derivations of n ≤ n! and of m ≤ n ⇒ m! ≤ n!.

Johan G. F. Belinfante
2005 April 23

```
In[1]:= SetDirectory["i:"]; << goedel68.21c; << tools.m

        :Package Title: goedel68.21c          2005 April 21 at 11:10 p.m.

        It is now:  2005 Apr 23 at 14:53

        Loading Simplification Rules

        TOOLS.M                       Revised 2005 April 16

        weightlimit = 40
```

summary

Quaife's theorem **(F!4)** about factorials is derived in this notebook, along with some other facts concerning monotonicity of the factorial function.

```
In[2]:= "Art Quaife, Automated Development of Fundamental
          Mathematical Theories, Kluwer Academic Publishers,
          Dordrecht, the Netherlands, 1992.  See page 212."

Out[2]= Art Quaife, Automated Development of
          Fundamental Mathematical Theories, Kluwer Academic
          Publishers, Dordrecht, the Netherlands, 1992.  See page 212.
```

corollary of a dichotomy theorem

In this section a corollary is derived of a version of the dichotomy of ordering for natural numbers.

```
In[3]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
          implies[and[p1, p3], p4], not[implies[and[p1, p2], p4]], {p1 → member[x, y],
           p2 → member[y, omega], p3 → member[x, omega], p4 → not[subclass[y, x]]}]]

Out[3]= or[not[member[x, y]], not[member[y, omega]], not[subclass[y, x]]] == True

In[4]:= or[not[member[x_, y_]], not[member[y_, omega]], not[subclass[y_, x_]]] := True
```

Comment. The **GOEDEL** program already contained a similar result, but with the literal **member[x,omega]** in place of the literal **member[y,omega]**. That old result was used here to derive the new one.

---

sethood and thin-ness

The factorial function is a set.

```
In[5]:= SubstTest[implies, and[subclass[u, v], member[v, V]], member[u, V],
          {u → FACTORIAL, v → cart[omega, omega]}]

Out[5]= member[FACTORIAL, V] == True
```

The same fact can be derived more simply using **AssertTest**.

```
In[6]:= member[FACTORIAL, V] // AssertTest

Out[6]= member[FACTORIAL, V] == True

In[7]:= member[FACTORIAL, V] := True
```

Corollary.

```
In[8]:= thin[inverse[FACTORIAL]]

Out[8]= True
```

The following consequences of thin-ness may produce formulas involving **IMAGE** functions that we wish to avoid.

```
In[9]:= composite[FACTORIAL, inverse[E]]

Out[9]= composite[inverse[E], IMAGE[FACTORIAL]]

In[10]:= composite[E, FACTORIAL]

Out[10]= composite[inverse[IMAGE[inverse[FACTORIAL]]], E]
```

---

consequences of the fact that n! is nonzero

In this section some rewrite rules are derived from the known fact that **n!** cannot be zero.

```
In[11]:= Map[not,
          SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → image[FACTORIAL, x], v → range[FACTORIAL], w -> complement[set[0]]}]]

Out[11]= member[0, image[FACTORIAL, x]] == False

In[12]:= member[0, image[FACTORIAL, x_]] := False
```

For inverse images, the **GOEDEL** program recognizes the truth of the following, but currently lacks a corresponding rewrite rule, so it is added now:

```
In[13]:= equal[image[inverse[FACTORIAL], set[0]], 0]

Out[13]= True

In[14]:= image[inverse[FACTORIAL], set[0]] := 0
```

Another consequence:

```
In[15]:= equal[composite[id[complement[set[0]]], FACTORIAL], FACTORIAL]

Out[15]= True

In[16]:= composite[id[complement[set[0]]], FACTORIAL] := FACTORIAL
```

$n \leq n!$

Most of the time, divisors are less than the numbers they divide and numbers divide their factorials. The only problem is dealing with zero:

```
In[17]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → FACTORIAL, v → union[DIV, cart[set[0], omega]],
           w → union[S, cart[omega, set[0]]]}]

Out[17]= subclass[FACTORIAL, union[S, cart[omega, set[0]]]] == True

In[18]:= % /. Equal → SetDelayed
```

This result can be cleaned up to yield a simpler and stronger inclusion.

```
In[19]:= SubstTest[subclass, u, intersection[v, w], {u → FACTORIAL,
           v -> union[S, cart[omega, set[0]]], w → complement[cart[V, set[0]]]}]

Out[19]= subclass[FACTORIAL, S] == True

In[20]:= subclass[FACTORIAL, S] := True
```

Corollary.

```
In[21]:= equal[intersection[FACTORIAL, S], FACTORIAL]

Out[21]= True

In[22]:= intersection[FACTORIAL, S] := FACTORIAL
```

Here is a version with variables, amounting to a statement that **n ≤ n!**.

```
In[23]:= SubstTest[implies, subclass[u, v],
          subclass[APPLY[v, x], APPLY[u, x]], {u → FACTORIAL, v → S}]

Out[23]= subclass[x, APPLY[FACTORIAL, x]] == True

In[24]:= subclass[x_, APPLY[FACTORIAL, x_]] := True
```

Note that one need not add any hypothesis about **x** being a natural number. If **x** is not a number then applying the factorial function with **APPLY** yields **V**. (This would not be the case if one had used **apply** instead.)

Quaife's theorem (F!4)

Lemma.

```
In[25]:= SubstTest[intersection, x, composite[FACTORIAL, id[y]], y → omega] // Reverse

Out[25]= composite[intersection[FACTORIAL, x], id[omega]] == intersection[FACTORIAL, x]

In[26]:= composite[intersection[FACTORIAL, x_], id[omega]] :=
          intersection[FACTORIAL, x]
```

The basic idea is to use the dichotomy theorem.

```
In[27]:= AssInt[FACTORIAL, composite[inverse[E], id[omega]], S] // Reverse

Out[27]= intersection[FACTORIAL, inverse[E]] == 0

In[28]:= intersection[FACTORIAL, inverse[E]] := 0
```

Corollary.

```
In[29]:= SubstTest[range, intersection[u, v], {u → FACTORIAL, v → inverse[E]}] // Reverse

Out[29]= fix[composite[FACTORIAL, E]] == 0

In[30]:= fix[composite[FACTORIAL, E]] := 0
```

Introducing variables, one has:

```
In[31]:= SubstTest[implies, and[member[v, omega], member[u, v]], not[subclass[v, u]],
          {v → x, u → APPLY[FACTORIAL, x]}]

Out[31]= or[not[member[x, omega]], not[member[APPLY[FACTORIAL, x], x]]] == True

In[32]:= (% /. x → x_) /. Equal → SetDelayed
```

Removing an unnecessary hypothesis yields Quaife's theorem **(F!4)** which says that the statement **x!** < **x** is always false. Note that the statement is also false when **x** is not a natural number.

```
In[33]:= Map[not, SubstTest[and, implies[p, q], implies[not[p], q],
            {p → member[x, omega], q → not[member[APPLY[FACTORIAL, x], x]]}]] // Reverse

Out[33]= member[APPLY[FACTORIAL, x], x] == False

In[34]:= member[APPLY[FACTORIAL, x_], x_] := False
```

Comment. Quaife's corollary for this theorem has already been proved earlier.

```
In[35]:= equal[0, APPLY[FACTORIAL, x]]

Out[35]= False
```

---

a general result

Lemma.

```
In[36]:= SubstTest[implies, subclass[u, v],
          subclass[composite[w, u], composite[w, v]], {u →
            intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]],
           v → composite[inverse[SECOND], y], w → NATMUL}]

Out[36]= subclass[composite[NATMUL, intersection[composite[inverse[FIRST], x],
             composite[inverse[SECOND], y]]], composite[DIV, y]] == True

In[37]:= subclass[composite[NATMUL, intersection[composite[inverse[FIRST], x_],
             composite[inverse[SECOND], y_]]], composite[DIV, y_]] := True
```

The following result is similar, but will not actually be used here.

```
In[38]:= SubstTest[implies, subclass[u, v],
          subclass[composite[w, u], composite[w, v]], {u →
            intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]],
           v → composite[inverse[FIRST], x], w → NATMUL}]

Out[38]= subclass[composite[NATMUL, intersection[composite[inverse[FIRST], x],
             composite[inverse[SECOND], y]]], composite[DIV, x]] == True

In[39]:= subclass[composite[NATMUL, intersection[composite[inverse[FIRST], x_],
             composite[inverse[SECOND], y_]]], composite[DIV, x_]] := True
```

---

a subtwining argument

A subtwining result is recalled which shows that **m < n ⇒ m! | n!**.

```
In[40]:= SubstTest[implies, subclass[composite[v, w], composite[u, v]],
          subclass[composite[v, trv[w]], composite[trv[u], v]],
          {u → DIV, v → FACTORIAL, w → composite[id[omega], SUCC]}]

Out[40]= subclass[composite[FACTORIAL, E], composite[DIV, FACTORIAL]] == True

In[41]:= % /. Equal → SetDelayed
```

The following lemma is needed to replace **DIV** with **S**.

```
In[42]:= SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
          {u → DIV, v → union[S, cart[omega, set[0]]], w → FACTORIAL}]

Out[42]= subclass[composite[DIV, FACTORIAL],
           union[cart[omega, set[0]], composite[S, FACTORIAL]]] == True

In[43]:= % /. Equal → SetDelayed
```

Carrying out this replacement yields:

```
In[44]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u -> composite[FACTORIAL, E], v -> composite[DIV, FACTORIAL],
           w → composite[union[S, cart[omega, set[0]]], FACTORIAL]}]

Out[44]= subclass[composite[FACTORIAL, E],
           union[cart[omega, set[0]], composite[S, FACTORIAL]]] == True

In[45]:= % /. Equal → SetDelayed
```

The result can be cleaned up, yielding a variable-free formulation of the property **m < n ⇒ m! ≤ n!**.

```
In[46]:= SubstTest[subclass, u, intersection[v, w],
          {u -> composite[FACTORIAL, E],
           v -> union[cart[omega, set[0]], composite[S, FACTORIAL]],
           w → cart[V, complement[set[0]]]}]

Out[46]= subclass[composite[FACTORIAL, E], composite[S, FACTORIAL]] == True

In[47]:= subclass[composite[FACTORIAL, E], composite[S, FACTORIAL]] := True
```

---

monotonicity of the factorial function: $m \leq n \Rightarrow m! \leq n!$.

Lemma.

```
In[48]:= SubstTest[composite, u, union[v, w],
            {u → FACTORIAL, v → composite[id[omega], E], w → id[omega]}] // Reverse

Out[48]= union[FACTORIAL, composite[FACTORIAL, E]] == composite[FACTORIAL, S, id[omega]]

In[49]:= % /. Equal → SetDelayed
```

Corollary 1.

```
In[50]:= SubstTest[subclass, union[u, v], w,
          {u -> FACTORIAL, v -> composite[FACTORIAL, E], w -> composite[S, FACTORIAL]}]

Out[50]= subclass[composite[FACTORIAL, S, id[omega]], composite[S, FACTORIAL]] == True

In[51]:= subclass[composite[FACTORIAL, S, id[omega]], composite[S, FACTORIAL]] := True
```

Corollary 2. Variable-free formulation of the property $\mathbf{m \leq n \Rightarrow m! \leq n!}$.

```
In[52]:= SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
          {u -> composite[FACTORIAL, S, id[omega]],
           v -> composite[S, FACTORIAL], w → inverse[FACTORIAL]}]

Out[52]= subclass[composite[FACTORIAL, S, inverse[FACTORIAL]], S] == True

In[53]:= subclass[composite[FACTORIAL, S, inverse[FACTORIAL]], S] := True
```

---

further corollaries

Lemma.

```
In[54]:= composite[inverse[FACTORIAL], id[omega]] // DoubleInverse

Out[54]= composite[inverse[FACTORIAL], id[omega]] == inverse[FACTORIAL]
```

```
In[55]:= composite[inverse[FACTORIAL], id[omega]] := inverse[FACTORIAL]
```

Further corollaries:

```
In[56]:= Map[range, AssInt[composite[FACTORIAL, S, inverse[FACTORIAL]],
           complement[S], composite[inverse[E], id[omega]]]]

Out[56]= fix[composite[FACTORIAL, S, inverse[FACTORIAL], E]] == 0

In[57]:= fix[composite[FACTORIAL, S, inverse[FACTORIAL], E]] := 0

In[58]:= SubstTest[composite, id[range[x]], x, x → intersection[
             composite[FACTORIAL, S, inverse[FACTORIAL]], inverse[E]]] // Reverse

Out[58]= intersection[composite[FACTORIAL, S, inverse[FACTORIAL]], inverse[E]] == 0

In[59]:= intersection[composite[FACTORIAL, S, inverse[FACTORIAL]], inverse[E]] := 0
```