left subtraction

Johan G. F. Belinfante  
2002 September 12

Subtraction is obtained from addition by rotation. For example, the fact \(3 - 1 = 2\) is obtained from \(1 + 2 = 3\) by rotating the three numbers:

\[
\begin{align*}
\text{image[rotate[NATADD],} \\
\text{cart[singleton[succ[succ[singleton[0]]]]], singleton[singleton[0]]]}  \\
\text{] // Normality}
\end{align*}
\]

\[
\text{image[image[inverse[NATADD], singleton[succ[succ[singleton[0]]]]]],} \\
\text{singleton[singleton[0]]] == singleton[succ[succ[singleton[0]]]]}
\]

Since addition is commutative, adding on the left is the same as adding on the right:

\[
\begin{align*}
\text{composite[NATADD, LEFT[x]]} \\
\text{composite[NATADD, RIGHT[x]]}
\end{align*}
\]

On the other hand, subtraction is not commutative, and so the process of subtracting a fixed number \(x\) differs from the process of subtracting from a fixed number \(x\). Subtracting \(x\) is just the inverse of adding \(x\).

\[
\begin{align*}
\text{composite[rotate[NATADD], RIGHT[x]]} \\
\text{composite[inverse[RIGHT[x]], inverse[NATADD]]}
\end{align*}
\]

The process of subtracting from \(x\) is an involution, that is, this process is its own inverse:

\[
\begin{align*}
\text{composite[rotate[NATADD], LEFT[x]]} \\
\text{image[inverse[NATADD], singleton[x]]} \\
\text{inverse[image[inverse[NATADD], singleton[x]]]} \\
\text{image[inverse[NATADD], singleton[x]]}
\end{align*}
\]

Both of these processes are one-to-one functions:
The goal in this notebook is to derive a formula for the domain and range of left-subtraction, \( \text{composite} \{ \\text{rotate} \{ \\text{NATADD} \}, \text{LEFT} [x] \} \).

**a formula for the successor of a natural number**

We will need as a prerequisite a formula that says that the successor of a natural number \( x \) is precisely the set of all natural numbers contained in \( x \). To derive this result, we begin with this observation:

\[
\begin{align*}
\text{equal}[0, \text{composite}[\text{id}[\omega]], \\
\text{intersection}[\text{composite}[\text{complement}[\text{inverse}[E]], \text{SUCC}], \text{inverse}[S]], \text{id}[\omega]]
\end{align*}
\]

\[
\text{True}
\]

While the GOEDEL program recognizes the truth of this assertion, it lacks the corresponding rewrite rule, which we now add on a temporary basis:

\[
\begin{align*}
\text{composite}[\text{id}[\omega]], \\
\text{intersection}[\text{composite}[\text{complement}[\text{inverse}[E]], \text{SUCC}], \text{inverse}[S]], \text{id}[\omega]] := 0
\end{align*}
\]

When the ImageComp test is performed using this fact, one encounters the following expression which we simplify using Renormality:

\[
\begin{align*}
\text{fix}[\text{composite}[\text{complement}[\text{inverse}[E]], \text{SUCC}, \\
\text{id}[\text{intersection}[\omega, \text{singleton}[x]]]], S] & \text{ // Renormality} \\
\text{fix}[ \\
\text{composite}[\text{complement}[\text{inverse}[E]], \text{SUCC}, \text{id}[\text{intersection}[\omega, \text{singleton}[x]]]], S] & = \\
\text{intersection}[\text{complement}[x], \text{complement}[\text{singleton}[x]]], \text{image}[V, \text{intersection}[\omega, \text{singleton}[x]]], \text{P}[x]]
\end{align*}
\]

\[
\begin{align*}
\text{fix}[\text{composite}[\text{complement}[\text{inverse}[E]], \text{SUCC}, \\
\text{id}[\text{intersection}[\omega, \text{singleton}[x]]]], S] & := \text{intersection}[\text{complement}[x], \\
\text{complement}[\text{singleton}[x]], \text{image}[V, \text{intersection}[\omega, \text{singleton}[x]]]], \text{P}[x]]
\end{align*}
\]

Now the ImageComp test is performed, yielding almost what we want:

\[
\begin{align*}
\text{Map}[\text{equal}[0, #] & \&, \text{ImageComp}[\text{id}[\omega]], \\
\text{composite}[\text{intersection}[\text{composite}[\text{complement}[\text{inverse}[E]], \text{SUCC}], \text{inverse}[S]], \\
\text{id}[\omega]], \text{singleton}[x]] & \text{ // Reverse} \\
\text{or}[\text{not}[\text{member}[x, \omega]], \text{subclass}[\text{intersection}[\omega, \text{P}[x]], \text{succ}[x]]] & = \text{True} \\
\text{or}[\text{not}[\text{member}[x_-, \omega]], \text{subclass}[\text{intersection}[\omega, \text{P}[x_-]], \text{succ}[x_-]]] & = \text{True}
\end{align*}
\]

We can derive a stronger result, replacing the inclusion with equality. This is obtained by combining AssertTest with double negation.
Map[nот, Map[nот[implies[member[x, omega], #]] &
equal[intersection[omega, P[x]], succ[x]] // AssertTest]]

or[equal[intersection[omega, P[x]], succ[x]], not[member[x, omega]]] == True

or[equal[intersection[omega, P[x_]], succ[x_]], not[member[x_, omega]]] := True

The following corollary is noted:

equal[intersection[omega, image[V, intersection[omega, singleton[x]]]], P[x]],
intersection[image[V, intersection[omega, singleton[x]]], succ[x]]

True

This fact justifies adding the following new rewrite rule:

intersection[omega, image[V, intersection[omega, singleton[x_]]], P[x_]] :=
intersection[image[V, intersection[omega, singleton[x]]], succ[x]]

■ application to left−subtraction

The rewrite rule derived in the preceding section is used here to get a simple formula for the range of left−subtraction function:

SubstTest[image, rotate[w], cart[V, singleton[x]], w -> rotate[NATADD]]

range[image[inverse[NATADD], singleton[x]]] ==
intersection[image[V, intersection[omega, singleton[x]]], succ[x]]

range[image[inverse[NATADD], singleton[x_]]] :=
intersection[image[V, intersection[omega, singleton[x]]], succ[x]]

The domain is the same:

SubstTest[image, inverse[w], V, w -> image[inverse[NATADD], singleton[x]]] // Reverse

domain[image[inverse[NATADD], singleton[x]]] ==
intersection[image[V, intersection[omega, singleton[x]]], succ[x]]

domain[image[inverse[NATADD], singleton[x_]]] :=
intersection[image[V, intersection[omega, singleton[x]]], succ[x]]

More generally, one has the following, but it is unclear how to orient this more general formula.

SubstTest[image, inverse[w], V, w -> image[inverse[NATADD], x]] // Reverse

domain[image[inverse[NATADD], x]] == range[image[inverse[NATADD], x]]

The following involution property does not require adding any new rules:

composite[image[inverse[NATADD], singleton[x]], image[inverse[NATADD], singleton[x]]]

id[intersection[image[V, intersection[omega, singleton[x]]], succ[x]]]
■ serendipity: variable−free formulation of the involution property

The following formula was discovered accidentally:

\[
\text{Assoc}[\text{DUP}, \text{inverse}[\text{DUP}], \text{union}[\text{composite}[\text{DUP}, \text{id}[\omega]], \text{composite}[\text{inverse}[\text{E}], \text{IMAGE}[\text{DUP}], \text{id}[\omega]]]]] // \text{Reverse}
\]

\[
\text{union}[\text{composite}[\text{DUP}, \text{id}[\omega]], \text{composite}[\text{inverse}[\text{E}], \text{IMAGE}[\text{DUP}], \text{id}[\omega]]] ==
\text{composite}[\text{DUP}, \text{id}[\omega], \text{inverse}[\text{S}], \text{id}[\omega]]
\]

\[
\text{union}[\text{composite}[\text{DUP}, \text{id}[\omega]], \text{composite}[\text{inverse}[\text{E}], \text{IMAGE}[\text{DUP}], \text{id}[\omega]]] :=
\text{composite}[\text{DUP}, \text{id}[\omega], \text{inverse}[\text{S}], \text{id}[\omega]]
\]

The following variable−free formulation of the involution property follows as a corollary of this discovery:

\[
\text{Map}[\text{inverse}, \text{composite}[\text{RIF}, \text{cross}[\text{inverse}[\text{NATADD}], \text{inverse}[\text{NATADD}]]], \text{DUP}] // \text{VSNormality}
\]

\[
\text{composite}[\text{intersection}[\text{composite}[\text{NATADD}, \text{FIRST}], \text{composite}[\text{NATADD}, \text{SECOND}]], \text{inverse}[\text{RIF}]] == \text{composite}[\text{id}[\omega], \text{S}, \text{id}[\omega], \text{inverse}[\text{DUP}]]
\]

\[
\text{composite}[\text{intersection}[\text{composite}[\text{NATADD}, \text{FIRST}], \text{composite}[\text{NATADD}, \text{SECOND}]], \text{inverse}[\text{RIF}]] := \text{composite}[\text{id}[\omega], \text{S}, \text{id}[\omega], \text{inverse}[\text{DUP}]]
\]