]]]],
            SEMIGPS], not[member[x_, SEMIGPS]]] := True
```

Comment. This derivation is remarkable in that many steps have been left out. When all steps of the proof are provided, the derivation takes 7.3 seconds. By leaving out most of the steps, the execution time has been reduced to 0.3 seconds, an overall speedup by a factor of 26. The steps of the original proof that have been left out in the above derivation are the following: **implies[p1, p2]**, **implies[p1,p3], implies[p2,p4]**, **implies[and[p5,p6],p7]**.

---

example: addition of sets of numbers

Corresponding to the binary operation **NATADD** for addition of natural numbers, there is a binary operation on sets of natural numbers:

```
In[31]:= SubstTest[implies, member[x, BINOPS], member[composite[IMAGE[x], CART,
            id[cart[P[fix[domain[x]]], P[fix[domain[x]]]]], BINOPS], x → NATADD]

Out[31]= member[composite[IMAGE[NATADD], CART, id[cart[P[omega], P[omega]]]], BINOPS] == True

In[32]:= member[composite[IMAGE[NATADD], CART, id[cart[P[omega], P[omega]]]], BINOPS] := True
```

```
In[33]:= SubstTest[implies, member[x, SEMIGPS],
          member[composite[IMAGE[x], CART, id[cart[P[fix[domain[x]]], P[fix[domain[x]]]]],
           SEMIGPS], x → NATADD]

Out[33]= member[composite[IMAGE[NATADD], CART, id[cart[P[omega], P[omega]]]], SEMIGPS] == True

In[34]:= member[composite[IMAGE[NATADD], CART, id[cart[P[omega], P[omega]]]], SEMIGPS] := True
```

example: vector addition of sets of numbers

The points **PAIR[x,y]** of the plane **cart[omega,omega]** can be thought of as vectors. The binary operation of vector addition is the direct product of **NATADD** with itself. Note that **TWIST** is needed to interchange **x2** and **y1** so that the first component of the vector sum is the sum of the first components of the individual vectors, and similarly for the second components.

```
In[35]:= APPLY[composite[cross[NATADD, NATADD], TWIST], PAIR[PAIR[x1, x2], PAIR[y1, y2]]]

Out[35]= PAIR[natadd[x1, y1], natadd[x2, y2]]
```

More generally, if **s** and **t** are binary operations, then so is their direct product, **composite[cross[s, t], TWIST]**. Specializing to the case **s = t = NATADD**, one finds

```
In[36]:= SubstTest[implies, and[member[s, BINOPS], member[t, BINOPS]],
          member[composite[cross[s, t], TWIST], BINOPS], {s -> NATADD, t -> NATADD}]

Out[36]= member[composite[cross[NATADD, NATADD], TWIST], BINOPS] == True

In[37]:= member[composite[cross[NATADD, NATADD], TWIST], BINOPS] := True
```

Vector addition is associative.

```
In[38]:= SubstTest[implies, and[member[s, SEMIGPS], member[t, SEMIGPS]],
          member[composite[cross[s, t], TWIST], SEMIGPS], {s -> NATADD, t -> NATADD}]

Out[38]= member[composite[cross[NATADD, NATADD], TWIST], SEMIGPS] == True

In[39]:= member[composite[cross[NATADD, NATADD], TWIST], SEMIGPS] := True
```

There is a corresponding binary operation on subsets of the number plane **cart[omega,omega]**:

```
In[40]:= SubstTest[implies, member[x, BINOPS],
          member[composite[IMAGE[x], CART, id[cart[P[fix[domain[x]]], P[fix[domain[x]]]]],
           BINOPS], x → composite[cross[NATADD, NATADD], TWIST]]

Out[40]= member[composite[IMAGE[cross[NATADD, NATADD]], CROSS,
            id[cart[P[cart[omega, omega]], P[cart[omega, omega]]]]], BINOPS] == True

In[41]:= member[composite[IMAGE[cross[NATADD, NATADD]], CROSS,
            id[cart[P[cart[omega, omega]], P[cart[omega, omega]]]]], BINOPS] := True
```

Vector addition of sets is also associative.

```
In[42]:= SubstTest[implies, member[x, SEMIGPS],
          member[composite[IMAGE[x], CART, id[cart[P[fix[domain[x]]], P[fix[domain[x]]]]],
           SEMIGPS], x → composite[cross[NATADD, NATADD], TWIST]]

Out[42]= member[composite[IMAGE[cross[NATADD, NATADD]], CROSS,
            id[cart[P[cart[omega, omega]], P[cart[omega, omega]]]]], SEMIGPS] == True

In[43]:= member[composite[IMAGE[cross[NATADD, NATADD]], CROSS,
            id[cart[P[cart[omega, omega]], P[cart[omega, omega]]]]], SEMIGPS] := True
```

Explicitly, if **x** and **y** are subsets of **cart[omega, omega]**, then their vector sum is the subset **composite[NATADD, cross[x,y], inverse[NATADD]]**.

```
In[44]:= APPLY[composite[IMAGE[cross[NATADD, NATADD]], CROSS], PAIR[setpart[x], setpart[y]]]

Out[44]= composite[NATADD, cross[setpart[x], setpart[y]], inverse[NATADD]]
```