multiplying even and odd numbers

Johan G. F. Belinfante
2006 August 16

```
In[1]:= SetDirectory["l:"]; << goedel84.16a; << tools.m

        :Package Title: goedel84.16a       2006 August 16 at 6:45 a.m.

        It is now:  2006 Aug 16 at 10:2

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 August 15

        weightlimit = 40
```

summary

The product of two natural numbers is odd if both numbers are odd, and otherwise the product is even.

non-emptiness of the sets even and odd

Neither of the sets **even** and **odd** is empty.

```
In[2]:= Map[not, SubstTest[implies, member[x, y], not[empty[y]], {x → 0, y → even}]]

Out[2]= equal[0, even] == False

In[3]:= equal[0, even] := False

In[4]:= Map[not, SubstTest[implies, member[x, y], not[empty[y]], {x → set[0], y → odd}]]

Out[4]= equal[0, odd] == False

In[5]:= equal[0, odd] := False
```

multiples of even numbers

Any multiple of an even number is even.

```
In[6]:= SubstTest[image, NATMUL, cart[image[DIV, set[x]], omega], x → succ[set[0]]]

Out[6]= image[DIV, even] == even
```

```
In[7]:= image[DIV, even] := even
```

Corollary.

```
In[8]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → set[PAIR[x, y]], v → cart[even, omega], w → NATMUL}]
Out[8]= or[member[natmul[x, y], even], not[member[x, even]], not[member[y, omega]]] == True

In[9]:= or[member[natmul[x_, y_], even], not[member[x_, even]], not[member[y_, omega]]] := True
```

multiples of odd numbers

Lemma. Since **1** is an odd number, one has:

```
In[10]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
           {u → set[set[0]], v → odd, w → DIV}]
Out[10]= subclass[omega, image[DIV, odd]] == True

In[11]:= % /. Equal → SetDelayed
```

Theorem. Every natural number is a multiple of an odd number.

```
In[12]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → omega, v → image[DIV, odd]}]
Out[12]= True == equal[omega, image[DIV, odd]]

In[13]:= image[DIV, odd] := omega
```

multiplying even and odd numbers

The product of two even numbers is even.

```
In[14]:= Map[not, SubstTest[and, implies[p1, p3],
            implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
            {p1 → member[x, even], p2 -> member[y, even],
             p3 → member[x, omega], p4 → member[natmul[x, y], even]}]]
Out[14]= or[member[natmul[x, y], even], not[member[x, even]], not[member[y, even]]] == True

In[15]:= or[member[natmul[x_, y_], even], not[member[x_, even]], not[member[y_, even]]] := True
```

The product of an even and an odd number is even.

```
In[16]:= Map[not, SubstTest[and, implies[p1, p3],
           implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
           {p1 → member[x, odd], p2 -> member[y, even],
            p3 → member[x, omega], p4 → member[natmul[x, y], even]}]]

Out[16]= or[member[natmul[x, y], even], not[member[x, odd]], not[member[y, even]]] == True

In[17]:= or[member[natmul[x_, y_], even], not[member[x_, odd]], not[member[y_, even]]] := True
```

Lemma.

```
In[18]:= Map[implies[and[#, member[x, odd]], member[y, odd]] &,
           SubstTest[member, pair[x, y], image[inverse[NATADD], z], z → even]] // Reverse

Out[18]= or[member[y, odd], not[member[x, odd]], not[member[natadd[x, y], even]]] == True

In[19]:= or[member[y_, odd], not[member[x_, odd]], not[member[natadd[x_, y_], even]]] := True
```

Lemma.

```
In[20]:= SubstTest[or, member[natmul[x, z], even],
          not[member[x, odd]], not[member[z, even]], z → succ[y]]

Out[20]= or[member[natadd[x, natmul[x, y]], even],
           not[member[x, odd]], not[member[y, odd]]] == True

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

Theorem. The product of two odd numbers is odd.

```
In[22]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[and[p1, p3], p4],
           not[implies[and[p1, p2], p4]], {p1 → member[x, odd], p2 → member[y, odd],
            p3 -> member[natadd[x, natmul[x, y]], even], p4 -> member[natmul[x, y], odd]}]]

Out[22]= or[member[natmul[x, y], odd], not[member[x, odd]], not[member[y, odd]]] == True

In[23]:= or[member[natmul[x_, y_], odd], not[member[x_, odd]], not[member[y_, odd]]] := True
```

divisors of even and odd numbers

Every divisor of an odd number is odd.

```
In[24]:= SubstTest[subclass, w, intersection[x, y, z],
          {w -> image[inverse[NATMUL], odd], x -> cart[omega, omega],
           y -> cart[omega, complement[even]], z → cart[complement[even], omega]}]

Out[24]= subclass[image[inverse[DIV], odd], odd] == True

In[25]:= % /. Equal → SetDelayed
```

Lemma.

```
In[26]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
           {v -> P[fix[x]], w -> subvar[x]}] /. {u → odd, x → inverse[DIV]}
Out[26]= subclass[odd, image[inverse[DIV], odd]] == True

In[27]:= % /. Equal → SetDelayed

In[28]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → image[inverse[DIV], odd], v → odd}]
Out[28]= True == equal[odd, image[inverse[DIV], odd]]

In[29]:= image[inverse[DIV], odd] := odd
```

Theorem. Every natural number is a divisor of an even number.

```
In[30]:= ImageComp[inverse[DIV], DIV, even] // Reverse
Out[30]= image[inverse[DIV], even] == omega

In[31]:= image[inverse[DIV], even] := omega
```

inverse image rules for NATMUL

Lemma.

```
In[32]:= Map[equal[0, composite[Id, complement[#]]] &,
           complement[dif[cart[odd, odd], image[inverse[NATMUL], odd]]] // RelnNormality]
Out[32]= subclass[cart[odd, odd], image[inverse[NATMUL], odd]] == True

In[33]:= % /. Equal → SetDelayed
```

Theorem. If a product of two natural numbers is odd, then both factors are odd.

```
In[34]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> image[inverse[NATMUL], odd], v → cart[odd, odd]}]
Out[34]= True == equal[cart[odd, odd], image[inverse[NATMUL], odd]]

In[35]:= image[inverse[NATMUL], odd] := cart[odd, odd]
```

Corollary. If a product is even, at least one factor is even.

```
In[36]:= SubstTest[intersection, image[inverse[NATMUL], x],
           image[inverse[NATMUL], y], {x → omega, y → complement[odd]}] // Reverse
Out[36]= image[inverse[NATMUL], even] == union[cart[even, omega], cart[omega, even]]

In[37]:= image[inverse[NATMUL], even] := union[cart[even, omega], cart[omega, even]]
```

image rules for NATMUL

The following inclusion cannot be sharpened to an equation because, for example, one cannot write **2** as the product of two even numbers.

```
In[38]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
           {u → cart[even, even], v → cart[even, omega], w → NATMUL}]
```

```
Out[38]= subclass[image[NATMUL, cart[even, even]], even] == True
```

```
In[39]:= subclass[image[NATMUL, cart[even, even]], even] := True
```

Lemma.

```
In[40]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
           {u → cart[even, odd], v → cart[even, omega], w → NATMUL}]
```

```
Out[40]= subclass[image[NATMUL, cart[even, odd]], even] == True
```

```
In[41]:= % /. Equal → SetDelayed
```

Lemma.

```
In[42]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
           {u → set[set[0]], v → odd, w → composite[NATMUL, id[cart[even, V]], inverse[SECOND]]}]
```

```
Out[42]= subclass[even, image[NATMUL, cart[even, odd]]] == True
```

```
In[43]:= % /. Equal → SetDelayed
```

Theorem.

```
In[44]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u → image[NATMUL, cart[even, odd]], v -> even}]
```

```
Out[44]= True == equal[even, image[NATMUL, cart[even, odd]]]
```

```
In[45]:= image[NATMUL, cart[even, odd]] := even
```

Corollary.

```
In[46]:= ImageComp[NATMUL, SWAP, cart[odd, even]]
```

```
Out[46]= image[NATMUL, cart[odd, even]] == even
```

```
In[47]:= image[NATMUL, cart[odd, even]] := even
```

Theorem. The set of odd numbers is equal to the set of products of odd numbers.

In[48]:= **ImageComp[NATMUL, inverse[NATMUL], odd] // Reverse**

Out[48]= image[NATMUL, cart[odd, odd]] == odd

In[49]:= **image[NATMUL, cart[odd, odd]] := odd**