addition and subtraction

Johan G. F. Belinfante  
2002 October 31

For natural numbers, unlike for integers, addition and subtraction do not quite commute. For example, \((0 + 1) - 1\) is not the same as \((0 - 1) + 1\) because the latter involves the term \(0 - 1\) which is not a natural number. The functors \textbf{natadd} and \textbf{natsub} are defined to be equal to \(V\) when applied to arguments that do not produce natural numbers. A formula is derived in this notebook which expresses the fact that addition and subtraction of natural numbers do commute when the result is a natural number, and takes care of the unnatural cases as well. A variable-free version of this law is also derived. The proposed orientation of these formulas as rewrite rules is tentative.

\begin{itemize}
  \item \textbf{a lemma}
  \end{itemize}

To simplify the final formulas, a temporary lemma is useful:

\[
\text{image}[V, \text{intersection}[	ext{complement}[	ext{natadd}[x, z]], \text{natadd}[y, z]]] // \text{Normality}
\]

\[
\text{image}[V, \text{intersection}[	ext{complement}[	ext{natadd}[x, z]], \text{natadd}[y, z]]] == \\
\text{union}\left[\text{intersection}[	ext{complement}[	ext{image}[V, \text{intersection}[\text{omega}, \text{singleton}[y]]], \text{image}[V, \text{intersection}[\text{omega}, \text{singleton}[x]]], \text{intersection}[	ext{image}[V, \text{intersection}[\text{omega}, \text{singleton}[z]]], \text{image}[V, \text{intersection}[\text{omega}, \text{singleton}[x]]], \text{image}[V, \text{intersection}[\text{omega}, \text{singleton}[z]]], \text{image}[V, \text{intersection}[\text{omega}, \text{singleton}[z]]], \text{image}[V, \text{intersection}[y, \text{complement}[x]]]]]\right]
\]

\[
\text{image}[V, \text{intersection}[	ext{complement}[	ext{natadd}[x, z]], \text{natadd}[y, z]]] == \\
\text{union}\left[\text{intersection}[	ext{complement}[	ext{image}[V, \text{intersection}[\text{omega}, \text{singleton}[y]]], \text{image}[V, \text{intersection}[\text{omega}, \text{singleton}[x]]], \text{image}[V, \text{intersection}[\text{omega}, \text{singleton}[z]]], \text{image}[V, \text{intersection}[\text{omega}, \text{singleton}[x]]], \text{image}[V, \text{intersection}[\text{omega}, \text{singleton}[z]]], \text{image}[V, \text{intersection}[y, \text{complement}[x]]]]\right]
\]

\begin{itemize}
  \item \textbf{the basic result}
  \end{itemize}

The main formula can be derived in a single step:
Map[A[Singleton[#]] & , SubstTest[natadd, natsub[u, v], natsub[v, w],
{u -> natadd[y, x], v -> natadd[z, x], w -> z}]]

natadd[x, natsub[y, z]] ==
union[Image[V, Intersection[z, complement[y]]], natsub[natadd[x, y], z]]

The following orientation of this formula as a rewrite rule is tentative:

natadd[x_, natsub[y_, z_]] :=
union[Image[V, Intersection[z, complement[y]]], natsub[natadd[x, y], z]]

■ a variable–freeversion

A quick way to derive a variable–freeversion of this result uses symdif and SubstTest.

symdif[composite[rotate[natadd], cross[natadd, Id], id[composite[inverse[S], second]],
composite[natadd, cross[Id, rotate[natadd]], Assoc]], //ＶＳTerNormality

union[composite[intersection[composite[complement[rotate[natadd]]], cross[natadd, Id]],
composite[natadd, cross[Id, rotate[natadd]], Assoc]],
id[composite[inverse[S], second]],
composite[intersection[composite[complement[rotate[natadd]]], cross[natadd, Id]],
composite[complement[natadd], cross[Id, rotate[natadd]], Assoc]],
id[composite[inverse[S], second]]] == 0

union[composite[intersection[composite[complement[rotate[natadd]]], cross[natadd, Id]],
composite[natadd, cross[Id, rotate[natadd]], Assoc]],
id[composite[inverse[S], second]],
composite[intersection[composite[complement[rotate[natadd]]], cross[natadd, Id]],
composite[complement[natadd], cross[Id, rotate[natadd]], Assoc]],
id[composite[inverse[S], second]]] == 0

It is not clear how to orient the following formula as a rewrite rule, but we tentatively opt to orient it the same way as the formula for variables.

SubstTest[equal, 0, symdif[u, v],
{u ->
composite[rotate[natadd], cross[natadd, Id], id[composite[inverse[S], second]]],
v -> composite[natadd, cross[Id, rotate[natadd]], Assoc]]

True == equal[composite[natadd, cross[Id, rotate[natadd]], Assoc],
composite[rotate[natadd], cross[natadd, Id], id[composite[inverse[S], second]]]]

Before making this into a rewrite rule it is useful to move the ASSOC to the other side.

Map[composite[#, inverse[ASSOC]] & ,
composite[natadd, cross[Id, rotate[natadd]], Assoc] ==
composite[rotate[natadd], cross[Id, rotate[natadd]], id[composite[inverse[S], second]]]]

composite[natadd, cross[Id, rotate[natadd]]] == composite[rotate[natadd],
cross[natadd, Id], id[composite[inverse[S], second]], inverse[ASSOC]]

composite[natadd, cross[Id, rotate[natadd]]] == composite[rotate[natadd],
cross[natadd, Id], id[composite[inverse[S], second]], inverse[ASSOC]]

A flipped version of this can be derived as well:
Assoc[NATADD, cross[Id, rotate[NATADD]], SWAP]

composite[NATADD, cross[rotate[NATADD], Id]] ==
  composite[rotate[NATADD], cross[NATADD, Id], id[composite[inverse[S], SECOND]], ROT]

composite[NATADD, cross[rotate[NATADD], Id]] :=
  composite[rotate[NATADD], cross[NATADD, Id], id[composite[inverse[S], SECOND]], ROT]