m divides n! if 0 < m ≤ n.

Johan G. F. Belinfante
2005 April 21

```
In[1]:= SetDirectory["i:"]; << goedel68.19a; << tools.m

        :Package Title: goedel68.19a          2005 April 19 at 2:00 p.m.

        It is now:  2005 Apr 21 at 12:47

        Loading Simplification Rules

        TOOLS.M                      Revised 2005 April 16

        weightlimit = 40
```

summary

If **m** is a nonzero number, then **m** divides **m!**. More generally, **m** divides **n!** when **0 < m ≤ n**. A variable-free version of this result is derived first, and then interpreted by introducing variables wrapped with **nat**.

simplification rules

This section contains some rewrite rules that simplify various expressions encountered in this notebook.

```
In[2]:= Assoc[FACTORIAL, id[omega], id[x]]

Out[2]= composite[FACTORIAL, id[intersection[omega, x]]] == composite[FACTORIAL, id[x]]

In[3]:= composite[FACTORIAL, id[intersection[omega, x_]]] :=
          composite[FACTORIAL, id[x]]

In[4]:= Assoc[DIV, id[omega], id[x]]

Out[4]= composite[DIV, id[intersection[omega, x]]] == composite[DIV, id[x]]

In[5]:= composite[DIV, id[intersection[omega, x_]]] := composite[DIV, id[x]]
```

a lemma

Lemma.

```
In[6]:= SubstTest[implies, subclass[u, v],
          subclass[composite[t, u, w], composite[t, v, w]],
          {t → NATMUL, u → id[x], v → Id, w → inverse[FIRST]}]

Out[6]= subclass[composite[NATMUL, id[x], inverse[FIRST]], DIV] == True

In[7]:= subclass[composite[NATMUL, id[x_], inverse[FIRST]], DIV] := True
```

Theorem. (If **n** is a nonzero number, then **n** divides **n!** .)

```
In[8]:= Map[subclass[#, DIV] &, Assoc[FACTORIAL,
           composite[id[omega], SUCC], composite[inverse[SUCC], id[omega]]]]

Out[8]= subclass[composite[FACTORIAL, id[complement[set[0]]]], DIV] == True

In[9]:= subclass[composite[FACTORIAL, id[complement[set[0]]]], DIV] := True
```

Corollary.

```
In[10]:= SubstTest[subclass, dif[u, v], w,
            {u → FACTORIAL, v → cart[set[0], omega], w → DIV}] // Reverse

Out[10]= subclass[FACTORIAL, union[DIV, cart[set[0], omega]]] == True

In[11]:= subclass[FACTORIAL, union[DIV, cart[set[0], omega]]] := True
```

a more general result

Lemma.

```
In[12]:= 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[12]= subclass[composite[NATMUL, intersection[composite[inverse[FIRST], x],
              composite[inverse[SECOND], y]]], composite[DIV, y]] == True

In[13]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

A subtwining result is applied here:

```
In[14]:= 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[14]= subclass[composite[FACTORIAL, E], composite[DIV, FACTORIAL]] == True

In[15]:= % /. Equal → SetDelayed
```

Lemma.

```
In[16]:= SubstTest[implies, subclass[u, v], subclass[composite[w, u], composite[w, v]],
          {u -> FACTORIAL, v -> union[DIV, cart[set[0], omega]], w → DIV}]

Out[16]= subclass[composite[DIV, FACTORIAL], union[DIV, cart[set[0], omega]]] == True

In[17]:= % /. Equal → SetDelayed
```

Theorem.

```
In[18]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
          subclass[u, w], {u -> composite[FACTORIAL, E],
           v -> composite[DIV, FACTORIAL], w -> union[DIV, cart[set[0], omega]]}]

Out[18]= subclass[composite[FACTORIAL, E], union[DIV, cart[set[0], omega]]] == True

In[19]:= subclass[composite[FACTORIAL, E], union[DIV, cart[set[0], omega]]] := True
```

Lemma.

```
In[20]:= SubstTest[composite, u, union[v, w],
           {u → FACTORIAL, v → composite[id[omega], E], w → id[omega]}] // Reverse

Out[20]= union[FACTORIAL, composite[FACTORIAL, E]] == composite[FACTORIAL, S, id[omega]]

In[21]:= % /. Equal → SetDelayed
```

Main result.

```
In[22]:= SubstTest[subclass, union[u, v], w, {u → composite[FACTORIAL, E],
           v → FACTORIAL, w -> union[DIV, cart[set[0], omega]]}]

Out[22]= subclass[composite[FACTORIAL, S, id[omega]],
           union[DIV, cart[set[0], omega]]] == True

In[23]:= subclass[composite[FACTORIAL, S, id[omega]],
           union[DIV, cart[set[0], omega]]] := True
```

interpretation using variables

This section helps understand the result obtained by introducing variables. The natural number wrapper **nat** is used to avoid repetitious hyotheses about the variables being numbers. To help place a variable into a composite, a further corollary is needed:

```
In[25]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
          subclass[u, w], {u → composite[FACTORIAL, id[t], S, id[omega]],
           v → composite[FACTORIAL, S, id[omega]], w -> union[DIV, cart[set[0], omega]]}]

Out[25]= subclass[composite[FACTORIAL, id[t], S, id[omega]],
           union[DIV, cart[set[0], omega]]] == True

In[26]:= (% /. t → t_) /. Equal → SetDelayed
```

Lemma.

```
In[27]:= SubstTest[implies, and[FUNCTION[f], member[z, domain[f]]],
          member[APPLY[f, z], range[f]], {f → FACTORIAL, z → nat[x]}]

Out[27]= member[APPLY[FACTORIAL, nat[x]], range[FACTORIAL]] == True

In[28]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[29]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
          {u → APPLY[FACTORIAL, nat[x]], v → range[FACTORIAL], w → omega}]

Out[29]= member[APPLY[FACTORIAL, nat[x]], omega] == True

In[30]:= member[APPLY[FACTORIAL, nat[x_]], omega] := True
```

Lemma.

```
In[31]:= SubstTest[implies, and[FUNCTION[f], member[u, domain[f]]],
          member[pair[u, APPLY[f, u]], f], {u → nat[x], f → FACTORIAL}]

Out[31]= member[pair[nat[x], APPLY[FACTORIAL, nat[x]]], FACTORIAL] == True

In[32]:= member[pair[nat[x_], APPLY[FACTORIAL, nat[x_]]], FACTORIAL] := True
```

Corollary. If **x** is a nonzero number, and **y** is a number not less than **x**, then **x** divides **y!** .

```
In[33]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
           subclass[u, w], {u → cart[set[nat[x]], set[APPLY[FACTORIAL, nat[y]]]],
            v -> composite[FACTORIAL, id[t], S, id[omega]],
            w -> union[DIV, cart[set[0], omega]]}] /. t → set[nat[y]]

Out[33]= or[equal[0, nat[x]], member[nat[y], nat[x]],
           member[pair[nat[x], APPLY[FACTORIAL, nat[y]]], DIV]] == True

In[34]:= or[equal[0, nat[x_]], member[nat[y_], nat[x_]],
           member[pair[nat[x_], APPLY[FACTORIAL, nat[y_]]], DIV]] := True
```