# NATEXP, part 4.

Johan G. F. Belinfante
2006 July 2

In[1]:=
```
SetDirectory["l:"]; << goedel83.01a; << tools.m
```

```
:Package Title: goedel83.01a         2006 July 1 at 11:50 a.m.

It is now:  2006 Jul 2 at 7:13

Loading Simplification Rules

TOOLS.M                     Revised 2006 June 27

weightlimit = 40
```

## summary

In this fourth notebook in a series about exponentiation for natural numbers it is shown by induction that the number of subsets of a natural number **n** is $\mathbf{2}^{n}$. More generally, the cardinality of the power set of any finite set **x** is equal to $\mathbf{2}^{\mathbf{card}[x]}$.

## ADJOIN

The restriction of **ADJOIN[set[x]]** to the power set of a natural number **x** is one-to-one.

In[2]:=
```
Map[subclass[intersection[#, cart[P[nat[x]], P[nat[x]]]], Id] &,
  composite[inverse[ADJOIN[set[nat[x]]]], ADJOIN[set[nat[x]]]] // RelnNormality]
```

Out[2]=
```
FUNCTION[composite[id[P[nat[x]]], inverse[ADJOIN[set[nat[x]]]]]] == True
```

In[3]:=
```
FUNCTION[composite[id[P[nat[x_]]], inverse[ADJOIN[set[nat[x_]]]]]] := True
```

This function is also finite.

In[4]:=
```
SubstTest[member, domain[funpart[w]], FINITE,
   w -> composite[ADJOIN[set[nat[x]]], id[P[nat[x]]]]] // Reverse
```

Out[4]=
```
member[composite[ADJOIN[set[nat[x]]], id[P[nat[x]]]], FINITE] == True
```

In[5]:=
```
member[composite[ADJOIN[set[nat[x_]]], id[P[nat[x_]]]], FINITE] := True
```

The domain and range of a finite bijection have equal cardinality.

```
In[6]:= SubstTest[implies, and[member[w, BIJ], member[domain[w], FINITE]],
         equal[card[domain[w]], card[range[w]]],
         w -> composite[ADJOIN[set[nat[x]]], id[P[nat[x]]]]]

Out[6]= equal[card[intersection[complement[P[complement[set[nat[x]]]]], P[succ[nat[x]]]]],
          card[P[nat[x]]]] == True

In[7]:= card[intersection[complement[P[complement[set[nat[x_]]]]], P[succ[nat[x_]]]]] :=
         card[P[nat[x]]]
```

The union of the power set of a natural number **x** and its image under **ADJOIN[set[x]]** is the power set of **succ[x]**.

```
In[8]:= SubstTest[image, CUP, cart[P[u], P[v]], {u → nat[x], v → set[nat[x]]}]

Out[8]= union[intersection[complement[P[complement[set[nat[x]]]]], P[succ[nat[x]]]],
          P[nat[x]]] == P[succ[nat[x]]]

In[9]:= union[intersection[complement[P[complement[set[nat[x_]]]]], P[succ[nat[x_]]]],
          P[nat[x_]]] := P[succ[nat[x]]]
```

The image of **x** under **ADJOIN[set[x]]** is finite.

```
In[10]:= SubstTest[member, card[w], omega, w ->
            intersection[complement[P[complement[set[nat[x]]]]], P[succ[nat[x]]]]] // Reverse

Out[10]= member[intersection[complement[P[complement[set[nat[x]]]]], P[succ[nat[x]]]],
           FINITE] == True

In[11]:= member[intersection[
            complement[P[complement[set[nat[x_]]]]], P[succ[nat[x_]]]], FINITE] := True
```

The cardinality of the union of two disjoint finite sets is the sum of their cardinalities.

```
In[12]:= SubstTest[implies, and[disjoint[u, v], member[u, FINITE], member[v, FINITE]],
          equal[card[union[u, v]], natadd[card[u], card[v]]],
          {u -> intersection[complement[P[complement[set[nat[x]]]]], P[succ[nat[x]]]],
           v -> P[nat[x]]}]

Out[12]= equal[card[P[succ[nat[x]]]], natadd[card[P[nat[x]]], card[P[nat[x]]]]] == True

In[13]:= card[P[succ[nat[x_]]]] := natadd[card[P[nat[x]]], card[P[nat[x]]]]
```

## base case

Lemma.

```
In[14]:= equal[union[complement[image[V, set[natexp[x, y]]]], natexp[x, y]], natexp[x, y]]

Out[14]= True

In[15]:= union[complement[image[V, set[natexp[x_, y_]]]], natexp[x_, y_]] := natexp[x, y]
```

The **0**th power of any number is **1**. For a non-number, the **0**th power is **V**.

```
In[16]:= Map[A, ImageComp[NATEXP, RIGHT[0], set[x]]] // Reverse

Out[16]= natexp[x, 0] == union[complement[image[V, intersection[omega, set[x]]]], set[0]]

In[17]:= natexp[x_, 0] := union[complement[image[V, intersection[omega, set[x]]]], set[0]]
```

These results suffice for the base case.

```
In[18]:= equal[card[P[0]], natexp[succ[set[0]], 0]]

Out[18]= True
```

## induction step

In general, one obtains the next power of a natural number **x** by multiplying by another factor of **x**.

```
In[19]:= SubstTest[natexp, nat[x], natadd[nat[y], z], z → set[0]]

Out[19]= natexp[nat[x], succ[nat[y]]] == natmul[nat[x], natexp[nat[x], nat[y]]]

In[20]:= natexp[nat[x_], succ[nat[y_]]] := natmul[nat[x], natexp[nat[x], nat[y]]]
```

For the case **x = 2**, multiplying a number by **2** is equivalent to adding the number to itself.

```
In[21]:= SubstTest[natexp, nat[w], succ[nat[x]], w → succ[set[0]]]

Out[21]= natexp[succ[set[0]], succ[nat[x]]] ==
          natadd[natexp[succ[set[0]], nat[x]], natexp[succ[set[0]], nat[x]]]

In[22]:= natexp[succ[set[0]], succ[nat[x_]]] :=
          natadd[natexp[succ[set[0]], nat[x]], natexp[succ[set[0]], nat[x]]]
```

## induction

To carry out an induction argument using the **GOEDEL** program, one needs to construct an inductive class. To construct this class, one can use the fact that the combination of **VERTSECT** with **reify** is equivalent to **lambda**.

```
In[23]:= VERTSECT[reify[x, card[P[x]]]]

Out[23]= composite[CARD, POWER]

In[24]:= VERTSECT[reify[x, natexp[succ[set[0]], x]]]

Out[24]= composite[NATEXP, LEFT[succ[set[0]]]]
```

One is thus led to consider the following class.

```
In[25]:= fix[composite[inverse[composite[CARD, POWER]], composite[NATEXP, LEFT[succ[set[0]]]]]]

Out[25]= image[fix[composite[inverse[NATEXP], CARD, POWER, SECOND]], set[succ[set[0]]]]
```

The following membership rule is needed to work with this class.

```
In[26]:= member[pair[u, v], fix[composite[w, SECOND]]] // AssertTest

Out[26]= member[pair[u, v], fix[composite[w, SECOND]]] ==
          and[member[u, V], member[v, V], member[pair[union[v,
              complement[image[V, set[u]]], complement[image[V, set[v]]]], pair[u, v]], w]]

In[27]:= member[pair[u_, v_], fix[composite[w_, SECOND]]] :=
          and[member[u, V], member[v, V], member[pair[union[v,
              complement[image[V, set[u]]], complement[image[V, set[v]]]], pair[u, v]], w]]
```

One also needs a special membership rule for **NATEXP**.

```
In[28]:= SubstTest[member, pair[pair[x, y], z],
           composite[w, id[cart[V, V]]], w → NATEXP] // Reverse

Out[28]= and[member[x, V], member[y, V], member[z, V], member[pair[pair[x, y], z], NATEXP]] ==
          member[pair[pair[x, y], z], NATEXP]

In[29]:= and[member[x_, V], member[y_, V], member[z_, V],
           member[pair[pair[x_, y_], z_], NATEXP]] := member[pair[pair[x, y], z], NATEXP]

In[30]:= SubstTest[member, z, image[w, cart[set[x], set[y]]], w → NATEXP] // Reverse

Out[30]= member[pair[pair[x, y], z], NATEXP] == and[equal[z, natexp[x, y]], member[z, V]]

In[31]:= member[pair[pair[x_, y_], z_], NATEXP] := and[equal[z, natexp[x, y]], member[z, V]]
```

Finally, one needs to use the fact that any finite set belongs to the class **image[Q, OMEGA]**.

```
In[32]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
          {u → P[nat[x]], v → FINITE, w → image[Q, OMEGA]}]

Out[32]= member[P[nat[x]], image[Q, OMEGA]] == True

In[33]:= member[P[nat[x_]], image[Q, OMEGA]] := True

In[34]:= SubstTest[member, P[nat[y]], image[Q, OMEGA], y → succ[nat[x]]]

Out[34]= member[P[succ[nat[x]]], image[Q, OMEGA]] == True

In[35]:= member[P[succ[nat[x_]]], image[Q, OMEGA]] := True
```

The induction step needs to be rewritten without **nat** wrappers so that one can eliminate the variables.

```
In[36]:= SubstTest[implies, equal[x, nat[y]],
          or[and[equal[card[P[succ[x]]], natexp[succ[set[0]], succ[x]]],
            member[P[succ[x]], image[Q, OMEGA]]],
           not[equal[card[P[x]], natexp[succ[set[0]], x]]], not[member[x, V]],
           not[member[P[x], image[Q, OMEGA]]]]], y → x]

Out[36]= or[and[equal[card[P[succ[x]]], natexp[succ[set[0]], succ[x]]],
            member[P[succ[x]], image[Q, OMEGA]]],
           not[equal[card[P[x]], natexp[succ[set[0]], x]]], not[member[x, omega]],
           not[member[P[x], image[Q, OMEGA]]]] == True

In[37]:= (% /. x → x_) /. Equal → SetDelayed
```

At this point, one can eliminate the variable **x**.

```
In[38]:= Map[equal[V, #] &,
           SubstTest[class, x, implies[and[member[x, omega], member[x, y]], member[succ[x], y]],
            y -> image[fix[composite[inverse[NATEXP], CARD, POWER, SECOND]],
              set[succ[set[0]]]]]] // Reverse

Out[38]= subclass[intersection[omega, image[SUCC,
             image[fix[composite[inverse[NATEXP], CARD, POWER, SECOND]], set[succ[set[0]]]]]],
           image[fix[composite[inverse[NATEXP], CARD, POWER, SECOND]],
            set[succ[set[0]]]]] == True

In[39]:= % /. Equal → SetDelayed
```

Applying induction, one finds:

```
In[40]:= SubstTest[implies, INDUCTIVE[w], subclass[omega, w],
          w → intersection[omega,
            image[fix[composite[inverse[NATEXP], CARD, POWER, SECOND]], set[succ[set[0]]]]]]

Out[40]= subclass[omega, image[
            fix[composite[inverse[NATEXP], CARD, POWER, SECOND]], set[succ[set[0]]]]] == True

In[41]:= % /. Equal → SetDelayed
```

Finally, variables are reintroduced to obtain the main theorem:

```
In[42]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
          {u → nat[x], v → omega,
           w -> image[fix[composite[inverse[NATEXP], CARD, POWER, SECOND]], set[succ[set[0]]]]}]

Out[42]= equal[card[P[nat[x]]], natexp[succ[set[0]], nat[x]]] == True

In[43]:= card[P[nat[x_]]] := natexp[succ[set[0]], nat[x]]
```

## variable-free formulation

Lemma.

```
In[44]:= Map[equal[0, #] &,
           SubstTest[reify, x, dif[set[P[nat[x]]], y], y -> image[Q, OMEGA]]] // Reverse

Out[44]= subclass[image[POWER, omega], image[Q, OMEGA]] == True

In[45]:= % /. Equal → SetDelayed
```

Variable-free formulation of the theorem derived in the preceding section.

```
In[46]:= Map[composite[VERTSECT[#], id[omega]] &,
           SubstTest[reify, x, card[P[f[x]]], f → nat]] // Reverse

Out[46]= composite[CARD, POWER, id[omega]] == composite[NATEXP, LEFT[succ[set[0]]]]

In[47]:= composite[CARD, POWER, id[omega]] := composite[NATEXP, LEFT[succ[set[0]]]]
```

## generalization to arbitrary finite sets

Lemma.

```
In[48]:= Map[member[fin[y], #] &, ImageComp[Q, id[FINITE], set[nat[x]]]] // Reverse

Out[48]= member[pair[nat[x], fin[y]], Q] == equal[card[fin[y]], nat[x]]

In[49]:= member[pair[nat[x_], fin[y_]], Q] := equal[card[fin[y]], nat[x]]
```

More generally, equipollence of finite sets is equivalent to the statement that their cardinalities are equal.

```
In[50]:= Map[member[fin[y], #] &, ImageComp[Q, id[FINITE], set[fin[x]]]] // Reverse

Out[50]= member[pair[fin[x], fin[y]], Q] == equal[card[fin[x]], card[fin[y]]]

In[51]:= member[pair[fin[x_], fin[y_]], Q] := equal[card[fin[x]], card[fin[y]]]
```

Lemma.

```
In[52]:= SubstTest[member, pair[fin[u], fin[v]],
           Q, {u → P[nat[x]], v → P[fin[y]]}] /. x → card[fin[y]]

Out[52]= member[pair[P[card[fin[y]]], P[fin[y]]], Q] ==
          equal[card[P[fin[y]]], natexp[succ[set[0]], card[fin[y]]]]

In[53]:= member[pair[P[card[fin[y_]]], P[fin[y_]]], Q] :=
          equal[card[P[fin[y]]], natexp[succ[set[0]], card[fin[y]]]]
```

Equipollent sets have equipollent power sets. It follows that the cardinality of the power set of a finite set is equal to 2 raised to the power of the cardinality of the set.

```
In[54]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
            {u → pair[nat[x], fin[y]], v → Q, w → composite[inverse[POWER], Q, POWER]}] /.
          x → card[fin[y]]

Out[54]= equal[card[P[fin[y]]], natexp[succ[set[0]], card[fin[y]]]] == True

In[55]:= card[P[fin[x_]]] := natexp[succ[set[0]], card[fin[x]]]
```

A variable-free restatement of this general result:

```
In[56]:= Map[composite[VERTSECT[#], id[FINITE]] &,
            SubstTest[reify, x, card[P[f[x]]], f → fin]] // Reverse

Out[56]= composite[CARD, POWER, id[FINITE]] == composite[NATEXP, LEFT[succ[set[0]]], CARD]

In[57]:= composite[CARD, POWER, id[FINITE]] := composite[NATEXP, LEFT[succ[set[0]]], CARD]
```