monotonicity properties of exponentiation

Johan G. F. Belinfante
2007 January 18

```
In[1]:= SetDirectory["l:"]; << goedel89.13a; << tools.m

        :Package Title: goedel89.13a        2007 January 13 at 9:15 p.m.

        It is now:  2007 Jan 18 at 13:32

        Loading Simplification Rules

        TOOLS.M                       Revised 2007 January 7

        weightlimit = 40
```

summary

In this notebook it is shown that for natural numbers, if $\mathbf{m} \le \mathbf{n}$, then $\mathbf{m}^{\mathbf{p}} \le \mathbf{n}^{\mathbf{p}}$, The converse holds if the power $\mathbf{p}$ is not zero, If the base $\mathbf{b}$ is not zero and if $\mathbf{m} \le \mathbf{n}$, then $\mathbf{b}^{\mathbf{m}} \le \mathbf{b}^{\mathbf{n}}$. The converse holds if $\mathbf{b} > \mathbf{1}$. The proofs use only cardinality arguments and the laws of exponents, and do not require the use of induction. A key observation is that one can reduce inequalities to equations. Note that the inequality $\mathbf{m} \le \mathbf{n}$ holds if and only if one can write $\mathbf{n}$ as $\mathbf{m}$ plus some natural number, and that mumber is in fact $\mathbf{n} - \mathbf{m}$.

```
In[2]:= equal[nat[x], natadd[nat[y], natsub[nat[x], nat[y]]]]

Out[2]= not[member[nat[x], nat[y]]]
```

fixed power

The monotonicity rule for fixed powers can be deduced most simply using a cardinality argument.

```
In[3]:= SubstTest[implies, subclass[fin[u], fin[v]], subclass[card[fin[u]], card[fin[v]]],
          {u → map[nat[z], nat[y]], v → map[nat[z], nat[x]]}] // Reverse // MapNotNot

Out[3]= or[member[nat[x], nat[y]],
          not[member[natexp[nat[x], nat[z]], natexp[nat[y], nat[z]]]]] == True

In[4]:= or[member[nat[x_], nat[y_]],
          not[member[natexp[nat[x_], nat[z_]], natexp[nat[y_], nat[z_]]]]] := True
```

Conversely, one can show that if $\mathbf{p}$ is not zero and $\mathbf{m} < \mathbf{n}$, then $\mathbf{m}^{\mathbf{p}} < \mathbf{n}^{\mathbf{p}}$. The proof also does not require the use of induction. A lemma about mapping classes is needed:

```
In[5]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → map[x, y], v → map[x, z]}]

Out[5]= equal[map[x, y], map[x, z]] == or[equal[0, x], equal[y, z], not[member[x, V]]]

In[6]:= equal[map[x_, y_], map[x_, z_]] := or[equal[0, x], equal[y, z], not[member[x, V]]]
```

Lemma.

```
In[7]:= SubstTest[implies, and[subclass[u, v], member[v, FINITE], equal[card[u], card[v]]],
          equal[u, v], {u → map[nat[z], nat[x]], v → map[nat[z], nat[y]]}] // Reverse // MapNotNot

Out[7]= or[equal[0, nat[z]], not[equal[natexp[nat[x], nat[z]], natexp[nat[y], nat[z]]]],
          not[member[nat[x], nat[y]]]] == True

In[8]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The rule of trichotomy is also needed to obtain the converse theorem:

```
In[9]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[p1, p4],
          implies[p4, p5], implies[and[p3, p5], p6], not[implies[and[p1, p2], p6]],
          {p1 → member[nat[x], nat[y]], p2 → not[equal[0, nat[z]]],
           p3 → not[equal[natexp[nat[x], nat[z]], natexp[nat[y], nat[z]]]],
           p4 → not[member[nat[y], nat[x]]],
           p5 -> not[member[natexp[nat[y], nat[z]], natexp[nat[x], nat[z]]]],
           p6 -> member[natexp[nat[x], nat[z]], natexp[nat[y], nat[z]]]}]] // Reverse

Out[9]= or[equal[0, nat[z]], member[natexp[nat[x], nat[z]], natexp[nat[y], nat[z]]],
          not[member[nat[x], nat[y]]]] == True

In[10]:= or[equal[0, nat[z_]], member[natexp[nat[x_], nat[z_]], natexp[nat[y_], nat[z_]]],
           not[member[nat[x_], nat[y_]]]] := True
```

---

fixed base b = 2

The special case that $\mathbf{m \le n}$ implies $\mathbf{2^m \le 2^n}$ can also be deduced using a cardinality argument.

```
In[11]:= SubstTest[implies, subclass[fin[u], fin[v]],
           subclass[card[fin[u]], card[fin[v]]], {u → P[nat[y]], v → P[nat[x]]}] // Reverse

Out[11]= or[member[nat[x], nat[y]],
           not[member[natexp[succ[set[0]], nat[x]], natexp[succ[set[0]], nat[y]]]]] == True

In[12]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Again, one can obtain a converse as follows.

```
In[13]:= SubstTest[implies, and[subclass[u, v], member[v, FINITE], equal[card[u], card[v]]],
           equal[u, v], {u → P[nat[x]], v → P[nat[y]]}] // Reverse

Out[13]= or[not[equal[natexp[succ[set[0]], nat[x]], natexp[succ[set[0]], nat[y]]]],
           not[member[nat[x], nat[y]]]] == True
```

```
In[14]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed

In[15]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[p3, p4],
           implies[and[p2, p4], p5], not[implies[p1, p5]], {p1 → member[nat[x], nat[y]],
            p2 → not[equal[natexp[succ[set[0]], nat[x]], natexp[succ[set[0]], nat[y]]]],
            p3 → not[member[nat[y], nat[x]]],
            p4 -> not[member[natexp[succ[set[0]], nat[y]], natexp[succ[set[0]], nat[x]]]], p5 ->
             member[natexp[succ[set[0]], nat[x]], natexp[succ[set[0]], nat[y]]]}]] // Reverse

Out[15]= or[member[natexp[succ[set[0]], nat[x]], natexp[succ[set[0]], nat[y]]],
          not[member[nat[x], nat[y]]]] == True

In[16]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

These two results can be combined into a single rewrite rule.

```
In[17]:= equiv[member[natexp[succ[set[0]], nat[x]], natexp[succ[set[0]], nat[y]]],
           member[nat[x], nat[y]]]

Out[17]= True

In[18]:= member[natexp[succ[set[0]], nat[x_]], natexp[succ[set[0]], nat[y_]]] :=
           member[nat[x], nat[y]]
```

---

general case of a fixed base

Theorem. If $\mathbf{m \le n}$, then $\mathbf{b^m \mid b^n}$.

```
In[19]:= SubstTest[implies, equal[nat[x], natadd[nat[w], nat[y]]],
           member[pair[natexp[nat[z], nat[y]], natexp[nat[z], nat[x]]], DIV],
           w → natsub[nat[x], nat[y]]] // Reverse

Out[19]= or[member[nat[x], nat[y]],
           member[pair[natexp[nat[z], nat[y]], natexp[nat[z], nat[x]]], DIV]] == True

In[20]:= or[member[nat[x_], nat[y_]],
           member[pair[natexp[nat[z_], nat[y_]], natexp[nat[z_], nat[x_]]], DIV]] := True
```

Lemma.

```
In[21]:= or[not[member[x, y]], not[member[pair[y, x], DIV]]] // NotNotTest

Out[21]= or[not[member[x, y]], not[member[pair[y, x], DIV]]] ==
           or[equal[0, y], not[equal[0, x]], not[member[y, omega]]]

In[22]:= or[not[member[x_, y_]], not[member[pair[y_, x_], DIV]]] :=
           or[equal[0, y], not[equal[0, x]], not[member[y, omega]]]
```

Corollary. If $\mathbf{b}$ is not zero and if $\mathbf{m \le n}$, then $\mathbf{b^m \le b^n}$.

```
In[23]:= Map[not, SubstTest[and, implies[p2, p3],
           implies[and[p1, p3], p4], not[implies[and[p1, p2], p4]],
           {p1 → not[empty[nat[x]]], p2 → subclass[nat[z], nat[y]],
            p3 -> member[pair[natexp[nat[x], nat[z]], natexp[nat[x], nat[y]]], DIV],
            p4 → subclass[natexp[nat[x], nat[z]], natexp[nat[x], nat[y]]]}]] // Reverse

Out[23]= or[equal[0, nat[x]], member[nat[y], nat[z]],
           not[member[natexp[nat[x], nat[y]], natexp[nat[x], nat[z]]]] == True

In[24]:= or[equal[0, nat[x_]], member[nat[y_], nat[z_]],
           not[member[natexp[nat[x_], nat[y_]], natexp[nat[x_], nat[z_]]]] := True
```

Corollary.

```
In[25]:= SubstTest[or, equal[0, nat[w]], member[nat[x], nat[y]], not[
            member[natexp[nat[w], nat[x]], natexp[nat[w], nat[y]]]], w → succ[nat[z]]] // Reverse

Out[25]= or[member[nat[x], nat[y]],
           not[member[natexp[succ[nat[z]], nat[x]], natexp[succ[nat[z]], nat[y]]]] == True

In[26]:= or[member[nat[x_], nat[y_]],
           not[member[natexp[succ[nat[z_]], nat[x_]], natexp[succ[nat[z_]], nat[y_]]]] := True
```

---

converse

For the converse, one needs to exclude the case $\mathbf{b = 1}$. It will be shown that if $\mathbf{b > 1}$ and $\mathbf{m < n}$ then $\mathbf{b^m < b^n}$. This proof again does not require the use of induction. This is known: if $\mathbf{b > 0}$ and $\mathbf{m \leq n}$, then $\mathbf{b^m \leq b^n}$. What we want is an implication going in the opposite direction. To begin, just replace **x** with its successor. This yields:

```
In[27]:= SubstTest[implies, and[not[equal[0, nat[w]]], subclass[nat[t], nat[y]]],
           subclass[natexp[nat[w], nat[t]], natexp[nat[w], nat[y]]], t → succ[nat[x]]] // Reverse

Out[27]= or[equal[0, nat[w]], not[member[nat[x], nat[y]]],
           not[member[natexp[nat[w], nat[y]], natmul[nat[w], natexp[nat[w], nat[x]]]]] == True

In[28]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed

Lemma.

In[29]:= equiv[and[member[set[0], nat[w]], not[equal[0, nat[w]]]], member[set[0], nat[w]]]

Out[29]= True

In[30]:= and[member[set[0], nat[w_]], not[equal[0, nat[w_]]]] := member[set[0], nat[w]]
```

Theorem. If $\mathbf{b > 1}$ then $\mathbf{b^m < b^{m+1}}$, and conversely.

```
In[31]:= SubstTest[member, nat[t], natmul[nat[t], nat[w]],
           t -> natexp[nat[w], nat[x]]] // MapNotNot // Reverse

Out[31]= member[natexp[nat[w], nat[x]], natmul[nat[w], natexp[nat[w], nat[x]]]] ==
          member[set[0], nat[w]]

In[32]:= member[natexp[nat[w_], nat[x_]], natmul[nat[w_], natexp[nat[w_], nat[x_]]]] :=
          member[set[0], nat[w]]
```

A form of the transitive law is needed: if $\mathbf{b^m} < \mathbf{b^{m+1}}$ and $\mathbf{b^{m+1}} \leq \mathbf{b^n}$, then $\mathbf{b^m} < \mathbf{b^n}$.

```
In[33]:= SubstTest[implies,
           and[member[nat[t], nat[u]], subclass[nat[u], nat[v]]], member[nat[t], nat[v]],
           {t → natexp[nat[w], nat[x]], u → natexp[nat[w], succ[nat[x]]],
            v → natexp[nat[w], nat[y]]}] // Reverse

Out[33]= or[member[natexp[nat[w], nat[x]], natexp[nat[w], nat[y]]],
           member[natexp[nat[w], nat[y]], natmul[nat[w], natexp[nat[w], nat[x]]]],
           not[member[set[0], nat[w]]]] == True

In[34]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Main theorem.

```
In[35]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p4], p5],
            implies[and[p2, p3], p4], not[implies[and[p1, p3], p5]],
            {p1 → member[set[0], nat[w]], p2 → not[equal[0, nat[w]]], p3 → member[nat[x], nat[y]],
             p4 -> not[member[natexp[nat[w], nat[y]], natmul[nat[w], natexp[nat[w], nat[x]]]]],
             p5 -> member[natexp[nat[w], nat[x]], natexp[nat[w], nat[y]]]}]] // Reverse

Out[35]= or[member[natexp[nat[w], nat[x]], natexp[nat[w], nat[y]]],
           not[member[nat[x], nat[y]]], not[member[set[0], nat[w]]]] == True

In[36]:= or[member[natexp[nat[w_], nat[x_]], natexp[nat[w_], nat[y_]]],
           not[member[nat[x_], nat[y_]]], not[member[set[0], nat[w_]]]] := True
```

The hypothesis **member[set[0],nat[w]]** can be replaced with a double **succ** wrapper. A natural number is greater than **1** if and only if it is the successor of the successor of some number.

```
In[37]:= SubstTest[implies, and[member[set[0], nat[t]], member[nat[x], nat[y]]],
           member[natexp[nat[t], nat[x]], natexp[nat[t], nat[y]]],
           t → succ[succ[nat[w]]]] // Reverse

Out[37]= or[member[natexp[succ[succ[nat[w]]], nat[x]], natexp[succ[succ[nat[w]]], nat[y]]],
           not[member[nat[x], nat[y]]]] == True

In[38]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

The converse also holds:

```
In[39]:= SubstTest[implies, member[natexp[succ[nat[t]], nat[x]], natexp[succ[nat[t]], nat[y]]],
           member[nat[x], nat[y]], t → succ[nat[w]]] // Reverse

Out[39]= or[member[nat[x], nat[y]], not[member[
             natexp[succ[succ[nat[w]]], nat[x]], natexp[succ[succ[nat[w]]], nat[y]]]]] == True

In[40]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Combining the two directions, one finds:

```
In[41]:= equiv[member[natexp[succ[succ[nat[w]]], nat[x]], natexp[succ[succ[nat[w]]], nat[y]]],
          member[nat[x], nat[y]]]

Out[41]= True

In[42]:= member[natexp[succ[succ[nat[w_]]], nat[x_]], natexp[succ[succ[nat[w_]]], nat[y_]]] :=
          member[nat[x], nat[y]]
```

Comment. The special case **w = 0** was derived previously using a cardinality argument.