uniqueness of square roots

Johan G. F. Belinfante
2003 May 18

```
In[1]:= << goedel52.r74; << tools.m

        :Package Title: goedel52.r74         2003 May 16 at 9:55 p.m.

        It is now:  2003 May 22 at 12:1

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 May 21

        weightlimit = 40
```

■ summary

This notebook contains a derivation of the fact that a natural number can have at most one square root. The idea is basically this: if $x^2 = y^2$, then $0 = x^2 - y^2 = (x + y)(x - y)$, which implies that either $x - y = 0$ or $x + y = 0$. If $x - y = 0$, then $x = y$. If $x + y = 0$, then $x = y = 0$. A variable−free version of this result is also derived; this version states that the function **composite[NATMUL,DUP]** is one−to−one.

■ a simplification rule

A preliminary simplification rule is derived in this section. This fact...

```
In[2]:= equiv[or[and[equal[x, y], member[x, omega], member[y, omega]],
          and[equal[0, x], equal[0, y], member[x, omega], member[y, omega]]],
        and[equal[x, y], member[x, omega], member[y, omega]]]

Out[2]= True
```

... justifies adding this simplification rule:

```
In[3]:= or[and[equal[x_, y_], member[x_, omega], member[y_, omega]],
          and[equal[0, x_], equal[0, y_], member[x_, omega], member[y_, omega]]] :=
         and[equal[x, y], member[x, omega], member[y, omega]]
```

■ the basic idea

The basic idea of the derivation is carried out in this fashion:

```
In[4]:= SubstTest[equal, 0, natmul[u, v], {u -> natadd[x, y], v -> natsub[x, y]}]

Out[4]= or[and[equal[natmul[x, x], natmul[y, y]], member[x, omega],
          member[y, omega], subclass[y, x]], and[equal[0, x], equal[0, y],
          equal[natmul[x, x], natmul[y, y]], member[x, omega], member[y, omega]]] ==
        and[equal[x, y], member[x, omega], member[y, omega]]
```

This result can be simplified:

```
In[5]:= Map[implies[and[equal[natmul[x, x], natmul[y, y]],
            member[x, omega], member[y, omega], subclass[y, x]], #] &, %] // Reverse

Out[5]= or[equal[x, y], not[equal[natmul[x, x], natmul[y, y]]],
          not[member[x, omega]], not[member[y, omega]], not[subclass[y, x]]] == True

In[6]:= or[equal[x_, y_], not[equal[natmul[x_, x_], natmul[y_, y_]]],
          not[member[x_, omega]], not[member[y_, omega]], not[subclass[y_, x_]]] := True
```

The hypothesis **subclass[y,x]** is removed by using dichotomy:

```
In[7]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p5],
          implies[and[p1, p2, p4], p5], implies[p2, or[p3, p4]],
               not[implies[and[p1, p2], p5]],
            {p1 -> equal[natmul[x, x], natmul[y, y]],
             p2 -> and[member[x, omega], member[y, omega]],
               p3 -> subclass[x, y], p4 -> subclass[y, x], p5 -> equal[x, y]}]]

Out[7]= or[equal[x, y], not[equal[natmul[x, x], natmul[y, y]]],
          not[member[x, omega]], not[member[y, omega]]] == True
```

This completes the derivation of the uniqueness theorem for square roots:

```
In[8]:= or[equal[x_, y_], not[equal[natmul[x_, x_], natmul[y_, y_]]],
          not[member[x_, omega]], not[member[y_, omega]]] := True
```

A negative form of this will also be needed below:

```
In[9]:= and[equal[natmul[x, x], natmul[y, y]],
          member[x, omega], member[y, omega], not[equal[x, y]]] // NotNotTest

Out[9]= and[equal[natmul[x, x], natmul[y, y]],
          member[x, omega], member[y, omega], not[equal[x, y]]] == False

In[10]:= and[equal[natmul[x_, x_], natmul[y_, y_]],
           member[x_, omega], member[y_, omega], not[equal[x_, y_]]] := False
```

■ eliminating the variables x and y

The following membership rule involving **inverse[DUP]** is needed:

```
In[11]:= member[pair[x, y], composite[inverse[DUP], z]] // AssertTest

Out[11]= member[pair[x, y], composite[inverse[DUP], z]] ==
          and[member[x, V], member[y, V], member[pair[x, pair[y, y]], z]]

In[12]:= member[pair[x_, y_], composite[inverse[DUP], z_]] :=
          and[member[x, V], member[y, V], member[pair[x, pair[y, y]], z]]
```

The following simplification rule for **iterate** is useful:

```
In[13]:= Map[implies[#, member[y, V]] &,
          SubstTest[member, pair[x, y], composite[Id, w], w -> iterate[u, v]]]

Out[13]= or[member[y, V], not[member[pair[x, y], iterate[u, v]]]] == True

In[14]:= or[member[y_, V], not[member[pair[x_, y_], iterate[u_, v_]]]] := True
```

This fact implies:

```
In[15]:= equiv[and[member[y, V], member[pair[x, y], iterate[u, v]]],
          member[pair[x, y], iterate[u, v]]]

Out[15]= True
```

A temporary simplification rule expressing this fact is added:

```
In[16]:= and[member[y_, V], member[pair[x_, y_], iterate[u_, v_]]] :=
          member[pair[x, y], iterate[u, v]]
```

To connect **NATMUL** with **natmul** one needs to deal with a messy expression involving **iterate**:

```
In[17]:= SubstTest[member, pair[x, y], image[inverse[NATMUL], z], z -> singleton[w]]

Out[17]= and[member[x, omega],
           member[pair[y, w], iterate[iterate[SUCC, singleton[x]], singleton[0]]] ==
          and[equal[w, natmul[x, y]], member[x, omega], member[y, omega]]

In[18]:= and[member[x_, omega],
           member[pair[y_, w_], iterate[iterate[SUCC, singleton[x_]], singleton[0]]] :=
          and[equal[w, natmul[x, y]], member[x, omega], member[y, omega]]
```

From this, one derives a membership rule for **composite[inverse[NATMUL],NATMUL]**:

```
In[19]:= SubstTest[member, pair[x, y], image[inverse[NATMUL], image[w, singleton[z]]],
           {w -> NATMUL, z -> pair[u, v]}] // Reverse

Out[19]= member[pair[pair[u, v], pair[x, y]], composite[inverse[NATMUL], NATMUL]] ==
          and[equal[natmul[u, v], natmul[x, y]], member[u, omega],
           member[v, omega], member[x, omega], member[y, omega]]

In[20]:= member[pair[pair[u_, v_], pair[x_, y_]], composite[inverse[NATMUL], NATMUL]] :=
          and[equal[natmul[u, v], natmul[x, y]], member[u, omega],
           member[v, omega], member[x, omega], member[y, omega]]
```

The uniqueness theorem for square roots can now be cast in variable−free form by an application of **RelnNormality**:

```
In[21]:= Map[equal[0, #] &, intersection[Di,
            composite[inverse[DUP], inverse[NATMUL], NATMUL, DUP]] // RelnNormality]

Out[21]= subclass[composite[inverse[DUP], inverse[NATMUL], NATMUL, DUP], Id] == True

In[22]:= subclass[composite[inverse[DUP], inverse[NATMUL], NATMUL, DUP], Id] := True
```

It remains to rewrite this in terms of **FUNCTION**:

```
In[23]:= SubstTest[subclass, composite[inverse[x], x], Id, x -> composite[NATMUL, DUP]] // Reverse

Out[23]= FUNCTION[composite[inverse[DUP], inverse[NATMUL]]] == True

In[24]:= FUNCTION[composite[inverse[DUP], inverse[NATMUL]]] := True
```

That is, the function **composite{NATMUL,DUP]** is one−to−one:

```
In[25]:= ONEONE[composite[NATMUL, DUP]]

Out[25]= True
```