\[ x^m \cdot x^n = x^{m+n} \]

Johan G. F. Belinfante
2002 June 6

\[ \langle \text{goedel52.o31}; \langle \text{tools.m} \rangle \text{: Package Title: goedel52.o31} \quad 2002 \text{ June 5 at 9:45 p.m.} \]
\[ \text{It is now: 2002 Jun 6 at 0:10} \]
Loading Simplification Rules
TOOLS.M Revised 2002 May 22

weightlimit = 40

### Introduction

The law of adding exponents, \( x^m \cdot x^n = x^{m+n} \), 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 \( \text{power[SUCC]} \). 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 \( \text{range[power[x]]} \) is a transitive relation. This will presumably be needed to show that this is the transitive closure of \( \text{union[Id,x]} \). 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:

\[
\text{composite[iterate[x, y], image[power[SUCC], z]]}
\]
\[
\text{composite[image[power[x], z], iterate[x, y]]}
\]

### Sequestered abstraction

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

\[
\text{SubstTest[class, pair[z, w], member[w, symdif[composite[i, image[s, singleton[z]]], composite[image[p, singleton[z]], i]]],}
\{i -> iterate[x, y], p -> power[x], s -> power[SUCC]\}]
\]
0 == union[intercetion[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:
We now use the following rewrite rule to eliminate \texttt{iterate} in favor of \texttt{power}.

\begin{verbatim}
iterate[cross[Id, z], Id]
power[z]
\end{verbatim}

For convenience we replace \texttt{equal} by \texttt{Equal} so that we can using \texttt{Map} instead of \texttt{SubstTest}.

\begin{verbatim}
(composite[iterate[x, y], power[SUCC]] ==
 composite[iterate[x, y], Id, power[x]]). /
{x -> cross[Id, z], y -> Id})
composite[cross[iterate[power[z]], SWAP], inverse[RIF], power[z]]

Map[composite, %]
composite[iterate[power[SUCC]], cross[Id, iterate[power[z]]]] ==
 composite[iterate[power[z]], RIF, cross[power[z], SWAP]]

Map[composite, %]
composite[iterate[power[SUCC]], rotate[power[SUCC]]] ==
 composite[SWAP, RIF, cross[power[z], power[z]]]
\end{verbatim}

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

\begin{verbatim}
Map[iterate[0, #] & %] // Reverse
iterate[iterate[x, y], power[SUCC]] == True
\end{verbatim}

\section*{Corollary: range[power[z]] is transitive}

\begin{verbatim}
ImageComp[iterate[power[z]], id[omega], V] // Reverse
image[power[z], omega] == range[power[z]]
image[power[z], omega] := range[power[z]]

image[power[z], omega] := range[power[z]]
SubstTest[image, iterate[power[x], y], z, \{x -> SUCC, y -> omega, z -> omega\}]
image[range[power[SUCC]], omega] == omega
\end{verbatim}
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]]