subclass[y, natmod[x,y]]

Johan G. F. Belinfante and Claudia Huang
2005 May 27

```
In[1]:= SetDirectory["i:"]; << goedel69.26a; << tools.m
```

```
:Package Title: goedel69.26a       2005 May 26 at 9:05 a.m.

It is now:  2005 May 27 at 11:5

Loading Simplification Rules

TOOLS.M                       Revised 2005 May 17

weightlimit = 40
```

---

summary

For natural numbers, **y** is less than **natmod[x,y]** only when **y = 0**. In this notebook, existing rewrite rules for this statement are generalized, removing the condition that **x** and **y** be natural numbers.

---

derivation

The rule with wrappers is this:

```
In[2]:= subclass[nat[y], natmod[nat[x], nat[y]]]

Out[2]= equal[0, nat[y]]
```

Removing the wrappers yields:

```
In[3]:= SubstTest[implies, and[equal[u, nat[x]], equal[v, nat[y]]],
         equiv[subclass[v, natmod[u, v]], equal[0, v]], {u → x, v → y}]

Out[3]= or[equal[0, y], not[member[x, omega]],
          not[member[y, omega]], not[subclass[y, natmod[x, y]]]] == True
```

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

In the opposite direction, one has:

```
In[5]:= SubstTest[implies, equal[v, V], subclass[y, v], v → natmod[x, y]]

Out[5]= or[and[member[x, omega], member[y, omega]], subclass[y, natmod[x, y]]] == True

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

Combining these, one obtains a logical equivalence that can be made into a rewrite rule:

```
In[7]:= equiv[subclass[y, natmod[x, y]],
          or[equal[0, y], not[member[x, omega]], not[member[y, omega]]]]

Out[7]= True

In[8]:= subclass[y_, natmod[x_, y_]] :=
          or[equal[0, y], not[member[x, omega]], not[member[y, omega]]]
```

---

variable-free formulation

Lemma 1.

```
In[9]:= AssInt[composite[E, FIRST], rotate[NATMOD], cart[cart[V, V], omega]] // Reverse

Out[9]= composite[id[omega], intersection[composite[E, FIRST], rotate[NATMOD]]] ==
          intersection[composite[E, FIRST], rotate[NATMOD]]

In[10]:= % /. Equal → SetDelayed
```

Lemma 2.

```
In[11]:= composite[intersection[composite[E, FIRST], rotate[NATMOD]],
           id[cart[V, omega]]] // TripleRotate

Out[11]= composite[intersection[composite[E, FIRST], rotate[NATMOD]],
           id[cart[V, omega]]] == intersection[composite[E, FIRST], rotate[NATMOD]]

In[12]:= % /. Equal → SetDelayed
```

A variable-free formulation can be obtained by using reification.

```
In[13]:= Map[inverse[reify[y, #]] &, Map[domain[reify[x, #]] &,
           image[V, intersection[y, set[natmod[x, y]]]]] // Normality]

Out[13]= composite[intersection[composite[E, FIRST], rotate[NATMOD]],
           inverse[SECOND]] == cart[omega, intersection[omega, complement[set[0]]]]

In[14]:= composite[intersection[composite[E, FIRST], rotate[NATMOD]],
           inverse[SECOND]] := cart[omega, intersection[omega, complement[set[0]]]]
```