cancellations for subtraction of natural numbers

Johan G. F. Belinfante
2005 June 27

```
In[1]:= SetDirectory["i:"]; << goedel70.24b; << tools.m
        :Package Title: goedel70.24b       2005 June 24 at 2:45 p.m.
        It is now:  2005 Jun 27 at 8:57
        Loading Simplification Rules
        TOOLS.M                       Revised 2005 June 17
        weightlimit = 40
```

summary

The following cancellation law holds for addition of natural numbers.

```
In[2]:= implies[and[member[z, omega], member[natadd[x, y], natadd[x, z]]], member[y, z]]
Out[2]= True
```

In this notebook, two analogous cancellation laws for subtraction are derived. Cancellation laws for subtraction require one additional literal which says that one cannot sensibly subtract a larger number from a smaller one. Note that both cancellation laws derived below contain a hypothesis that one difference is less than another one, with one variable in a common position for both differences, and both say that under certain conditions, one can in effect cancel the common variable. Note that the lesser difference automatically involves natural numbers, with a lesser number being subtracted from a greater one, but it is not automatically true that the greater difference makes sense. The upshot is that only one explicit numberhood condition is needed: one needs to know that the non-common variable in the greater difference is a natural number, and in addition, one needs to explicitly assume that for the greater difference one is not subtracting a larger number from a smaller one.

cancelling in x - y < x - z

The cancellation law for addition can be used to derive one for subtraction. For simplicity, **nat** wrappers are used on all variables.

```
In[3]:= Map[implies[#, or[member[nat[x], nat[z]], member[nat[z], nat[y]]]] &,
          SubstTest[member, natadd[u, v], natadd[u, w], {u → natadd[nat[y], nat[z]],
            v → natsub[nat[x], nat[y]], w → natsub[nat[x], nat[z]]}]] // Reverse

Out[3]= or[member[nat[x], nat[z]], member[nat[z], nat[y]],
          not[member[natsub[nat[x], nat[y]], natsub[nat[x], nat[z]]]]] == True

In[4]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The **nat** wrappers can be replaced with numberhood literals:

```
In[5]:= SubstTest[implies, and[equal[u, nat[x]], equal[v, nat[y]], equal[w, nat[z]]],
         or[member[u, w], member[w, v], not[member[natsub[u, v], natsub[u, w]]]],
         {u → x, v → y, w → z}]

Out[5]= or[member[x, z], member[z, y], not[member[x, omega]], not[member[y, omega]],
          not[member[z, omega]], not[member[natsub[x, y], natsub[x, z]]]] == True

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

This result can be cleaned up by removing two redundant numberhood literals.

```
In[7]:= Map[not, SubstTest[and, implies[p1, p4],
          implies[p1, p5], implies[and[p1, p2, p3, p4, p5], p6],
          not[implies[and[p1, p2, p3], p6]], {p1 -> member[natsub[x, y], natsub[x, z]],
           p2 -> member[z, omega], p3 → not[member[x, z]],
           p4 → member[x, omega], p5 → member[y, omega], p6 → member[z, y]}]]

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

In[8]:= or[member[x_, z_], member[z_, y_],
          not[member[natsub[x_, y_], natsub[x_, z_]]], not[member[z_, omega]]] := True
```

cancelling in x - z < y - z

The cancellation law for addition can be used to derive another one for subtraction. Again, **nat** wrappers are used on all variables.

```
In[9]:= Map[implies[#, or[member[nat[x], nat[y]], member[nat[y], nat[z]]]] &,
          SubstTest[member, natadd[u, w], natadd[v, w], {u -> natsub[nat[x], nat[z]],
            v -> natsub[nat[y], nat[z]], w → nat[z]}]] // Reverse

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

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

The **nat** wrappers can be replaced with numberhood literals:

```
In[11]:= SubstTest[implies, and[equal[u, nat[x]], equal[v, nat[y]], equal[w, nat[z]]],
           or[member[u, v], member[v, w], not[member[natsub[u, w], natsub[v, w]]]],
           {u → x, v → y, w → z}]

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

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

This result can be cleaned up by removing two redundant numberhood literals.

```
In[13]:= Map[not, SubstTest[and, implies[p1, p4],
           implies[p1, p5], implies[and[p1, p2, p3, p4, p5], p6],
           not[implies[and[p1, p2, p3], p6]], {p1 -> member[natsub[x, z], natsub[y, z]],
            p2 → member[y, omega], p3 → not[member[y, z]],
            p4 → member[x, omega], p5 → member[z, omega], p6 → member[x, y]}]]

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

In[14]:= or[member[x_, y_], member[y_, z_],
           not[member[natsub[x_, z_], natsub[y_, z_]]], not[member[y_, omega]]] := True
```