NATEXP, part 3

Johan G. F. Belinfante
2006 June 29

```
In[1]:= SetDirectory["l:"]; << goedel82.29a; << tools.m

        :Package Title: goedel82.29a         2006 June 29 at 2:00 a.m.

        It is now:  2006 Jun 29 at 10:4

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 June 27

        weightlimit = 40
```

summary

This is the third notebook in a series about the exponential function for natural numbers. Corollaries of the laws of exponents are derived. The main idea here is to use **reify** to eliminate one or more variables from the three laws of exponents derived in part 2 of this series. The first two laws of exponents can be viewed as two distinct distributive laws, and the third as a quasi-associative law. The choice of orientation for the rewrite rules derived in this notebook is in most cases guided by analogy with other distributive and associative laws already in the **GOEDEL** program.

some simplification rules

Several simplification rules needed in later sections are gathered together in this preliminary section. The first one is analogous to a similar one derived previously with **LEFT[x]** in place of **RIGHT[x]**. The **GOEDEL** program is able to recognize the truth of the following statement, but lacks a corresponding rewrite rule. So a new rewrite rule is added.

```
In[3]:= equal[composite[NATEXP, RIGHT[x], id[omega]], composite[NATEXP, RIGHT[x]]]

Out[3]= True

In[4]:= composite[NATEXP, RIGHT[x_], id[omega]] := composite[NATEXP, RIGHT[x]]
```

A similar situation holds for the next equation; its truth is recognized but a rewrite rule is lacking, so it is added now.

```
In[5]:= equal[composite[id[image[V, intersection[omega, set[x]]]], NATEXP, LEFT[x]],
         composite[NATEXP, LEFT[x]]]

Out[5]= True

In[6]:= composite[id[image[V, intersection[omega, set[x_]]]], NATEXP, LEFT[x_]] :=
         composite[NATEXP, LEFT[x]]
```

A similar result holds with **RIGHT[x]** in place of **LEFT[x]**.

```
In[7]:= equal[composite[id[image[V, intersection[omega, set[x]]]], NATEXP, RIGHT[x]],
          composite[NATEXP, RIGHT[x]]]

Out[7]= True

In[8]:= composite[id[image[V, intersection[omega, set[x_]]]], NATEXP, RIGHT[x_]] :=
          composite[NATEXP, RIGHT[x]]
```

reifying the first law of exponents

A lemma is needed that is just a variant of the last rewrite rule derived in the preceding section.

```
In[11]:= SubstTest[composite, id[image[V, intersection[omega, set[w]]]],
           NATEXP, RIGHT[w], w → natadd[y, z]]

Out[11]= composite[id[intersection[
             image[V, intersection[omega, set[y]]], image[V, intersection[omega, set[z]]]]],
           NATEXP, RIGHT[natadd[y, z]]] == composite[NATEXP, RIGHT[natadd[y, z]]]

In[12]:= composite[id[intersection[image[V, intersection[omega, set[y_]]],
             image[V, intersection[omega, set[z_]]]], NATEXP,
           RIGHT[natadd[y_, z_]]] := composite[NATEXP, RIGHT[natadd[y, z]]]
```

Applying **reify** to the first law of exponents yields:

```
In[29]:= Map[equal[VERTSECT[#],
            composite[NATMUL, intersection[composite[inverse[FIRST], NATEXP, RIGHT[x]],
              composite[inverse[SECOND], NATEXP, RIGHT[y]]]]] &,
          SubstTest[reify, z, natexp[z, f[x, y]], f → natadd]]

Out[29]= True == equal[composite[NATEXP, RIGHT[natadd[x, y]]],
           composite[NATMUL, intersection[composite[inverse[FIRST], NATEXP, RIGHT[x]],
             composite[inverse[SECOND], NATEXP, RIGHT[y]]]]]

In[31]:= composite[NATMUL, intersection[composite[inverse[FIRST], NATEXP, RIGHT[x_]],
            composite[inverse[SECOND], NATEXP, RIGHT[y_]]]] :=
          composite[NATEXP, RIGHT[natadd[x, y]]]
```

The above rule was oriented by analogy with the following rewrite rule already in the **GOEDEL** program:

```
In[32]:= composite[NATADD, intersection[
           composite[inverse[FIRST], times[x]], composite[inverse[SECOND], times[y]]]]

Out[32]= times[natadd[x, y]]
```

A second application of **reify** eliminates both of the remaining variables, producing a variable-free formulation of the first law of exponents:

```
In[43]:= Map[flip[rotate[inverse[#]]] &, SubstTest[reify, x,
           composite[w, intersection[composite[inverse[FIRST], NATEXP, RIGHT[first[x]]],
            composite[inverse[SECOND], NATEXP, RIGHT[second[x]]]]], w → NATMUL]] // Reverse

Out[43]= composite[NATMUL, cross[NATEXP, NATEXP], TWIST, cross[DUP, Id]] ==
          composite[NATEXP, cross[Id, NATADD]]

In[44]:= composite[NATMUL, cross[NATEXP, NATEXP], TWIST, cross[DUP, Id]] :=
          composite[NATEXP, cross[Id, NATADD]]
```

This rewrite rule is completely analogous to the following existing rule:

```
In[42]:= composite[NATADD, cross[NATMUL, NATMUL], TWIST, cross[DUP, Id]]

Out[42]= composite[NATMUL, cross[Id, NATADD]]
```

corollary of the first law of exponents

A corollary of the first law of exponents was derived in part 2 of this series, and was used to derive the third law of exponents. One can use **reify** to eliminate one of the variables in this corollary.

```
In[46]:= Map[rotate[inverse[#]] &,
          SubstTest[reify, y, composite[times[natexp[x, y]], z, LEFT[x]], z → NATEXP]]

Out[46]= composite[NATEXP, LEFT[x], NATADD] ==
          composite[NATMUL, cross[composite[NATEXP, LEFT[x]], composite[NATEXP, LEFT[x]]]]

In[47]:= composite[NATEXP, LEFT[x_], NATADD] :=
          composite[NATMUL, cross[composite[NATEXP, LEFT[x]], composite[NATEXP, LEFT[x]]]]
```

Again, the orientation of this rewrite rule was chosen by analogy with an existing rule:

```
In[48]:= composite[NATMUL, LEFT[x], NATADD]

Out[48]= composite[NATADD, cross[times[x], times[x]]]
```

reifying the second law of exponents

Lemma.

```
In[50]:= SubstTest[composite, id[image[V, intersection[omega, set[w]]]],
          NATEXP, LEFT[w], w → natmul[x, y]]

Out[50]= composite[id[intersection[
             image[V, intersection[omega, set[x]]], image[V, intersection[omega, set[y]]]]],
           NATEXP, LEFT[natmul[x, y]]] == composite[NATEXP, LEFT[natmul[x, y]]]
```

```
In[51]:= composite[id[intersection[image[V, intersection[omega, set[x_]]],
             image[V, intersection[omega, set[y_]]]]], NATEXP,
           LEFT[natmul[x_, y_]]] := composite[NATEXP, LEFT[natmul[x, y]]]
```

One can use **reify** to remove the variable **z**.

```
In[54]:= Map[equal[VERTSECT[#],
            composite[NATMUL, intersection[composite[inverse[FIRST], NATEXP, LEFT[x]],
              composite[inverse[SECOND], NATEXP, LEFT[y]]]]] &,
          SubstTest[reify, z, natexp[f[x, y], z], f → natmul]]
```

```
Out[54]= True == equal[composite[NATEXP, LEFT[natmul[x, y]]],
           composite[NATMUL, intersection[composite[inverse[FIRST], NATEXP, LEFT[x]],
             composite[inverse[SECOND], NATEXP, LEFT[y]]]]]
```

```
In[56]:= composite[NATMUL, intersection[composite[inverse[FIRST], NATEXP, LEFT[x_]],
            composite[inverse[SECOND], NATEXP, LEFT[y_]]]] :=
          composite[NATEXP, LEFT[natmul[x, y]]]
```

The remaining two variable are removed using a second application of **reify**.

```
In[57]:= Map[rotate[inverse[#]] &, SubstTest[reify, x,
            composite[w, intersection[composite[inverse[FIRST], NATEXP, LEFT[first[x]]],
              composite[inverse[SECOND], NATEXP, LEFT[second[x]]]]], w → NATMUL]] // Reverse
```

```
Out[57]= composite[NATMUL, cross[NATEXP, NATEXP], TWIST, cross[Id, DUP]] ==
          composite[NATEXP, cross[NATMUL, Id]]
```

```
In[58]:= composite[NATMUL, cross[NATEXP, NATEXP], TWIST, cross[Id, DUP]] :=
          composite[NATEXP, cross[NATMUL, Id]]
```

Corollary.

```
In[61]:= Assoc[composite[NATMUL, cross[NATEXP, NATEXP], TWIST],
           cross[Id, DUP], RIGHT[x]] // Reverse
```

```
Out[61]= composite[NATEXP, RIGHT[x], NATMUL] ==
          composite[NATMUL, cross[composite[NATEXP, RIGHT[x]], composite[NATEXP, RIGHT[x]]]]
```

```
In[62]:= composite[NATEXP, RIGHT[x_], NATMUL] :=
          composite[NATMUL, cross[composite[NATEXP, RIGHT[x]], composite[NATEXP, RIGHT[x]]]]
```

the third law of exponents

Removing **z** with **reify** yields:

```
In[67]:= Map[equal[composite[NATEXP, LEFT[x], times[y]], VERTSECT[#]] &,
          SubstTest[reify, z, natexp[w, z], w → natexp[x, y]]]
```

```
Out[67]= True ==
          equal[composite[NATEXP, LEFT[natexp[x, y]]], composite[NATEXP, LEFT[x], times[y]]]
```

The orientation of this rewrite rule was chosen so that the two occurrence of exponentiation is reduced to a single one.

```
In[68]:= composite[NATEXP, LEFT[natexp[x_, y_]]] := composite[NATEXP, LEFT[x], times[y]]
```

Reifying again, one obtains:

```
In[70]:= Map[rotate[inverse[#]] &,
          SubstTest[reify, y, composite[w, LEFT[natexp[x, y]]], w → NATEXP]]

Out[70]= composite[NATEXP, LEFT[x], NATMUL] ==
          composite[NATEXP, cross[composite[NATEXP, LEFT[x]], Id]]

In[71]:= composite[NATEXP, LEFT[x_], NATMUL] :=
          composite[NATEXP, cross[composite[NATEXP, LEFT[x]], Id]]
```

This rule is oriented by analogy with an existing rule:

```
In[73]:= composite[NATMUL, LEFT[x], NATMUL]

Out[73]= composite[NATMUL, cross[Id, times[x]]]
```

Another application of **reify** yields:

```
In[74]:= Map[composite[rotate[inverse[#]], RIGHT[PAIR[y, x]]] &,
          SubstTest[reify, x, composite[w, LEFT[x], NATMUL], w → NATEXP]]

Out[74]= composite[NATEXP, RIGHT[x], NATEXP, RIGHT[y]] == composite[NATEXP, RIGHT[natmul[x, y]]]

In[75]:= composite[NATEXP, RIGHT[x_], NATEXP, RIGHT[y_]] :=
          composite[NATEXP, RIGHT[natmul[x, y]]]
```

Reifying again, one obtains:

```
In[76]:= Map[flip[rotate[inverse[#]]] &,
           SubstTest[reify, y, composite[w, RIGHT[x], w, RIGHT[y]], w → NATEXP]] // Reverse

Out[76]= composite[NATEXP, RIGHT[x], NATEXP] == composite[NATEXP, cross[Id, times[x]]]

In[77]:= composite[NATEXP, RIGHT[x_], NATEXP] := composite[NATEXP, cross[Id, times[x]]]
```

A final application of **reify** yields a variable-free statement of the quasi-associative third law of exponents.

```
In[78]:= Map[flip[rotate[inverse[#]]] &,
          SubstTest[reify, x, composite[w, RIGHT[x], w], w → NATEXP]]

Out[78]= composite[NATEXP, cross[Id, NATMUL], ASSOC] == composite[NATEXP, cross[NATEXP, Id]]

In[79]:= composite[NATEXP, cross[Id, NATMUL], ASSOC] := composite[NATEXP, cross[NATEXP, Id]]
```