INTMUL, part 5. intmul[x, y]

Johan G. F. Belinfante
2007 January 5

```
In[1]:= SetDirectory["l:"]; << goedel89.05a; << tools.m

        :Package Title: goedel89.05a        2007 January 5 at 2:15 p.m.

        It is now:  2007 Jan 5 at 17:6

        Loading Simplification Rules

        TOOLS.M                       Revised 2007 January 5

        weightlimit = 40
```

summary

The following membership rule for **intmul** has been added to the **GOEDEL** program.

```
In[2]:= Begin["Goedel`Private`"];

In[3]:= FirstMatch[class[u_, member[v_, HoldPattern[intmul[x_, y_]]]]]

Out[3]= class[u_, member[v_, intmul[x_, y_]]] := Module[{w = Unique[]}, class[u, and[member[
          v, V], forall[w, implies[member[pair[pair[x, y], w], INTMUL], member[v, w]]]]]]
```

Normalization results for **intmul** are derived here, and justification for its attributes.

normalization

Lemma.

```
In[4]:= ImageComp[id[binhom[INTADD, INTADD]],
          inverse[eval[composite[id[omega], SUCC]]], x] // Reverse

Out[4]= intersection[binhom[INTADD, INTADD],
          image[inverse[eval[composite[id[omega], SUCC]]], x]] == image[INTTIMES, x]

In[5]:= intersection[binhom[INTADD, INTADD],
          image[inverse[eval[composite[id[omega], SUCC]]], x_]] := image[INTTIMES, x]

In[6]:= intmul[x, y] // Normality // Reverse

Out[6]= APPLY[INTMUL, PAIR[x, y]] == intmul[x, y]

In[7]:= APPLY[INTMUL, PAIR[x_, y_]] := intmul[x, y]
```

```
In[8]:= image[INTMUL, cart[set[x_], set[y_]]] =.

In[9]:= SubstTest[image, funpart[w], cart[set[x], set[y]], w → INTMUL] // Reverse

Out[9]= image[INTMUL, cart[set[x], set[y]]] == set[intmul[x, y]]

In[10]:= image[INTMUL, cart[set[x_], set[y_]]] := set[intmul[x, y]]
```

---

simplification rules

```
In[11]:= SubstTest[equal, V, APPLY[funpart[w], PAIR[x, y]], w → INTMUL] // Reverse

Out[11]= equal[V, intmul[x, y]] == or[not[member[x, Z]], not[member[y, Z]]]

In[12]:= equal[V, intmul[x_, y_]] := or[not[member[x, Z]], not[member[y, Z]]]

In[13]:= equal[union[complement[image[V, set[x]]], intmul[x, y]], intmul[x, y]]

Out[13]= True

In[14]:= equal[union[complement[image[V, set[x]]], intmul[x, y]], intmul[x, y]]

Out[14]= True

In[15]:= union[complement[image[V, set[x_]]], intmul[x_, y_]] := intmul[x, y]

In[16]:= equal[union[complement[image[V, set[y]]], intmul[x, y]], intmul[x, y]]

Out[16]= True

In[17]:= union[complement[image[V, set[y_]]], intmul[x_, y_]] := intmul[x, y]
```

---

sethood

```
In[18]:= SubstTest[member, APPLY[funpart[w], PAIR[x, y]], V, w → INTMUL] // Reverse

Out[18]= member[intmul[x, y], V] == and[member[x, Z], member[y, Z]]

In[19]:= member[intmul[x_, y_], V] := and[member[x, Z], member[y, Z]]
```

---

commutative law

```
In[20]:= Map[implies[and[member[x, Z], member[y, Z]], member[intmul[x, y], #]] &,
           ImageComp[INTMUL, SWAP, set[PAIR[x, y]]]] // Reverse

Out[20]= or[equal[intmul[x, y], intmul[y, x]], not[member[x, Z]], not[member[y, Z]]] == True

In[21]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

```
In[22]:= SubstTest[implies, and[equal[u, v], equal[v, w]], equal[u, w],
           {u → intmul[x, y], v → V, w → intmul[y, x]}] // Reverse
```

```
Out[22]= or[and[member[x, Z], member[y, Z]], equal[intmul[x, y], intmul[y, x]]] == True
```

```
In[23]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

```
In[24]:= SubstTest[and, implies[p, q], or[p, q],
          {p -> and[member[x, Z], member[y, Z]], q -> equal[intmul[x, y], intmul[y, x]]}]
```

```
Out[24]= equal[intmul[x, y], intmul[y, x]] == True
```

```
In[25]:= equal[intmul[x_, y_], intmul[y_, x_]] := True
```

---

integerhood

```
In[26]:= SubstTest[implies,
           and[FUNCTION[w], member[v, domain[w]]], member[APPLY[w, v], range[w]],
           {v → PAIR[x, y], w -> INTMUL}] // Reverse
```

```
Out[26]= or[member[intmul[x, y], Z], not[member[x, Z]], not[member[y, Z]]] == True
```

```
In[27]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

```
In[28]:= SubstTest[implies, member[t, Z], member[t, V], t -> intmul[x, y]] // Reverse
```

```
Out[28]= or[and[member[x, Z], member[y, Z]], not[member[intmul[x, y], Z]]] == True
```

```
In[29]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

```
In[30]:= equiv[member[intmul[x, y], Z], and[member[x, Z], member[y, Z]]]
```

```
Out[30]= True
```

```
In[31]:= member[intmul[x_, y_], Z] := and[member[x, Z], member[y, Z]]
```

---

associative law

```
In[32]:= Map[implies[and[member[x, Z], member[y, Z], member[z, Z]],
             member[intmul[x, intmul[y, z]], #]] &, ImageComp[
            composite[INTMUL, cross[Id, INTMUL]], ASSOC, set[PAIR[PAIR[x, y], z]]]] // MapNotNot
```

```
Out[32]= or[equal[intmul[x, intmul[y, z]], intmul[intmul[x, y], z]],
           not[member[x, Z]], not[member[y, Z]], not[member[z, Z]]] == True
```

```
In[33]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

```
In[34]:= SubstTest[implies, and[equal[u, v], equal[v, w]], equal[u, w],
           {u → intmul[x, intmul[y, z]], v → V, w → intmul[intmul[x, y], z]}] // Reverse //
         MapNotNot

Out[34]= or[and[member[x, Z], member[y, Z], member[z, Z]],
           equal[intmul[x, intmul[y, z]], intmul[intmul[x, y], z]]] == True

In[35]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed

In[36]:= SubstTest[and, implies[p, q], or[p, q],
         {p -> and[member[x, Z], member[y, Z], member[z, Z]],
          q -> equal[intmul[x, intmul[y, z]], intmul[intmul[x, y], z]]}]

Out[36]= equal[intmul[x, intmul[y, z]], intmul[intmul[x, y], z]] == True

In[37]:= equal[intmul[x_, intmul[y_, z_]], intmul[intmul[x_, y_], z_]] := True
```

multiplying by 1

```
In[38]:= Map[implies[member[x, Z], member[x, #]] &,
           ImageComp[INTMUL, LEFT[plus[set[0]]], set[x]]] // Reverse

Out[38]= or[equal[x, intmul[composite[id[omega], SUCC], x]], not[member[x, Z]]] == True

In[39]:= (% /. x → x_) /. Equal → SetDelayed

In[40]:= equal[intmul[composite[id[omega], SUCC], x],
          union[x, complement[image[V, intersection[Z, set[x]]]]]]

Out[40]= True

In[41]:= intmul[composite[id[omega], SUCC], x_] :=
          union[x, complement[image[V, intersection[Z, set[x]]]]]
```

mult by zero

```
In[42]:= Map[implies[member[x, Z], member[id[omega], #]] &,
           ImageComp[INTMUL, LEFT[plus[0]], set[x]]] // Reverse

Out[42]= or[equal[id[omega], intmul[id[omega], x]], not[member[x, Z]]] == True

In[43]:= (% /. x → x_) /. Equal → SetDelayed

In[44]:= equal[intmul[id[omega], x],
          union[id[omega], complement[image[V, intersection[Z, set[x]]]]]]

Out[44]= True

In[45]:= intmul[id[omega], x_] :=
          union[complement[image[V, intersection[Z, set[x]]]], id[omega]]
```

mult by -1

```
In[46]:= Map[implies[member[x, Z], member[inverse[x], #]] &,
           ImageComp[INTMUL, LEFT[inverse[plus[set[0]]]], set[x]]] // MapNotNot // Reverse

Out[46]= or[equal[intmul[composite[inverse[SUCC], id[omega]], x], inverse[x]],
           not[member[x, Z]]] == True

In[47]:= (% /. x → x_) /. Equal → SetDelayed

In[48]:= equal[intmul[composite[inverse[SUCC], id[omega]], x],
           union[inverse[x], complement[image[V, intersection[Z, set[x]]]]]]

Out[48]= True

In[49]:= intmul[composite[inverse[SUCC], id[omega]], x_] :=
           union[complement[image[V, intersection[Z, set[x]]]], inverse[x]]
```

an example: -1 times -1 is +1.

```
In[50]:= intmul[inverse[plus[set[0]]], inverse[plus[set[0]]]]

Out[50]= composite[id[omega], SUCC]
```

attributes

```
In[51]:= intmul[x_] := x

In[52]:= Attributes[intmul] := {Flat, OneIdentity, Orderless}
```