normalizing NATMUL and DIV

Johan G. F. Belinfante
2005 February 18

summary

The definition of multiplication for natural numbers involves a double iteration, which often gets expanded out in various ways when class rules are applied. It appears difficult to control this expansion via normalization rewrite rules. This notebook explores the idea of hiding the double-iteration construction in the definition of natural number multiplication by removing the explicit membership rules for the function NATMUL in favor of rewrite rules for image[V, intersection[NATMUL, set[x]]]. The connection between the membership rule and this expression is made evident by the following observation:

\[
\text{In}[2] := \text{class}[w, \text{member}[y, x]]
\]
\[
\text{Out}[2] = \text{image}[V, \text{intersection}[x, \text{set}[y]]]
\]

The removed membership rules can still be rederived on demand using assert, if desired. With the new rewrite rules in place, it becomes easy to normalize the function NATMUL as well as the associated divisibility relation DIV. Left untouched however is the long outstanding problem of finding a simple equational definition of NATMUL that could be used for automated reasoning about multiplication of natural numbers using a program such as Otter.
pair rule

The **GOEDEL** program actually contains two separate membership rules for the function **NATMUL** for multiplication of natural numbers, both of which will be removed. One must start with the more specialized rule, which involves membership of ordered pairs.

```math
In[3]:= member[ pair[ pair[x, y], z], NATMUL]
```

```math
Out[3]= \text{and}[\text{member}[x, \text{omega}],
\text{member}[\text{pair}[y, z], \text{iterate}[\text{iterate}[\text{SUCC}, \text{set}[x]], \text{set}[0]])]
```

To obtain a simple replacement for this rule, the following lemma is helpful.

```math
In[4]:= \text{image}[V, \text{intersection}[\text{iterate}[x, \text{set}[0]], \text{set}[y]]] // \text{Normality} // \text{Reverse}
```

```math
Out[4]= \text{image}[V, \text{intersection}[
\text{complement}[\text{P}[\text{complement}[\text{set}[y]]]], \text{P}[\text{cart}[\text{omega}, V]], \text{subvar}[\text{union}[
\text{cart}[\text{cart}[\text{set}[0], \text{set}[0]], \text{cart}[\text{set}[0], \text{set}[0]]], \text{cross}[\text{SUCC}, x]]]) =
\text{image}[V, \text{intersection}[\text{iterate}[x, \text{set}[0]], \text{set}[y]]]
```

```math
In[5]:= \text{image}[V, \text{intersection}[
\text{complement}[\text{P}[\text{complement}[\text{set}[y_]]]], \text{P}[\text{cart}[\text{omega}, V]], \text{subvar}[\text{union}[
\text{cart}[\text{cart}[\text{set}[0], \text{set}[0]], \text{cart}[\text{set}[0], \text{set}[0]]], \text{cross}[\text{SUCC}, x_]])] :=
\text{image}[V, \text{intersection}[\text{iterate}[x, \text{set}[0]], \text{set}[y]]]
```

The desired replacement rule can now be derived.

```math
In[6]:= \text{SubstTest}[\text{class}, w, \text{member}[\text{pair}[\text{pair}[x, y], z], t], t \rightarrow \text{NATMUL}] // \text{Reverse}
```

```math
Out[6]= \text{image}[V, \text{intersection}[\text{NATMUL}, \text{set}[[\text{pair}[\text{pair}[x, y], z]]]] =
\text{intersection}[\text{image}[V, \text{intersection}[\text{omega}, \text{set}[x]]], \text{image}[V,
\text{intersection}[\text{iterate}[\text{iterate}[\text{SUCC}, \text{set}[x]], \text{set}[0]], \text{set}[[\text{pair}[y, z]]]]]
```

This rule will be made permanent.

```math
In[7]:= \text{image}[V, \text{intersection}[\text{NATMUL}, \text{set}[[\text{pair}[\text{pair}[x_, y_], z_]]]] :=
\text{intersection}[\text{image}[V, \text{intersection}[\text{omega}, \text{set}[x]]], \text{image}[V,
\text{intersection}[\text{iterate}[\text{iterate}[\text{SUCC}, \text{set}[x]], \text{set}[0]], \text{set}[[\text{pair}[y, z]]]]]
```

The membership rule involving pairs can now be removed.

```math
In[8]:= \text{member}[\text{pair}[\text{pair}[x_, y_], z_], \text{NATMUL}] =.
```
The general membership rule

The **GOEDEL** program also contains a more general membership rule that does not explicitly involve **pair**:

\[
\text{In}[9]:= \text{member}[x, \text{NATMUL}]
\]

\[
\text{Out}[9]= \text{and}[\text{member}[\text{first}[\text{first}[x]], \text{omega}], \text{member}[\text{pair}[\text{second}[\text{first}[x]], \text{second}[x]], \\
\text{iterate}[\text{iterate}[\text{SUCC}, \text{set}[\text{first}[\text{first}[x]]], \text{set}[0]]])
\]

The replacement rule is derived:

\[
\text{In}[10]:= \text{image}[V, \text{intersection}[\text{NATMUL}, \text{set}[x]]] \text{ // Normality}
\]

\[
\text{Out}[10]= \text{image}[V, \text{intersection}[\text{NATMUL}, \text{set}[x]]] = \\
\text{intersection}[\text{image}[V, \text{intersection}[\text{omega}, \text{set}[\text{first}[\text{first}[x]]]]], \\
\text{image}[V, \text{intersection}[\text{iterate}[\text{iterate}[\text{SUCC}, \text{set}[\text{first}[\text{first}[x]]], \text{set}[0]], \\
\text{set}[\text{pair}[\text{second}[\text{first}[x]], \text{second}[x]]])]]]
\]

\[
\text{In}[11]:= \text{image}[V, \text{intersection}[\text{NATMUL}, \text{set}[x_\_]]] := \\
\text{intersection}[\text{image}[V, \text{intersection}[\text{omega}, \text{set}[\text{first}[\text{first}[x]]]]], \\
\text{image}[V, \text{intersection}[\text{iterate}[\text{iterate}[\text{SUCC}, \text{set}[\text{first}[\text{first}[x]]], \text{set}[0]], \\
\text{set}[\text{pair}[\text{second}[\text{first}[x]], \text{second}[x]]])]]]
\]

The membership rule can now be removed:

\[
\text{In}[12]:= \text{member}[x\_, \text{NATMUL}] = .
\]

how to resurrect the removed rules if need be

If one does need either one of the two removed membership rules, these can be quickly resurrected on demand using **assert**.

\[
\text{In}[13]:= \text{member}[x, \text{NATMUL}] \text{ // AssertTest}
\]

\[
\text{Out}[13]= \text{member}[x, \text{NATMUL}] = \text{and}[\text{member}[\text{first}[\text{first}[x]], \text{omega}], \\
\text{member}[\text{pair}[\text{second}[\text{first}[x]], \text{second}[x]], \\
\text{iterate}[\text{iterate}[\text{SUCC}, \text{set}[\text{first}[\text{first}[x]]], \text{set}[0]]])]
\]

\[
\text{In}[14]:= \text{member}[\text{pair}[\text{pair}[x, y], z], \text{NATMUL}] \text{ // AssertTest}
\]

\[
\text{Out}[14]= \text{member}[\text{pair}[\text{pair}[x, y], z], \text{NATMUL}] = \text{and}[\text{member}[x, \text{omega}], \\
\text{member}[\text{pair}[y, z], \text{iterate}[\text{iterate}[\text{SUCC}, \text{set}[x]], \text{set}[0]]])
\]
normalization rules for NATMUL

The function NATMUL does not require much in the way of normalization rules. One does need one new rule to cope with this expression:

In[15]:= NATMUL // VSTriNormality

Out[15]= NATMUL = union[NATMUL, composite[S, NATMUL,
   id[union[cart[V, complement[omega]], cart[complement[omega], V]]]]]

The following cure is ad hoc, but it suffices.

In[16]:= Assoc[NATMUL, id[cart[omega, omega]], id[   union[cart[V, complement[omega]], cart[complement[omega], V]]]] // Reverse

Out[16]= composite[NATMUL,
   id[union[cart[V, complement[omega]], cart[complement[omega], V]]]] = 0

In[17]:= composite[NATMUL,
   id[union[cart[V, complement[omega]], cart[complement[omega], V]]]] := 0

There still remain problems with expressions such as rotate[NATMUL], but these do not appear to be severe.

In[18]:= rotate[NATMUL] // VSTriNormality // Reverse

Out[18]= composite[id[omega],
   intersection[complement[composite[SECOND, id[cart[V, omega]]],
   intersection[composite[inverse[FIRST], SECOND],
   composite[inverse[NATMUL], complement[S, FIRST]]],
   complement[rotate[composite[complement[S], NATMUL]]],
   id[cart[V, omega]]] = rotate[NATMUL]

In[19]:= composite[id[omega],
   intersection[complement[composite[SECOND, id[cart[V, omega]]],
   intersection[composite[inverse[FIRST], SECOND],
   composite[inverse[NATMUL], complement[S, FIRST]]],
   complement[rotate[composite[complement[S], NATMUL]]],
   id[cart[V, omega]]] := rotate[NATMUL]
normalization rules for \textsc{DIV}

To normalize the divisibility relation \textsc{DIV}, one needs a few new rewrite rules. This is one:

\begin{verbatim}
In[20]:= Assoc[NATMUL, id[cart[omega, omega]],
                  id[cart[x, complement[omega]]]] // Reverse
Out[20]= composite[NATMUL, id[cart[x, complement[omega]]]] = 0
In[21]:= composite[NATMUL, id[cart[x_, complement[omega]]]] := 0
\end{verbatim}

Various inverses are needed of existing rules:

\begin{verbatim}
In[22]:= composite[SWAP, inverse[NATMUL]] // DoubleInverse
Out[22]= composite[SWAP, inverse[NATMUL]] = inverse[NATMUL]
In[23]:= composite[SWAP, inverse[NATMUL]] := inverse[NATMUL]
\end{verbatim}

Here is another:

\begin{verbatim}
In[24]:= composite[id[cart[omega, V]], inverse[NATMUL]] // DoubleInverse
In[25]:= composite[id[cart[omega, V]], inverse[NATMUL]] := inverse[NATMUL]
\end{verbatim}

And another:

\begin{verbatim}
In[26]:= composite[FIRST, inverse[NATMUL]] // DoubleInverse
Out[26]= composite[FIRST, inverse[NATMUL]] = inverse[DIV]
In[27]:= composite[FIRST, inverse[NATMUL]] := inverse[DIV]
\end{verbatim}

The main new rule needed is this one:

\begin{verbatim}
In[28]:= fix[composite[complement[inverse[rotate[composite[complement[S], NATMUL]]]],
                  id[omega], complement[
                  rotate[composite[complement[inverse[S]], NATMUL]]]]] // Renormality
Out[28]= fix[composite[
                  complement[inverse[rotate[composite[complement[S], NATMUL]]]], id[omega],
                  complement[rotate[composite[complement[inverse[S]], NATMUL]]]]] =
                  union[complement[cart[V, omega]], inverse[DIV]]
\end{verbatim}
In[29]:= `fix[composite[
    complement[inverse[rotate[composite[complement[S], NATMUL]]], id[omega],
    complement[rotate[composite[complement[inverse[S]], NATMUL]]]] :=
union[complement[cart[V, omega]], inverse[DIV]]
]

This suffices for DIV to pass all six tests in the RelnNormalityBattery. Execution
times are quite reasonable.

In[30]:= DIV // RelnNormality // Timing

Out[30]= {5.297 Second, True}