Mixed multiplication, part 2. Distributive laws.

Johan G. F. Belinfante
2006 December 14

In introduction and summary

This notebook is concerned with distributive laws for mixed multiplication `MIXMUL` of integers times natural numbers. The function `MIXMUL` satisfies two distinct distributive laws, one involving sums of integers and the other involving sums of natural numbers. Since mixed multiplication was defined in terms of binary homomorphisms from `NATADD` to `INTADD`, the latter distributive law is an immediate consequence:

\[
\text{In [2]} := \text{composite[INTADD, cross[MIXMUL, MIXMUL], TWIST, cross[DUP, Id]]}
\]

\[
\text{Out [2]} := \text{composite[MIXMUL, cross[Id, NATADD]]}
\]

In this notebook, the other distributive law is derived as a consequence of the fact that sums of homomorphisms from `NATADD` to `INTADD` are homomorphisms and that any such homomorphism is uniquely determined by its value at the natural number \(1 = \text{set[0]}\).

map[\text{omega}, \text{Z}] is closed under sums

In this section it is shown that the sum of any two mappings from \text{omega} to \text{Z} is another one. If \(f\) and \(g\) are functions from \text{omega} to \text{Z}, then their sum is the function \(h = f + g\) defined by \(h(n) = f(n) + g(n)\). Note that the natural number variable \(n\) occurs only once on the left, but twice on the right. The sum of mappings \(f\) and \(g\) is defined to be `composite[INTADD, cross[f, g], DUP]`. The following temporary abbreviation will be used:

\[
\text{In [3]} := \text{intsum[x_, y_]} := \text{composite[INTADD,}
\]
\[
\text{intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]]}
\]

Lemma. The sum is a set.
Lemma. The sum is a function.

In [6]:  SubstTest[implies, and[equal[x, funpart[u]], equal[y, funpart[v]]],
      FUNCTION[composite[INTADD, intersection[composite[inverse[FIRST], x],
      composite[inverse[SECOND], y]]], {u -> x, v -> y}] // Reverse
Out [6]:  or[FUNCTION[composite[INTADD, intersection[composite[inverse[FIRST], x],
                        composite[inverse[SECOND], y]]],
       not[FUNCTION[x]], not[FUNCTION[y]]] = True

In [7]:  (% /. {x -> x, y -> y}) /. Equal -> SetDelayed

Corollary.

In [8]:  Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
      not[implies[p1, p4]], {p1 -> and[member[x, map[omega, Z]], member[y, map[omega, Z]]],
      p2 -> FUNCTION[x], p3 -> FUNCTION[y], p4 -> FUNCTION[composite[INTADD, intersection[composite[inverse[FIRST], x],
                        composite[inverse[SECOND], y]]]]}] // Reverse
Out [8]:  or[FUNCTION[composite[INTADD, intersection[composite[inverse[FIRST], x],
                        composite[inverse[SECOND], y]]],
       not[member[x, map[omega, Z]]], not[member[y, map[omega, Z]]]] = True

In [9]:  (% /. {x -> x, y -> y}) /. Equal -> SetDelayed

Lemma.

In [10]:  Map[not, SubstTest[and, implies[p1, p2],
        implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
        {p1 -> member[x, map[omega, Z]], p2 -> equal[domain[x], omega], p3 ->
         subclass[range[x], Z], p4 -> equal[domain[x], image[inverse[x], Z]]}] // Reverse
Out [10]:  or[equal[domain[x], image[inverse[x], Z]], not[member[x, map[omega, Z]]]] = True

In [11]:  (% /. x -> x) /. Equal -> SetDelayed

Lemma about the domain. Comment: this derivation is expedited by omitting the following steps of the proof: 
implies[and[p2,p5],p7], implies[and[p1,p4],p6], implies[p1,p4], implies[p2,p5]. Adding these steps increases the 
execution time from 5.9 seconds to 8.7 seconds.
Lemma. \textit{Out} [16] = \textit{In} [16]:=

Theorem. The set \textit{map}[\textit{omega}, \textit{Z}] is closed under the sum operation:

\begin{verbatim}
In[14]:= Map[not, SubstTest[and, implies[and[p3, p4, p5, p6, p7], p8],
                  not[implies[and[p1, p2, p3], p8]], {p1 \to member[x, map[omega, Z]],
                  p2 \to member[y, map[omega, Z]], p3 \to equal[z, composite[INTADD, intersection[composite[inverse[\textit{FIRST}], x], composite[inverse[\textit{SECOND}], y]]]],
                  p4 \to equal[domain[x], omega], p5 \to equal[domain[y], omega],
                  p6 \to equal[domain[x], image[inverse[x], \textit{Z}]],
                  p7 \to equal[domain[y], image[inverse[y], \textit{Z}]], p8 \to equal[domain[z], omega]]] /.
                  {p1 \to and[member[x, map[omega, Z]], member[y, map[omega, Z]]],
                  p2 \to equal[z, composite[INTADD, intersection[composite[inverse[\textit{FIRST}], x], composite[inverse[\textit{SECOND}], y]]]],
                  p3 \to member[z, \textit{Z}], p4 \to \textit{FUNCTION}[x], p5 \to equal[omega, domain[z]],
                  p6 \to subclass[range[z], \textit{Z}], p7 \to member[z, map[omega, \textit{Z}]]}] /.
                  z \to composite[INTADD, intersection[composite[inverse[\textit{FIRST}], x],
                  composite[inverse[\textit{SECOND}], y]]] // Reverse

Out[14]= or[member[composite[INTADD, intersection[composite[inverse[\textit{FIRST}], x],
                  composite[inverse[\textit{SECOND}], y]]], map[omega, \textit{Z}]],
                  not[member[x, map[omega, \textit{Z}]], not[member[y, map[omega, \textit{Z}]]]]] = True
\end{verbatim}

\begin{verbatim}
In[15]:= (% /. \{x \to x\_, y \to y\_\}) /. \textit{Equal} \to \textit{SetDelayed}
\end{verbatim}

\begin{verbatim}
\end{verbatim}

\textbf{sums of homomorphisms}

A homomorphism \textit{t} from \textit{NATADD} to \textit{INTADD} is a mapping from \textit{omega} to \textit{Z} that satisfies composite[\textit{t}, NATADD] = composite[INTADD, cross[\textit{t}.\textit{t}]]. In this section it is shown that the sum of two homomorphisms is another. The commutative and associative laws for \textit{INTADD} imply the following twist rule which forms the basis of the derivation in this section.

\begin{verbatim}
In[16]:= composite[INTADD, cross[INTADD, INTADD], TWIST]

Out[16]= composite[INTADD, cross[INTADD, INTADD]]
\end{verbatim}

Lemma.
Lemma.

Theorem. The sum of homomorphisms is an homomorphism. (Remark: The execution time for this derivation was reduced from 93.5 seconds to 7.1 seconds by omitting the following step of the proof: `implies[and[p4, p5], p7]`. This
dramatic reduction in time is presumably due to the fact that \( p_4 \) and \( p_5 \) are both equations, so omitting that step reduces the number of equality substitutions that can be attempted.)

\[
\text{derivative of a distributive law}
\]

All binary homomorphisms from \( \text{NATADD} \) to \( \text{INTADD} \) are of the form \( \text{composite}\left[\text{MIXMUL}, \text{LEFT}\left[ x \right] \right] \). Therefore one can reformulate the theorem of the last section as follows:

\[
\text{Theorem. (A distributive law formulated using variables for the two integers. The uniqueness theorem for homomorphisms is used in this derivation.)}
\]
Using \texttt{reify}, one can quickly eliminate the two integer variables, yielding this variable-free distributive law:

\begin{verbatim}
In[29]:= Map[rotate[inverse[#]] &, SubstTest[reify, x,
            composite[z, intersection[composite[inverse[FIRST], MIXMUL, LEFT[first[x]]],
                composite[inverse[SECOND], MIXMUL, LEFT[second[x]]], z -> INTADD]]
    Out[29]= composite[INTADD, cross[MIXMUL, MIXMUL], TWIST, cross[Id, DUP]] =
            composite[MIXMUL, cross[INTADD, Id]]

In[30]:= composite[INTADD, cross[MIXMUL, MIXMUL], TWIST, cross[Id, DUP]] :=
            composite[MIXMUL, cross[INTADD, Id]]
\end{verbatim}

Note that the right side of this rewrite rule involves \texttt{INTADD}. The other distributive law mentioned in the introduction had \texttt{NATADD} on the right hand side. Both distributive laws have \texttt{INTADD} on the left hand side. The other difference between the two distributive laws is the position of \texttt{DUP}. For one law a natural number argument is duplicated, and for the other one, an integer argument is duplicated.