x^m . x^n = x^(m+n)

Johan G. F. Belinfante
2002 June 6

```
<< goedel52.o31; << tools.m
```

```
:Package Title: goedel52.o31            2002 June 5 at 9:45 p.m.

It is now:  2002 Jun 6 at 0:10

Loading Simplification Rules

TOOLS.M                    Revised 2002 May 22

weightlimit = 40
```

■ Introduction

The law of adding exponents, $x^m \circ x^n = x^{m+n}$, is derived in this notebook. The catch is that addition has not been defined, and so the result comes out in terms of the function family **power[SUCC]**. For this reason the rules derived here are somewhat tentative and may need to be replaced later with rules involving arithmetical operations yet to be introduced. As a corollary, we derive a rewrite rule for the fact that **range[power[x]]** is a transitive relation. This will presumedly be needed to show that this is the transitive closure of **union[Id,x]**. This conjecture is not settled here, although it has been an issue with which we have wrestled for over a month now. The point of departure in this notebook is the following formula, derived yesterday morning:

```
composite[iterate[x, y], image[power[SUCC], z]]
```

```
composite[image[power[x], z], iterate[x, y]]
```

■ Sequestered abstraction

Our first step is to remove the variable **z** in the above formula by using a sequestered abstraction on the symmetric difference:

```
SubstTest[class, pair[z, w], member[w,
        symdif[composite[i, image[s, singleton[z]]],
   composite[image[p, singleton[z]], i]]],
{i -> iterate[x, y], p -> power[x], s -> power[SUCC]}]
```

```
0 == union[intersection[complement[composite[cross[Id, iterate[x, y]], power[SUCC]]],
   composite[cross[inverse[iterate[x, y]], Id], power[x]]],
  intersection[complement[composite[cross[inverse[iterate[x, y]], Id], power[x]]],
   composite[cross[Id, iterate[x, y]], power[SUCC]]]]
```

Since the symmetric difference is empty, the relations are equal:

```
Map[equal[0, #] &, %] // Reverse

equal[composite[cross[Id, iterate[x, y]], power[SUCC]],
  composite[cross[inverse[iterate[x, y]], Id], power[x]]] == True
```

We now use the following rewrite rule to eliminate **iterate** in favor of **power**.

```
iterate[cross[Id, z], Id]

power[z]

Assoc[id[complement[cart[V, V]]], id[cart[V, V]], power[z]]

composite[id[complement[cart[V, V]]], power[z]] == 0

composite[id[complement[cart[V, V]]], power[z_]] := 0
```

For convenience we replace **equal** by **Equal** so that we can using **Map** instead of **SubstTest**.

```
(composite[cross[Id, iterate[x, y]], power[SUCC]] ==
        composite[cross[inverse[iterate[x, y]], Id], power[x]]) /.
 {x -> cross[Id, z], y -> Id}

composite[cross[Id, power[z]], power[SUCC]] ==
 composite[cross[inverse[power[z]], SWAP], inverse[RIF], power[z]]

Map[inverse, %]

composite[inverse[power[SUCC]], cross[Id, inverse[power[z]]]] ==
 composite[inverse[power[z]], RIF, cross[power[z], SWAP]]

Map[rotate, %]

composite[power[z], rotate[inverse[power[SUCC]]]] ==
 composite[SWAP, RIF, cross[power[z], power[z]]]
```

A plain version of this is the law for adding exponents:

```
Map[image[#, cart[u, v]] &, %]

image[power[z], image[iterate[SUCC, v], u]] ==
 composite[image[power[z], u], image[power[z], v]]

composite[image[power[z_], u_], image[power[z_], v_]] :=
 image[power[z], image[iterate[SUCC, v], u]]
```

■ Corollary: range[power[z]] is transitive

```
ImageComp[power[z], id[omega], V] // Reverse

image[power[z], omega] == range[power[z]]

image[power[z_], omega] := range[power[z]]

SubstTest[image, image[power[x], y], z, {x -> SUCC, y -> omega, z -> omega}]

image[range[power[SUCC]], omega] == omega
```

```
image[range[power[SUCC]], omega] := omega

Map[image[#, cart[V, omega]] &, composite[power[z], rotate[inverse[power[SUCC]]]] ==
   composite[SWAP, RIF, cross[power[z], power[z]]]] // Reverse

composite[range[power[z]], range[power[z]]] == range[power[z]]

composite[range[power[z_]], range[power[z_]]] := range[power[z]]
```