powers of odd numbers

Johan G. F. Belinfante
2006 August 16

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

        :Package Title: goedel84.16b       2006 August 16 at 10:30 a.m.

        It is now:  2006 Aug 16 at 11:38

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 August 15

        weightlimit = 40
```

summary

It is shown by induction that every power of an odd number is odd.

the induction argument

In this section, an odd number **x** is fixed, and induction is applied to the exponent **w**. The first step involves removing **nat** wrappers:

```
In[2]:= SubstTest[implies, and[equal[x, nat[u]], equal[w, nat[v]]],
         implies[and[member[x, odd], member[natexp[x, w], odd]],
          member[natexp[x, succ[w]], odd]], {u → x, v → w}]

Out[2]= or[member[natexp[x, succ[w]], odd], not[member[w, omega]],
          not[member[x, odd]], not[member[natexp[x, w], odd]]] == True

In[3]:= (% /. {x → x_, w → w_}) /. Equal → SetDelayed
```

Eliminating the variable **w** yields:

```
In[4]:= Map[equal[V, #] &, SubstTest[class, w, implies[
            and[member[w, omega], member[x, V], member[x, v], member[w, u]], member[succ[w], u]],
           {u -> image[image[inverse[NATEXP], odd], set[x]], v → odd}]] // Reverse

Out[4]= or[not[member[x, odd]], subclass[
           intersection[omega, image[SUCC, image[image[inverse[NATEXP], odd], set[x]]]],
           image[image[inverse[NATEXP], odd], set[x]]]] == True

In[5]:= (% /. x → x_) /. Equal → SetDelayed
```

Using induction, one finds:

```
In[6]:= SubstTest[implies, and[member[0, z], subclass[intersection[omega, image[SUCC, z]], z]],
         subclass[omega, z], z → intersection[image[V, intersection[odd, set[x]]],
          image[image[inverse[NATEXP], odd], set[x]]]]

Out[6]= or[not[member[x, odd]],
          subclass[omega, image[image[inverse[NATEXP], odd], set[x]]]] == True

In[7]:= (% /. x → x_) /. Equal → SetDelayed
```

eliminating the variable x

The variable **x** is eliminated as follows:

```
In[8]:= Map[equal[V, #] &,
          SubstTest[class, x, or[not[member[x, u]], subclass[omega, image[v, set[x]]]],
           {u → odd, v -> image[inverse[NATEXP], odd]}]] // Reverse

Out[8]= subclass[cart[odd, omega], image[inverse[NATEXP], odd]] == True

In[9]:= % /. Equal → SetDelayed
```

Taking the image with **NATEXP** yields this inclusion:

```
In[10]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → cart[odd, omega], v -> image[inverse[NATEXP], odd], w → NATEXP}]

Out[10]= subclass[image[NATEXP, cart[odd, omega]], odd] == True

In[11]:= % /. Equal → SetDelayed
```

Lemma.

```
In[12]:= ImageComp[NATEXP, RIGHT[set[0]], x] // Reverse

Out[12]= image[NATEXP, cart[x, set[set[0]]]] == intersection[omega, x]

In[13]:= image[NATEXP, cart[x_, set[set[0]]]] := intersection[omega, x]
```

The observation that the first power of any number is itself yields an inclusion in the opposite direction.

```
In[14]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → set[set[0]], v → omega, w → composite[NATEXP, id[cart[odd, V]], inverse[SECOND]]}]

Out[14]= subclass[odd, image[NATEXP, cart[odd, omega]]] == True

In[15]:= % /. Equal → SetDelayed
```

Combining these inclusions yields this equation:

In[16]:=
```
SubstTest[and, subclass[u, v], subclass[v, u],
  {u -> odd, v -> image[NATEXP, cart[odd, omega]]}]
```

Out[16]=
```
True == equal[odd, image[NATEXP, cart[odd, omega]]]
```

In[17]:=
```
image[NATEXP, cart[odd, omega]] := odd
```