uniqueness theorem for natmod[x,y]

Johan G. F. Belinfante
2005 June 27

```
In[1]:= SetDirectory["i:"]; << goedel70.27a; << tools.m

        :Package Title: goedel70.27a       2005 June 27 at 10:40 a.m.

        It is now:  2005 Jun 27 at 16:42

        Loading Simplification Rules

        TOOLS.M                       Revised 2005 June 17

        weightlimit = 40
```

summary

Call a natural number **z** a **remainder** for the division of a natural number **x** by a natural number **y** if **y** divides the difference **x – z**. By definition, **natmod[x,y]** is the least such remainder. The uniqueness theorem derived in this notebook asserts that if such a remainder is less than **y**, then it is the least remainder **natmod[x,y]**. This observation is sometimes proposed as a way to define **natmod[x,y]**, but doing so has the disadvantage that one would then need a separate definition for the case **y = 0**. When **y = 0** the only remainder is **x**.

```
In[2]:= "Ronald L. Graham, Donald E. Knuth and Oren
          Patashnik, Concrete Mathematics, Addison-Wesley Publishing
          Company, Reading, Massachusetts, 1989.  (See page 82)."

Out[2]= Ronald L. Graham, Donald E. Knuth and Oren
          Patashnik, Concrete Mathematics, Addison-Wesley Publishing
          Company, Reading, Massachusetts, 1989.  (See page 82).
```

difference of divisors

The difference of two divisors is a divisor.

```
In[3]:= Map[not,
         SubstTest[and, implies[p1, p4], implies[p2, p5], implies[and[p3, p4, p5], p6],
          implies[and[p1, p2, p6], p7], not[implies[and[p1, p2, p3], p7]],
          {p1 -> member[pair[y, u], DIV], p2 -> member[pair[y, v], DIV],
           p3 -> not[member[u, v]], p4 → member[u, omega], p5 → member[v, omega],
           p6 → subclass[v, u], p7 -> member[pair[y, natsub[u, v]], DIV]}]]

Out[3]= or[member[u, v], member[pair[y, natsub[u, v]], DIV],
          not[member[pair[y, u], DIV]], not[member[pair[y, v], DIV]]] == True

In[4]:= or[member[u_, v_], member[pair[y_, natsub[u_, v_]], DIV],
          not[member[pair[y_, u_], DIV]], not[member[pair[y_, v_], DIV]]] := True
```

Application:

```
In[5]:= Map[or[member[pair[nat[y], natsub[nat[z], natmod[nat[x], nat[y]]]], DIV], #] &,
         (SubstTest[implies, and[member[pair[nat[y], u], DIV], member[pair[nat[y], v],
               DIV], not[member[u, v]]], member[pair[nat[y], natsub[u, v]], DIV],
            {u → natsub[nat[x], nat[w]], v → natsub[nat[x], nat[z]]}] // MapNotNot) /.
          w → natmod[nat[x], nat[y]]]

Out[5]= or[member[natsub[nat[x], natmod[nat[x], nat[y]]], natsub[nat[x], nat[z]]],
          member[pair[nat[y], natsub[nat[z], natmod[nat[x], nat[y]]]], DIV],
          not[member[pair[nat[y], natsub[nat[x], nat[z]]], DIV]]] == True

In[6]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

A cancellation lemma for subtraction can be applied for the first literal in the preceding result.

```
In[7]:= SubstTest[implies, and[member[w, omega], member[natsub[u, v], natsub[u, w]]],
         or[member[u, w], member[w, v]],
         {u → nat[x], v -> natmod[nat[x], nat[y]], w → nat[z]}]

Out[7]= or[member[nat[x], nat[z]], member[nat[z], natmod[nat[x], nat[y]]], not[member[
            natsub[nat[x], natmod[nat[x], nat[y]]], natsub[nat[x], nat[z]]]]] == True

In[8]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

---

lemma

If **y | (x - z)**, then **x** cannot be less than **z**.

```
In[9]:= SubstTest[implies, member[pair[nat[y], w], DIV],
         member[w, omega], w → natsub[nat[x], nat[z]]]

Out[9]= or[not[member[nat[x], nat[z]]],
          not[member[pair[nat[y], natsub[nat[x], nat[z]]], DIV]]] == True

In[10]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

---

transposition lemma

If $\mathbf{z} < \mathbf{y}$, then also $\mathbf{z} - \mathbf{x} < \mathbf{y}$, provided the subtraction makes sense.

```
In[11]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, or[p3, p4]], not[implies[p1, or[p3, p4]]],
           {p1 -> member[nat[z], nat[y]], p2 → member[nat[z], natadd[nat[y], nat[x]]],
            p3 -> member[natsub[nat[z], nat[x]], nat[y]], p4 -> member[nat[z], nat[x]]}]]

Out[11]= or[member[nat[z], nat[x]],
           member[natsub[nat[z], nat[x]], nat[y]], not[member[nat[z], nat[y]]]] == True

In[12]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Application:

```
In[13]:= SubstTest[or, member[nat[z], nat[w]], member[natsub[nat[z], nat[w]], nat[y]],
          not[member[nat[z], nat[y]]], w → natmod[nat[x], nat[y]]]

Out[13]= or[member[nat[z], natmod[nat[x], nat[y]]],
           member[natsub[nat[z], natmod[nat[x], nat[y]]], nat[y]],
           not[member[nat[z], nat[y]]]] == True

In[14]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

---

least remainder property

Since **natmod[x,y]** is the least remainder, no other remainder can be lesser.

```
In[15]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 -> member[pair[nat[y], natsub[nat[x], nat[z]]], DIV],
            p2 → subclass[natmod[nat[x], nat[y]], nat[z]],
            p3 → not[member[nat[z], natmod[nat[x], nat[y]]]]}]]

Out[15]= or[not[member[nat[z], natmod[nat[x], nat[y]]]],
           not[member[pair[nat[y], natsub[nat[x], nat[z]]], DIV]]] == True

In[16]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

derivation of the uniqueness theorem

Putting together all these results yields the following uniqueness theorem.

```
In[17]:= Map[not, SubstTest[and, implies[p1, p3], implies[p1, p6],
           implies[and[p3, p6], p5], implies[and[p1, p5], p4],
           implies[and[p2, p6], p7], p1, p2, not[and[p4, p7]],
           {p1 -> member[pair[nat[y], natsub[nat[x], nat[z]]], DIV],
            p2 → member[nat[z], nat[y]], p3 → not[member[nat[x], nat[z]]],
            p4 -> member[pair[nat[y], natsub[nat[z], natmod[nat[x], nat[y]]]], DIV],
            p5 → not[member[natsub[nat[x], natmod[nat[x], nat[y]]],
               natsub[nat[x], nat[z]]]],
            p6 → not[member[nat[z], natmod[nat[x], nat[y]]]],
            p7 → member[natsub[nat[z], natmod[nat[x], nat[y]]], nat[y]]}]]

Out[17]= or[equal[nat[z], natmod[nat[x], nat[y]]], not[member[nat[z], nat[y]]],
           not[member[pair[nat[y], natsub[nat[x], nat[z]]], DIV]]] == True

In[18]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The **nat** wrappers can now be replaced by numberhood literals, which are then automatically removed:

```
In[19]:= SubstTest[implies, and[equal[u, nat[x]], equal[v, nat[y]],
           equal[w, nat[z]], member[w, v], member[pair[v, natsub[u, w]], DIV]],
          equal[w, natmod[u, v]], {u → x, v → y, w → z}]

Out[19]= or[equal[z, natmod[x, y]], not[member[z, y]],
           not[member[pair[y, natsub[x, z]], DIV]]] == True

In[20]:= or[equal[z_, natmod[x_, y_]], not[member[z_, y_]],
           not[member[pair[y_, natsub[x_, z_]], DIV]]] := True
```