commutative law of multiplication

Johan G. F. Belinfante
2002 August 28

<< goedel52.p17; << tools.m

:Package Title: goedel52.p17 2002 August 27 at 10:40 p.m.

It is now: 2002 Aug 28 at 18:9

Loading Simplification Rules

TOOLS.M Revised 2002 August 22

weightlimit = 40

summary

In this notebook a distributive law for multiplication is used to derive the commutative law for multiplication. The idea is this: at this stage, we already have a recursion relation for multiplication of natural numbers which followed directly from the definition, but it only works on one side:

\[ \text{natmul}[x, \text{succ}[y]] \]
\[ \text{natadd}[x, \text{natmul}[x, y]] \]
\[ \text{natmul}[	ext{succ}[x], y] \]
\[ \text{natmul}[	ext{succ}[x], y] \]

On the other hand, we have in the meantime used a theorem about powers of commuting factors to derive this distributive law:

\[ \text{natmul}[	ext{natadd}[x, y], z] \]
\[ \text{natadd}[	ext{natmul}[x, z], \text{natmul}[y, z]] \]

The idea is now to set \( y = 1 \) in this distributive law to derive the missing other-side recursion relation. Then left and right multiplication satisfy the same recursion relation and the uniqueness of iteration can be used to show they are equal.

a variant of the distributive law

The first step will be to rewrite the distributive law in another way:
The messy \( \texttt{RIGHT} \) factor can be cleaned up using a combination of associativity and \texttt{VSNormality}.

\[
\text{RIGHT}[\text{union}[x, \text{complement}[^{\texttt{cart}}V, \text{intersection}[^{\texttt{cart}}V, \text{intersection}[y, \text{singleton}[z]]]]]]] \equiv \text{composite}[\text{id}[^{\texttt{cart}}V, \text{intersection}[^{\texttt{cart}}V, \text{intersection}[y, \text{singleton}[z]]], \text{singleton}[x]]], \text{inverse}[\text{FIRST}]]
\]

This mess needs to be cleaned up a little to get the desired recursion relation.

\section*{cleaning up the recursion relation}

The \texttt{GOEDEL} program sometimes recognizes the truth of an assertion even when the corresponding rewrite rule is missing. Here is an example:

\[
equal[\text{composite}[\text{NATMUL}, \text{RIGHT}[x]], \text{composite}[\text{NATMUL}, \text{RIGHT}[x]]]
\]

True

This justifies adding the corresponding rewrite rule:

\[
\text{composite}[\text{NATMUL}, \text{RIGHT}[x]], \text{id}[^{\texttt{omega}}]\) := \text{composite}[\text{NATMUL}, \text{RIGHT}[x]]
\]
We replace \( \text{iterate} \) with \( \text{Equal} \) and rotate twice:

\[
\text{Equal} \left[ \text{composite} \left[ \text{NATMUL}, \text{RIGHT}[x] \right], \text{NATMUL}, \text{RIGHT}[y] \right], \text{composite} \left[ \text{NATMUL}, \text{RIGHT}[y], \text{NATADD}, \text{RIGHT}[x] \right] \text{ / . } x \rightarrow \text{singleton}[0]
\]

We turn this around and make it a rewrite rule:

\[
\text{composite} \left[ \text{NATMUL}, \text{RIGHT}[x], \text{SUCCE} \right] := \text{composite} \left[ \text{NATADD}, \text{RIGHT}[x], \text{NATMUL}, \text{RIGHT}[x] \right]
\]

Finally, the uniqueness theorem for \( \text{iterate} \) is used to derive one version of the commutative law:

\[
\text{SubstTest} [\text{implies}, \text{equal} \left[ \text{composite} [u, w], \text{composite} [w, \text{SUCCE}] \right], \text{equal} \left[ \text{composite} \left[ \text{w}, \text{id} [\text{omega}] \right], \text{iterate} [u, \text{image} [w, \text{singleton}[0]]] \right], \{ u \rightarrow \text{composite} \left[ \text{NATADD}, \text{RIGHT}[x] \right], w \rightarrow \text{composite} \left[ \text{NATMUL}, \text{RIGHT}[x] \right] \}] \text{ / . } \text{NATMUL}, \text{LEFT}[x] = \text{composite} \left[ \text{NATMUL}, \text{RIGHT}[x] \right] = \text{True}
\]

It is unclear how best to orient this rule, so we avoid the issue for now:

\[
\text{equal} \left[ \text{composite} \left[ \text{NATMUL}, \text{LEFT}[x] \right], \text{composite} \left[ \text{NATMUL}, \text{RIGHT}[x] \right] \right] := \text{True}
\]

\section*{SWAP rule}

Other versions of the commutative law can be derived. For example:

\[
\text{SubstTest} [\text{assert}, \forall x, \text{equal} \left[ \text{composite} [x, \text{RIGHT}[x]], \text{composite} [x, \text{LEFT}[x]] \right], z \rightarrow \text{NATMUL}]
\]

\[
\text{True} = \text{equal} [\text{rotate} [\text{NATMUL}], \text{rotate} [\text{composite} [\text{NATMUL}, \text{SWAP}]]]
\]

We replace \text{equal} with \text{Equal} and rotate twice:

\[
\text{Equal} \left[ \text{composite} \left[ \text{NATMUL}, \text{LEFT}[x] \right], \text{composite} \left[ \text{NATMUL}, \text{RIGHT}[x] \right] \right] := \text{True}
\]
Map[rotate[rotate[#]] &, 
    Equal[rotate[NATMUL], rotate[composite[NATMUL, SWAP]]]] // Reverse

composite[NATMUL, SWAP] := NATMUL

This certainly can be added as a rewrite rule

composite[NATMUL, SWAP] := NATMUL

■ yet another version

Yet another version is obtained:

Map[A, ImageComp[NATMUL, SWAP, cart[singleton[x], singleton[y]]]]

natmul[x, y] == natmul[y, x]

This cannot be directly made into a rewrite rule, but we could add it as a fact:

Map[equal[#, natmul[y, x]] &, %]

equal[natmul[x, y], natmul[y, x]] := True

equal[natmul[x_, y_], natmul[y_, x_]] := True

Better yet, we will declare natmul to be Orderless. Then some existing rewrite rules can safely be removed.