new absorptive laws

Johan G. F. Belinfante
2002 June 12

```
<< goedel52.o45; << tools.m

:Package Title: goedel52.o45 2002 June 11 at 1:00 a.m.

It is now: 2002 Jun 12 at 12:17

Loading Simplification Rules

TOOLS.M Revised 2002 June 12

weightlimit = 40
```

## introduction

It was a stroke of luck that I noticed this morning that adding an absorptive law for composite simplifies the derivation of the formula connecting NATADD to power[SUCC] enormously. There is a similar law for image, which is not needed for this application, but may be useful in other applications.

## an absorptive law for composites

```plaintext
SubstTest[composite, w, union[u, v], z, {u -> intersection[x, y], v -> x}] // Reverse

union[composite[w, x, z], composite[w, intersection[x, y], z]] == composite[w, x, z]

union[composite[w___, x__, z___], composite[w___, intersection[x__, y___], z___]] :=
  composite[w, x, z]
```

## application to NATADD

```plaintext
flip[rotate[inverse[power[SUCC]]]] // VSTriNormality // Reverse

rotate[composite[
  complement[composite[Di, SECOND, intersection[composite[complement[inverse[E]]],
    IMAGE[cross[SUCC, SUCC]]], inverse[E]]],
  id[intersection[P[cart[omega, V]]],
    subvar[union[cross[SUCC, SUCC], id[cart[Singleton[0], V]]]]], E]] ==
  composite[rotate[inverse[power[SUCC]]]], SWAP

rotate[composite[
  complement[composite[Di, SECOND, intersection[composite[complement[inverse[E]]],
    IMAGE[cross[SUCC, SUCC]]], inverse[E]]],
  id[intersection[P[cart[omega, V]]],
    subvar[union[cross[SUCC, SUCC], id[cart[Singleton[0], V]]]]], E]] :=
  composite[rotate[inverse[power[SUCC]]]], SWAP
```
NATADD // Normality

NATADD == composite[id[omega], rotate[inverse[power[SUCC]]]]

- other absorptive laws

There are similar absorptive laws for the unary functors A, funpart, invar, MAXIMAL, rank, tc and VERTSECT. Only the more promising ones will be added.

SubstTest[A, union[u, v], {u -> intersection[x, y], v -> x}] // Reverse
intersection[A[x], A[intersection[x, y]]] == A[x]


SubstTest[invar, union[u, v], {u -> intersection[x, y], v -> x}] // Reverse
intersection[invar[x], invar[intersection[x, y]]] == invar[x]

intersection[invar[x_], invar[intersection[x_, y_]]] := invar[x]

SubstTest[tc, union[u, v], {u -> intersection[x, y], v -> x}] // Reverse
union[tc[x], tc[intersection[x, y]]] == tc[x]

union[tc[x_], tc[intersection[x_, y_]]] := tc[x]

One can also add absorptive laws for the binary functors image and iterate.

SubstTest[image, union[u, v, z], {u -> intersection[x, inverse[y]], v -> x}] // Reverse
union[fix[composite[x, id[z], y]], image[x, z]] == image[x, z]

union[fix[composite[x_, id[z_], y_]], image[x_, z_]] := image[x, z]

SubstTest[iterate, w, union[u, v], {u -> intersection[x, y], v -> x}] // Reverse
union[iterate[w, x], iterate[w, intersection[x, y]]] == iterate[w, x]

union[iterate[w_, x_], iterate[w_, intersection[x_, y_]]] := iterate[w, x]