monotonicity of addition

Johan G. F. Belinfante
2002 June 19

In introduction

A formula expressing the monotonicity of addition is derived in this notebook for the case that one of the summands is fixed. The argument uses induction, and the relation COMMUTE. The basic commutativity result used is that if $x$ commutes with $y$ and $z$, then $x$ also commutes with $\text{composite}(y, z)$. Because NATADD does not currently satisfy any normality tests, it needs to be sequestered from the action of assert in several steps.

some sethood lemmas

For convenience we introduce a temporary abbreviation:

$r[x_] := \text{restrict}[x, \omega, \omega]$

Since $\omega$ is a set, all such restrictions are sets:

$\text{member}[r[x], V] // \text{AssertTest}$
$\text{member}[\text{composite}[\text{id}[\omega], x, \text{id}[\omega]], V] == \text{True}$

$\text{member}[\text{composite}[\text{id}[\omega], x_, \text{id}[\omega]], V] := \text{True}$

In particular:

$\text{member}[r[S], V]$

$\text{True}$

Various expressions involving NATADD are also sets, but one needs to sequester NATADD from the action of assert to derive these:
We add this as a temporary rule.

A temporary membership rule is needed to make good use of this function.

\[
\text{member}(\text{pair}(u, v), \text{composite}(w, \text{IMAGE}(\text{cross}(x, y)))) / / \text{AssertTest}
\]

\[
\text{member}(\text{pair}(u, v), \text{composite}(w, \text{IMAGE}(\text{cross}(x, y)))) = \\
\text{and}(\text{member}(u, V), \text{member}(v, V), \text{member}(\text{composite}(y, u, \text{inverse}(x)), V), \\
\text{member}(\text{pair}(\text{composite}(y, u, \text{inverse}(x)), v), w))
\]

\[
\text{member}(\text{pair}(u, v), \text{composite}(w, \text{IMAGE}(\text{cross}(x, y)))) := \\
\text{and}(\text{member}(u, V), \text{member}(v, V), \text{member}(\text{composite}(y, u, \text{inverse}(x)), V), \\
\text{member}(\text{pair}(\text{composite}(y, u, \text{inverse}(x)), v), w))
\]

\section{a membership rule}

For convenience we introduce the following function:

\[
\lambda x, \: \text{composite}(n, \text{RIGHT}(x)) / . \: n -> \text{NATADD}
\]

\[
\text{VERTSECT}(\text{composite}(\text{SWAP}, \text{inverse}(\text{rotate}(\text{NATADD}))))
\]

A temporary membership rule is needed to make good use of this function.

\[
\text{member}(\text{pair}(u, v), \text{composite}(w, \text{IMAGE}(\text{cross}(x, y)))) / / \text{AssertTest}
\]

\[
\text{member}(\text{pair}(u, v), \text{composite}(w, \text{IMAGE}(\text{cross}(x, y)))) = \\
\text{and}(\text{member}(u, V), \text{member}(v, V), \text{member}(\text{composite}(y, u, \text{inverse}(x)), V), \\
\text{member}(\text{pair}(\text{composite}(y, u, \text{inverse}(x)), v), w))
\]

\section{an induction argument}

The induction step will be essentially this argument:

\[
\text{SubstTest(implies, and[commute[w, y], commute[w, z]], commute[w, composite[y, z]],} \\
\{w -> x[S], y -> z[SUCC], z -> composite[\text{NATADD}, \text{RIGHT}(x)]\})
\]

\[
\text{or[equal[composite[id[omega]], inverse[\text{IMAGE}[\text{inverse}[\text{NATADD}]])],} \\
\text{inverse[\text{IMAGE}[\text{id[cart[V, singleton(x)]]]]]}, \text{inverse[\text{IMAGE}[\text{FIRST}]])}, \ E,} \\
\text{composite[\text{SUCC}, \text{NATADD}, \text{RIGHT}(x), \text{id[omega]}, S, \text{id[omega]}])}, \\
\text{not[equal[composite[id[omega]], S, \text{NATADD}, \text{RIGHT}(x)]],} \\
\text{composite[\text{NATADD}, \text{RIGHT}(x), \text{id[omega]}, S, \text{id[omega]}])}] = \text{True}
\]

We add this as a temporary rule.
The result looks better if one reintroduces a variable:

The induction theorem can now be applied:

Our next step is to eliminate the variable \( x \) in the above statement:

We add this as a temporary rule:

The induction theorem now can be applied:

The result looks better if one reintroduces a variable:
### final steps

The final steps are designed to remove the condition that \( x \) be a member of \( \omega \). The basic idea is that if this is not the case, then both sides of the equation reduce to 0. We begin with a simplification rule:

\[
\text{subclass}[\omega, 0] \quad \text{// AssertTest}
\]

\[
\text{subclass}[\omega, 0] == \text{False}
\]

\[
\text{subclass}[\omega, 0] := \text{False}
\]

With this rule in place, we have:

\[
\text{equal}[0, \text{composite}[\text{NATADD}, \text{RIGHT}[x]]]
\]

\[
\text{not}[\text{member}[x, \omega]]
\]

We now add a temporary rule:

\[
\text{SubstTest}[\text{implies, and}[\text{equal}[x, y], \text{subclass}[y, z]], \text{member}[x, z],
\{y \rightarrow \omega, 
z \rightarrow \text{image}[\text{inverse}[\text{VERTSECT}[\text{composite}[\text{SWAP}, \text{inverse}[\text{rotate}[\text{NATADD}]]],
\text{fix}[\text{composite}[\text{inverse}[\text{IMAGE}[\text{cross}[\text{Id}, \text{composite}[\text{id}[\omega], S, \text{id}[\omega]]]]],
\text{IMAGE}[\text{cross}[\text{composite}[\text{id}[\omega], \text{inverse}[S], \text{id}[\omega]], \text{Id}]]]]]]) \quad \text{// MapNotNot}
\]

\[
\text{or}[\text{equal}[\text{composite}[\text{id}[\omega]], \text{S}, \text{NATADD}, \text{RIGHT}[x]],
\text{composite}[\text{NATADD}, \text{RIGHT}[x], \text{id}[\omega], \text{S}, \text{id}[\omega]]], \text{not}[\text{member}[x, \omega]]] == \text{True}
\]

We add this as a temporary rule, but it will be replaced later with a better rule.

\[
\text{or}[\text{equal}[\text{composite}[\text{id}[\omega]], \text{S}, \text{NATADD}, \text{RIGHT}[x]],
\text{composite}[\text{NATADD}, \text{RIGHT}[x, \omega], \text{id}[\omega], \text{S}, \text{id}[\omega]]], \text{not}[\text{member}[x, \omega]]] := \text{True}
\]

We just need to do a bit of reasoning to complete the derivation
It is not entirely clear how to orient this equation, but composites with functions behave better with respect to complementation when the function is on the right. For this reason, it seems desirable to orient the rewrite rule this way:

```plaintext
composite[NATADD, RIGHT[x_], id[omega], S, id[omega]] :=
composite[id[omega], S, NATADD, RIGHT[x]]
```