idempotents and binary homomorphisms

Johan G. F. Belinfante
2006 November 27

```
In[1]:= SetDirectory["l:"]; << goedel87.25b; << tools.m

        :Package Title: goedel87.25b          2006 November 25 at 8:55 p.m.

        It is now:  2006 Nov 27 at 16:30

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 November 22

        weightlimit = 40
```

summary

This notebook is concerned with idempotents for binary operations. The class of all idempotents for **x** is:

```
In[2]:= class[w, member[pair[pair[w, w], w], x]]

Out[2]= fix[composite[x, DUP]]
```

Binary homomorphisms preserve idempotents. If a binary operation **y** has idempotents, then there are constant binary homomorphism from any binary operation **x** to **y**. An example is the zero homomorphism from **NATADD** to **INTADD**:

```
In[3]:= member[cart[omega, set[id[omega]]], binhom[NATADD, INTADD]] // AssertTest

Out[3]= member[cart[omega, set[id[omega]]], binhom[NATADD, INTADD]] == True

In[4]:= member[cart[omega, set[id[omega]]], binhom[NATADD, INTADD]] := True
```

The existence of binary homomorphisms allows one to compute domains of generalized dividibility relations.

binary homomorphisms preserve idempotents

Lemma.

```
In[5]:= SubstTest[implies,
          and[equal[w, funpart[t]], equal[composite[w, x], composite[y, cross[w, w]]]], subclass[
           fix[composite[w, x, DUP, inverse[w]]], fix[composite[y, DUP]]], t → w] // Reverse

Out[5]= or[not[equal[composite[w, x], composite[y, cross[w, w]]]], not[FUNCTION[w]],
          subclass[fix[composite[w, x, DUP, inverse[w]]], fix[composite[y, DUP]]]] == True
```

```
In[6]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[7]:= Map[not, SubstTest[and, implies[and[p2, p3], p4],
           not[implies[p1, p5]], {p1 → member[w, binhom[x, y]], p2 → FUNCTION[w],
            p3 -> equal[composite[w, x], composite[y, cross[w, w]]],
            p4 -> subclass[fix[composite[w, x, DUP, inverse[w]]], fix[composite[y, DUP]]],
            p5 → subclass[image[w, fix[composite[x, DUP]]], fix[composite[y, DUP]]]}]] // Reverse

Out[7]= or[not[member[w, binhom[x, y]]],
          subclass[image[w, fix[composite[x, DUP]]], fix[composite[y, DUP]]]] == True

In[8]:= or[not[member[w_, binhom[x_, y_]]],
          subclass[image[w_, fix[composite[x_, DUP]]], fix[composite[y_, DUP]]]] := True
```

Eliminating the variable **w** yields this corollary about the generalized divisibility relation **U[binhom[x,y]]**.

```
In[9]:= Map[equal[V, #] &, SubstTest[class, t, or[not[member[t, u]], subclass[image[t, v], w]],
          {u → binhom[x, y], v -> fix[composite[x, DUP]], w -> fix[composite[y, DUP]]}]]

Out[9]= subclass[image[U[binhom[x, y]], fix[composite[x, DUP]]], fix[composite[y, DUP]]] == True

In[10]:= subclass[image[U[binhom[x_, y_]], fix[composite[x_, DUP]]],
           fix[composite[y_, DUP]]] := True
```

---

an example: integer divisibility

From the general theory presented above, one obtains:

```
In[11]:= SubstTest[subclass,
           image[U[binhom[x, y]], fix[composite[x, DUP]]], fix[composite[y, DUP]],
           {x → INTADD, y → INTADD}] // Reverse

Out[11]= subclass[image[INTDIV, set[id[omega]]], set[id[omega]]] == True

In[12]:= % /. Equal → SetDelayed
```

The inclusion in the other direction amounts to this:

```
In[13]:= member[pair[id[omega], id[omega]], INTDIV] // AssertTest

Out[13]= member[pair[id[omega], id[omega]], INTDIV] == True

In[14]:= member[pair[id[omega], id[omega]], INTDIV] := True
```

These two inclusions can be combined to obtain an equation, which is made into a rewrite rule.

```
In[15]:= equal[image[INTDIV, set[id[omega]]], set[id[omega]]]

Out[15]= True
```

```
In[16]:= image[INTDIV, set[id[omega]]] := set[id[omega]]
```

This just says that the only integer which is divisible by zero is zero. Recall that the integer zero is **id[omega]**.

constant homomorphisms

Lemma.

```
In[17]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], not[implies[p1, p4]],
           {p1 → member[x, BINOPS], p2 → equal[domain[x], cartsq[fix[domain[x]]]],
            p3 → subclass[range[x], fix[domain[x]]], p4 -> equal[cart[fix[domain[x]],
              fix[domain[x]]], image[inverse[x], fix[domain[x]]]]}]] // Reverse
```

```
Out[17]= or[equal[cart[fix[domain[x]], fix[domain[x]]], image[inverse[x], fix[domain[x]]]],
           not[member[x, BINOPS]]] == True
```

```
In[18]:= or[equal[cart[fix[domain[x_]], fix[domain[x_]]], image[inverse[x_], fix[domain[x_]]]],
           not[member[x_, BINOPS]]] := True
```

The following lemma is needed to circumvent technical limitations on equality substitution in the **GOEDEL** program.

```
In[19]:= SubstTest[or, member[u, domain[funpart[y]]],
           not[equal[w, APPLY[funpart[y], u]]], not[member[w, V]], u → pair[w, w]] // Reverse
```

```
Out[19]= or[member[pair[w, w], domain[funpart[y]]],
           not[equal[w, APPLY[funpart[y], pair[w, w]]]], not[member[w, V]]] == True
```

```
In[20]:= (% /. {w → w_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[21]:= (Map[implies[and[equal[set[v], image[y, cart[set[v], set[v]]]],
             member[v, fix[domain[y]]], member[x, BINOPS]], #] &,
           (member[w, binhom[x, y]] // AssertTest) /.
            {w → cart[fix[domain[x]], set[v]]}]) // MapNotNot
```

```
Out[21]= or[member[cart[fix[domain[x]], set[v]], binhom[x, y]],
           not[equal[image[y, cart[set[v], set[v]]], set[v]]],
           not[member[v, fix[domain[y]]]], not[member[x, BINOPS]]] == True
```

```
In[22]:= (% /. {v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[23]:= (or[equal[image[u, cart[set[v], set[v]]], set[v]],
             not[member[v, fix[composite[u, DUP]]]]] // AssertTest) /. u → funpart[y]
```

```
Out[23]= or[equal[set[v], set[APPLY[funpart[y], PAIR[v, v]]]],
           not[member[v, fix[composite[funpart[y], DUP]]]]] == True
```

```
In[24]:= (% /. {v → v_, y → y_}) /. Equal → SetDelayed
```

Removing the **funpart** wrapper yields:

```
In[25]:= SubstTest[implies, and[equal[y, funpart[t]], member[v, fix[composite[y, DUP]]]],
           equal[set[v], image[y, cart[set[v], set[v]]]], t → y] // Reverse

Out[25]= or[equal[image[y, cart[set[v], set[v]]], set[v]],
           not[FUNCTION[y]], not[member[v, fix[composite[y, DUP]]]]] == True

In[26]:= (% /. {v → v_, y → y_}) /. Equal → SetDelayed
```

Theorem. Idempotents yield constant homomorphisms.

```
In[27]:= Map[not, SubstTest[and, implies[and[p3, p4], p5], implies[and[p6, p7], p8],
            not[implies[and[p1, p2, p3], p9]], {p1 → member[x, BINOPS],
             p2 → member[y, BINOPS], p3 -> member[v, fix[composite[y, DUP]]], p4 → FUNCTION[y],
             p5 -> equal[image[y, cart[set[v], set[v]]], set[v]], p6 → member[v, range[y]],
             p7 → subclass[range[y], fix[domain[y]]], p8 → member[v, fix[domain[y]]],
             p9 -> member[cart[fix[domain[x]], set[v]], binhom[x, y]]}]] // Reverse

Out[27]= or[member[cart[fix[domain[x]], set[v]], binhom[x, y]],
           not[member[v, fix[composite[y, DUP]]]],
           not[member[x, BINOPS]], not[member[y, BINOPS]]] == True

In[28]:= or[member[cart[fix[domain[x_]], set[v_]], binhom[x_, y_]],
           not[member[v_, fix[composite[y_, DUP]]]],
           not[member[x_, BINOPS]], not[member[y_, BINOPS]]] := True
```

some corollaries

Lemma.

```
In[29]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
            {p1 → and[member[x, BINOPS], member[y, BINOPS], equal[0, binhom[x, y]]],
             p2 → not[member[cart[fix[domain[x]], set[v]], binhom[x, y]]],
             p3 → not[member[v, fix[composite[y, DUP]]]]}]] // Reverse

Out[29]= or[not[equal[0, binhom[x, y]]], not[member[v, fix[composite[y, DUP]]]],
           not[member[x, BINOPS]], not[member[y, BINOPS]]] == True

In[30]:= (% /. {v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Corollary. The existence of idempotents implies the existence of binary homomorphisms.

```
In[31]:= Map[equal[V, #] &, SubstTest[class, v,
           or[not[equal[0, t]], not[member[v, w]], not[member[x, z]], not[member[y, z]]],
           {t → binhom[x, y], w -> fix[composite[y, DUP]], z → BINOPS}]]

Out[31]= or[equal[0, fix[composite[y, DUP]]], not[equal[0, binhom[x, y]]],
           not[member[x, BINOPS]], not[member[y, BINOPS]]] == True
```

```
In[32]:= or[equal[0, fix[composite[y_, DUP]]], not[equal[0, binhom[x_, y_]]],
           not[member[x_, BINOPS]], not[member[y_, BINOPS]]] := True
```

Corollary. If **x** and **y** are binary operations, and if **y** has idempotents, then the domain of the generalized divisibility relation **U[binhom[x,y]]** is equal to **fix[domain[x]]**.

```
In[33]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]],
            {p1 → and[member[x, BINOPS], member[y, BINOPS], not[empty[fix[composite[y, DUP]]]]],
             p2 → not[empty[binhom[x, y]]],
             p3 → equal[domain[U[binhom[x, y]]], fix[domain[x]]]}]] // Reverse

Out[33]= or[equal[0, fix[composite[y, DUP]]], equal[domain[U[binhom[x, y]]], fix[domain[x]]],
           not[member[x, BINOPS]], not[member[y, BINOPS]]] == True

In[34]:= or[equal[0, fix[composite[y_, DUP]]],
           equal[domain[U[binhom[x_, y_]]], fix[domain[x_]]],
           not[member[x_, BINOPS]], not[member[y_, BINOPS]]] := True
```