adding even and odd numbers

Johan G. F. Belinfante
2006 August 16

In[1]:=
```
SetDirectory["l:"]; << goedel84.15b; << tools.m
```

```
:Package Title: goedel84.15b       2006 August 15 at 7:20 p.m.

It is now:  2006 Aug 16 at 5:52

Loading Simplification Rules

TOOLS.M                       Revised 2006 August 15

weightlimit = 40
```

summary

If the sum of two natural numbers is even, then either both are even or both are odd. If the sum is odd, one of the numbers is even and the other is odd.

lemmas

The following formula for the set of odd numbers amounts to the statement that any odd number can be written as **n + (n + 1)**.

In[2]:=
```
ImageComp[composite[NATADD, cross[Id, NATADD]], ASSOC, cart[Id, set[set[0]]]] // Reverse
```

Out[2]=
```
image[NATADD, composite[id[omega], SUCC]] == odd
```

In[3]:=
```
image[NATADD, composite[id[omega], SUCC]] := odd
```

The following twist rule for **NATADD** is a variable-free statement of the fact that **(a + c) + (b + d) = (a + b) + (c + d)**.

In[4]:=
```
SubstTest[implies, and[associative[x], equal[flip[x], x]],
  equal[composite[x, cross[x, x], TWIST], composite[x, cross[x, x]]], x -> NATADD]
```

Out[4]=
```
equal[composite[NATADD, cross[NATADD, NATADD]],
  composite[NATADD, cross[NATADD, NATADD], TWIST]] == True
```

In[5]:=
```
composite[NATADD, cross[NATADD, NATADD], TWIST] :=
  composite[NATADD, cross[NATADD, NATADD]]
```

A similar result holds for **NATMUL**, but it will not be used in this notebook.

```
In[6]:= SubstTest[implies, and[associative[x], equal[flip[x], x]],
         equal[composite[x, cross[x, x], TWIST], composite[x, cross[x, x]]], x -> NATMUL]

Out[6]= equal[composite[NATMUL, cross[NATMUL, NATMUL]],
          composite[NATMUL, cross[NATMUL, NATMUL], TWIST]] == True

In[8]:= composite[NATMUL, cross[NATMUL, NATMUL], TWIST] :=
         composite[NATMUL, cross[NATMUL, NATMUL]]
```

closure under addition

The following three results follow from the lemmas in the preceding section.

```
In[9]:= ImageComp[composite[NATADD, cross[NATADD, NATADD]], TWIST, cart[Id, Id]]

Out[9]= image[NATADD, cart[even, even]] == even

In[10]:= image[NATADD, cart[even, even]] := even

In[11]:= ImageComp[composite[NATADD, cross[NATADD, NATADD]], TWIST, cart[Id, plus[set[0]]]]

Out[11]= image[NATADD, cart[even, odd]] == odd

In[12]:= image[NATADD, cart[even, odd]] := odd

In[13]:= ImageComp[composite[NATADD, cross[NATADD, NATADD]], TWIST, cart[plus[set[0]], Id]]

Out[13]= image[NATADD, cart[odd, even]] == odd

In[14]:= image[NATADD, cart[odd, even]] := odd
```

Comment. The image of **cart[odd,odd]** under **NATADD** is not equal to **even**. The even number **0** cannot be written as the sum of two odd natural numbers.

adding even and odd numbers

The sum of two even numbers is even.

```
In[15]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → set[PAIR[x, y]], v → cart[even, even], w → NATADD}]

Out[15]= or[member[natadd[x, y], even], not[member[x, even]], not[member[y, even]]] == True

In[16]:= or[member[natadd[x_, y_], even], not[member[x_, even]], not[member[y_, even]]] := True
```

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

```
In[17]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → set[PAIR[x, y]], v → cart[even, odd], w → NATADD}]

Out[17]= or[member[natadd[x, y], odd], not[member[x, even]], not[member[y, odd]]] == True

In[18]:= or[member[natadd[x_, y_], odd], not[member[x_, even]], not[member[y_, odd]]] := True
```

The remaining case is derived by replacing **x** with **succ[x]**.

```
In[19]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → set[PAIR[succ[x], y]], v → cart[even, odd], w → NATADD}]

Out[19]= or[member[natadd[x, y], even], not[member[x, odd]], not[member[y, odd]]] == True

In[20]:= or[member[natadd[x_, y_], even], not[member[x_, odd]], not[member[y_, odd]]] := True
```

inverse image lemmas

Four lemmas are obtained by eliminating variables from the results of the preceding section.

```
In[21]:= Map[equal[0, composite[Id, complement[#]]] &, SubstTest[class,
            pair[x, y], or[member[pair[x, y], v], not[member[x, u]], not[member[y, u]]],
            {u → odd, v → image[inverse[NATADD], even]}]] // Reverse

Out[21]= subclass[cart[odd, odd], image[inverse[NATADD], even]] == True

In[22]:= % /. Equal → SetDelayed

In[23]:= Map[equal[0, composite[Id, complement[#]]] &, SubstTest[class,
            pair[x, y], or[member[pair[x, y], v], not[member[x, u]], not[member[y, u]]],
            {u → even, v → image[inverse[NATADD], even]}]] // Reverse

Out[23]= subclass[cart[even, even], image[inverse[NATADD], even]] == True

In[24]:= % /. Equal → SetDelayed

In[25]:= Map[equal[0, composite[Id, complement[#]]] &, SubstTest[class,
            pair[x, y], or[member[pair[x, y], w], not[member[x, u]], not[member[y, v]]],
            {u → odd, v → even, w → image[inverse[NATADD], odd]}]] // Reverse

Out[25]= subclass[cart[odd, even], image[inverse[NATADD], odd]] == True

In[26]:= % /. Equal → SetDelayed

In[27]:= Map[equal[0, composite[Id, complement[#]]] &, SubstTest[class,
            pair[x, y], or[member[pair[x, y], w], not[member[x, u]], not[member[y, v]]],
            {u → even, v → odd, w → image[inverse[NATADD], odd]}]] // Reverse

Out[27]= subclass[cart[even, odd], image[inverse[NATADD], odd]] == True

In[28]:= % /. Equal → SetDelayed
```

Corollaries made into temporary rewrite rules.

```
In[29]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → cart[odd, even], v → image[inverse[NATADD], odd],
           w → complement[image[inverse[NATADD], even]]}]

Out[29]= equal[0, intersection[even, image[image[inverse[NATADD], even], odd]]] == True

In[30]:= intersection[even, image[image[inverse[NATADD], even], odd]] := 0

In[31]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → cart[even, odd], v → image[inverse[NATADD], odd],
           w → complement[image[inverse[NATADD], even]]}]

Out[31]= equal[0, intersection[odd, image[image[inverse[NATADD], even], even]]] == True

In[32]:= intersection[odd, image[image[inverse[NATADD], even], even]] := 0
```

domain and range lemmas for image[inverse[NATADD],x]

Domain lemma.

```
In[33]:= Map[or[#, subclass[domain[image[inverse[NATADD], x]], omega]] &,
          SubstTest[subclass, image[inverse[z], x], domain[z], z → NATADD]]

Out[33]= subclass[domain[image[inverse[NATADD], x]], omega] == True

In[34]:= subclass[domain[image[inverse[NATADD], x_]], omega] := True
```

Range lemma.

```
In[35]:= SubstTest[subclass, image[inverse[z], x], domain[z], z → NATADD]

Out[35]= subclass[range[image[inverse[NATADD], x]], omega] == True

In[36]:= subclass[range[image[inverse[NATADD], x_]], omega] := True
```

inverse image rules

Lemma.

```
In[37]:= SubstTest[subclass, image[inverse[NATADD], even], intersection[u, v, w],
          {u → cart[omega, omega],
           v -> complement[cart[even, odd]], w → complement[cart[odd, even]]}]

Out[37]= subclass[image[inverse[NATADD], even], union[cart[even, even], cart[odd, odd]]] == True

In[38]:= % /. Equal → SetDelayed
```

Theorem.

```
In[39]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> image[inverse[NATADD], even], v -> union[cart[even, even], cart[odd, odd]]}]

Out[39]= True == equal[image[inverse[NATADD], even], union[cart[even, even], cart[odd, odd]]]

In[40]:= image[inverse[NATADD], even] := union[cart[even, even], cart[odd, odd]]
```

Corollary.

```
In[41]:= SubstTest[intersection, image[inverse[NATADD], u],
            image[inverse[NATADD], v], {u → omega, v → complement[even]}] // Reverse

Out[41]= image[inverse[NATADD], odd] == union[cart[even, odd], cart[odd, even]]

In[42]:= image[inverse[NATADD], odd] := union[cart[even, odd], cart[odd, even]]
```