NATMOD fixed point theorems

Johan G. F. Belinfante
2005 June 24

```
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 24 at 15:0

        Loading Simplification Rules

        TOOLS.M                      Revised 2005 June 17

        weightlimit = 40
```

summary

Some fixed point theorems concerning **natmod** are derived.

basic result

Recall that **natmod[x,y] = x** when **x < y**.

```
In[2]:= implies[and[member[x, y], member[y, omega]], equal[natmod[x, y], x]]

Out[2]= True
```

Combining this result with a converse result yields this logical equivalence:

```
In[3]:= equiv[equal[x, natmod[x, y]], or[and[member[x, y], member[y, omega]],
            and[member[x, omega], equal[0, y]], equal[x, V]]] // not // not

Out[3]= True
```

This can be made into a rewrite rule.

```
In[4]:= equal[x_, natmod[x_, y_]] :=
         or[and[equal[0, y], member[x, omega]], and[member[x, y], member[y, omega]], equal[V, x]]
```

variable−free formulation

The following variable−free statement holds:

```
In[5]:= fix[composite[inverse[FIRST], NATMOD]] // RelnNormality

Out[5]= fix[composite[inverse[FIRST], NATMOD]] ==
         union[cart[omega, set[0]], composite[id[omega], E]]

In[6]:= fix[composite[inverse[FIRST], NATMOD]] :=
         union[cart[omega, set[0]], composite[id[omega], E]]
```

Companion result.

```
In[7]:= fix[composite[inverse[NATMOD], FIRST]] // InvertFixTest

Out[7]= fix[composite[inverse[NATMOD], FIRST]] ==
         union[cart[omega, set[0]], composite[id[omega], E]]

In[8]:= fix[composite[inverse[NATMOD], FIRST]] :=
         union[cart[omega, set[0]], composite[id[omega], E]]
```

idempotence result

Corollary.

```
In[9]:= equal[natmod[natmod[x, y], y], natmod[x, y]]

Out[9]= True

In[10]:= natmod[natmod[x_, y_], y_] := natmod[x, y]
```

Lemma.

```
In[11]:= symdif[composite[NATMOD, RIGHT[x], NATMOD, RIGHT[x]],
           composite[NATMOD, RIGHT[x]]] // VSNormality

Out[11]= union[composite[
            intersection[complement[NATMOD], composite[NATMOD, RIGHT[x], NATMOD]], RIGHT[x]],
           composite[id[union[intersection[complement[image[V, intersection[omega, set[x]]]],
               image[V, set[x]]], intersection[complement[omega], complement[x],
               image[V, intersection[omega, set[x]]]], intersection[complement[x],
               image[V, x], image[V, intersection[omega, set[x]]]]]], NATMOD, RIGHT[x]]] == 0

In[12]:= (% /. x → x_) /. Equal → SetDelayed

In[13]:= SubstTest[equal, 0, symdif[u, v],
          {u -> composite[NATMOD, RIGHT[x], NATMOD, RIGHT[x]], v -> composite[NATMOD, RIGHT[x]]}]

Out[13]= True ==
          equal[composite[NATMOD, RIGHT[x]], composite[NATMOD, RIGHT[x], NATMOD, RIGHT[x]]]

In[14]:= composite[NATMOD, RIGHT[x_], NATMOD, RIGHT[x_]] := composite[NATMOD, RIGHT[x]]
```

consequences of the idempotence result

The following temporary result will be subsumed by a more general corollary in a moment.

```
In[15]:= SubstTest[implies, and[FUNCTION[y], idempotent[y]],
          equal[range[y], fix[y]], y → composite[NATMOD, RIGHT[x]]]

Out[15]= equal[image[NATMOD, cart[V, set[x]]],
           union[intersection[omega, complement[image[V, x]]], nat[x]]] == True

In[19]:= image[NATMOD, cart[V, set[x_]]] :=
          union[intersection[omega, complement[image[V, x]]], nat[x]]
```

The variable **x** can be eliminated.

```
In[20]:= SubstTest[reify, x, image[z, cart[V, set[x]]], z → NATMOD] // Reverse

Out[20]= composite[NATMOD, inverse[SECOND]] ==
          union[cart[set[0], omega], composite[inverse[E], id[omega]]]

In[21]:= composite[NATMOD, inverse[SECOND]] :=
          union[cart[set[0], omega], composite[inverse[E], id[omega]]]
```

The following **image** rule subsumes the temporary rule derived earlier.

```
In[22]:= ImageComp[NATMOD, inverse[SECOND], x] // Reverse

Out[22]= image[NATMOD, cart[V, x]] == union[
           intersection[omega, image[V, intersection[x, set[0]]]], U[intersection[omega, x]]]

In[23]:= image[NATMOD, cart[V, x_]] := union[
           intersection[omega, image[V, intersection[x, set[0]]]], U[intersection[omega, x]]]
```

another variable−free rule

Lemma.

```
In[24]:= composite[rotate[composite[intersection[composite[rotate[x], SWAP, FIRST],
              composite[rotate[y], SWAP, SECOND]], inverse[RIF], SWAP]], SWAP] // VSTriNormality

Out[24]= composite[rotate[composite[intersection[composite[rotate[x], SWAP, FIRST],
              composite[rotate[y], SWAP, SECOND]], inverse[RIF], SWAP]], SWAP] == composite[x,
           intersection[composite[inverse[FIRST], y], composite[inverse[SECOND], SECOND]]]

In[25]:= composite[rotate[composite[intersection[composite[rotate[x_], SWAP, FIRST],
              composite[rotate[y_], SWAP, SECOND]], inverse[RIF], SWAP]], SWAP] := composite[x,
           intersection[composite[inverse[FIRST], y], composite[inverse[SECOND], SECOND]]]
```

This can be cleaned up:

```
In[26]:= Map[flip[rotate[inverse[#]]] &,
           SubstTest[reify, x, composite[z, RIGHT[x], z, RIGHT[x]], z → NATMOD]] // Reverse

Out[26]= composite[NATMOD, intersection[
            composite[inverse[FIRST], NATMOD], composite[inverse[SECOND], SECOND]]] == NATMOD

In[27]:= composite[NATMOD, intersection[composite[inverse[FIRST], NATMOD],
            composite[inverse[SECOND], SECOND]]] := NATMOD
```