rules for calculating natmod[x,y]

Johan G. F. Belinfante
2005 June 24

This notebook develops conditional rewrite rules for computing \texttt{natmod[x,y]} when \(x\) and \(y\) are natural numbers. For the special case \(y = 0\), one has \(\texttt{natmod[x,y]} = x\) by an existing rule. If \(x\) is less than \(y\), then \(\texttt{natmod[x,y]} = x\). Otherwise \(\texttt{natmod[x,y]}\) is unchanged if one replaces \(x\) with \(\texttt{natsub[x,y]}\).

the case \(x < y\)

In this section a rewrite rule is derived for computing \(\texttt{natmod[x,y]}\) when \(x < y\). It is convenient to first use \texttt{nat} wrappers to derive the main result, and later to remove the wrappers. The following lemma is needed.

\[
\text{Map}[	ext{implies}[, \text{equal}[	ext{nat}[x], \text{natmod}[\text{nat}[x], \text{nat}[y]]]] \&, \\
\text{SubstTest}[	ext{and}, \text{member}[	ext{pair}[	ext{nat}[y], w], \text{DIV}], \\
\text{member}[w, \text{nat}[y]], w \rightarrow \text{natsub}[\text{nat}[x], \text{natmod}[\text{nat}[x], \text{nat}[y]]]]]
\]

\[
\text{Out[2]} = \text{or}[	ext{equal}[	ext{nat}[x], \text{natmod}[\text{nat}[x], \text{nat}[y]]], \\
\text{not}[	ext{member}[	ext{natsub}[\text{nat}[x], \text{natmod}[\text{nat}[x], \text{nat}[y]]], \text{nat}[y]]]] = \text{True}
\]

\[
\text{In[3]} = \text{(x \rightarrow x, y \rightarrow y)} / \text{Equal} \rightarrow \text{SetDelayed}
\]

Transposition lemma:
Main theorem.

The \texttt{nat} wrappers can be removed using equality substitution.

Restatement:

This justifies the following conditional rewrite rule:

For the case $x \geq y$ it will be shown that $\texttt{natmod}[x, y]$ is equal to $\texttt{natmod}[\texttt{natsub}[x,y], y]$. This result will be derived by first showing that $\texttt{natmod}[x, y]$ is equal to $\texttt{natmod}[\texttt{natadd}[x,y], y]$, and then replacing $x$ with $\texttt{natsub}[x,y]$. 
an inclusion

Lemma.

```
In[12]:= SubstTest[{member, pair[v, natadd[v, u]], DIV,
    {u -> natsub[nat[x], natmod[nat[x], nat[y]]], v -> nat[y]}]
Out[12]= member[pair[nat[y],
    natsub[natadd[nat[x], nat[y]], natmod[nat[x], nat[y]]]], DIV] = True
```

The following inclusion follows:

```
In[14]:= SubstTest[implies, member[pair[v, natsub[u, w]], DIV],
    subclass[natmod[u, v], w],
    {u -> natadd[nat[x], nat[y]], v -> nat[y], w -> natmod[nat[x], nat[y]]}]
Out[14]= subclass[natmod[natadd[nat[x], nat[y]], nat[y]], nat[y],
    natmod[nat[x], nat[y]]] = True
```

The next task will be to show that the reverse inclusion also holds.

inclusion in the opposite direction

Lemma. (Transposition relates subtraction to addition.)

```
In[16]:= SubstTest[{subclass, nat[y],
    natsub[nat[x], nat[w]], w -> natmod[nat[x], nat[y]]}
Out[16]= subclass[nat[y], natsub[nat[x], natmod[nat[x], nat[y]]]] =
    not[member[nat[x], natadd[nat[y], natmod[nat[x], nat[y]]]]]
```

```
In[17]:= subclass[nat[y_], natsub[nat[x_], natmod[nat[x_], nat[y_]]]] :=
    not[member[nat[x], natadd[nat[y], natmod[nat[x], nat[y]]]]]
```

The difference \( x - y \) is divisible by \( y \) if \( x \) is divisible by \( y \).

```
In[18]:= SubstTest[{member, pair[v, natadd[v, u]],
    DIV, {u -> natsub[nat[x, y], v -> y]} // Reverse}
Out[18]= member[pair[y, natsub[x, y]], DIV] =
    and[member[pair[y, x], DIV], subclass[y, x]]
```
Application.

The number $\text{natmod}[u,v]$ is the least remainder when $v$ is divided into $u$.

By dichotomy, weak inequalities can be replaced with negated strict inequalities in the opposite direction.

By the transitive law, one finds:

The desired inclusion is obtained:
The new inclusion is combined with the result of the preceding section to yield an equation.

The \texttt{natsub} rule is obtained from the \texttt{natadd} rule of the preceding section by replacing \texttt{x} with \texttt{x - y}.

This can be cleaned up as follows:
The \texttt{nat} wrappers can be replaced with numberhood literals:

\begin{verbatim}
In[35]:= \texttt{or[equal[natmod[nat[x_]], nat[y_]]},
    natmod[natsub[nat[x_], nat[y_]], nat[y_]],
    member[nat[x_], nat[y_]]] := True
\end{verbatim}

This justifies the following conditional rewrite rule:

\begin{verbatim}
In[39]:= natmod[x_, y_] := natmod[natsub[x, y], y];
    and[member[x, omega], member[y, omega], not[member[x, y]]]
\end{verbatim}

Together, the two conditional rules allow one to recursively compute \texttt{natmod[x,y]} for any pair of natural numbers, except when \(y = 0\), in which case a separate rule applies.