nat[x] trichotomy rules

Johan G. F. Belinfante
2005 April 26

```
In[1]:= SetDirectory["i:"]; << goedel68.23a; << tools.m

        :Package Title: goedel68.23a         2005 April 23 at 1:00 p.m.

        It is now:  2005 Apr 26 at 14:52

        Loading Simplification Rules

        TOOLS.M                       Revised 2005 April 16

        weightlimit = 40
```

summary

Using natural number wrappers, the statement for trichotomy is this:

```
In[2]:= or[equal[nat[x], nat[y]], member[nat[x], nat[y]], member[nat[y], nat[x]]]

Out[2]= True
```

Rewrite rules involving the natural wrapper **nat** are derived that from the trichotomy rule for natural number order that permit any disjunction or conjunction of any pair of the three literals that appear in the trichotomy rule, negated or not, to be rewritten as a single literal. In all but two rules only one of the two variables needs to be wrapped with **nat**.

consequences of trichotomy involving one wrapped variable

Any two of the three alternatives cannot hold simultaneously. One such rule is available, but another is needed:.

```
In[3]:= and[equal[x, nat[y]], member[x, nat[y]]] // NotNotTest

Out[3]= and[equal[x, nat[y]], member[x, nat[y]]] == False

In[4]:= and[equal[x_, nat[y_]], member[x_, nat[y_]]] := False
```

Strict order implies inequality.

```
In[5]:= equiv[and[member[x, nat[y]], not[equal[x, nat[y]]]],
           member[x, nat[y]]] // not // not

Out[5]= True

In[6]:= and[member[x_, nat[y_]], not[equal[x_, nat[y_]]]] := member[x, nat[y]]
```

Corollary.

```
In[7]:= or[equal[x, nat[y]], not[member[x, nat[y]]]] // NotNotTest

Out[7]= or[equal[x, nat[y]], not[member[x, nat[y]]]] == not[member[x, nat[y]]]

In[8]:= or[equal[x_, nat[y_]], not[member[x_, nat[y_]]]] := not[member[x, nat[y]]]
```

Equality implies that no strict order holds.

```
In[9]:= equiv[and[equal[x, nat[y]], not[member[x, nat[y]]]], equal[x, nat[y]]]

Out[9]= True

In[10]:= and[equal[x_, nat[y_]], not[member[x_, nat[y_]]]] := equal[x, nat[y]]
```

Corollary.

```
In[11]:= or[member[x, nat[y]], not[equal[x, nat[y]]]] // NotNotTest

Out[11]= or[member[x, nat[y]], not[equal[x, nat[y]]]] == not[equal[x, nat[y]]]

In[12]:= or[member[x_, nat[y_]], not[equal[x_, nat[y_]]]] := not[equal[x, nat[y]]]
```

Strict order implies that the reverse order does not hold.

```
In[13]:= equiv[and[member[x, nat[y]], not[member[nat[y], x]]], member[x, nat[y]]]

Out[13]= True

In[14]:= and[member[x_, nat[y_]], not[member[nat[y_], x_]]] := member[x, nat[y]]
```

Corollary.

```
In[15]:= or[member[nat[x], y], not[member[y, nat[x]]]] // NotNotTest

Out[15]= or[member[nat[x], y], not[member[y, nat[x]]]] == not[member[y, nat[x]]]

In[16]:= or[member[nat[x_], y_], not[member[y_, nat[x_]]]] := not[member[y, nat[x]]]
```

rules with two wrapped variables

The negation of a weak inequality is the reverse of a strong one:

```
In[17]:= equiv[and[not[equal[nat[x], nat[y]]], not[member[nat[x], nat[y]]]],
           member[nat[y], nat[x]]] // not // not

Out[17]= True

In[18]:= and[not[equal[nat[x_], nat[y_]]], not[member[nat[x_], nat[y_]]]] :=
          member[nat[y], nat[x]]
```

Corollary:

```
In[19]:= or[equal[nat[x], nat[y]], member[nat[x], nat[y]]] // NotNotTest

Out[19]= or[equal[nat[x], nat[y]], member[nat[x], nat[y]]] ==
          not[member[nat[y], nat[x]]]

In[20]:= or[equal[nat[x_], nat[y_]], member[nat[x_], nat[y_]]] :=
          not[member[nat[y], nat[x]]]
```