power[x]

Johan G. F. Belinfante
2002 May 21

```
<< goedel52.n90; << tools.m
:Package Title: GOEDEL52.n90          2002 May 21 at 1:00 p.m.
It is now:  2002 May 21 at 22:41
Loading Simplification Rules
TOOLS.M                   Revised 2002 May 17
weightlimit = 40
```

■ Normality

power[x] // Normality // Reverse

iterate[cross[Id, x], Id] == power[x]

iterate[cross[Id, x_], Id] := power[x]

SubstTest[iterate, cross[Id, y], Id, y -> composite[Id, x]] // Reverse

power[composite[Id, x]] == power[x]

power[composite[Id, x_]] := power[x]

■ Properties of power[x]

SubstTest[composite, iterate[cross[Id, x], z], SUCC, z -> Id]

composite[power[x], SUCC] == composite[cross[Id, x], power[x]]

composite[power[x_], SUCC] := composite[cross[Id, x], power[x]]

SubstTest[composite, iterate[cross[Id, x], z], id[omega], z -> Id]

composite[power[x], id[omega]] == power[x]

composite[power[x_], id[omega]] := power[x]

■ domain of power[x]

```
SubstTest[member, domain[iterate[u, v]], V, {u -> cross[Id, x], v -> Id}]
```

```
member[domain[power[x]], V] == True
```

```
member[domain[power[x_]], V] := True
```

```
SubstTest[member, domain[iterate[u, v]], succ[omega], {u -> cross[Id, x], v -> Id}]
```

```
or[equal[omega, domain[power[x]]], member[domain[power[x]], omega]] == True
```

```
or[equal[omega, domain[power[x_]]], member[domain[power[x_]], omega]] := True
```

■ simple examples

```
SubstTest[iterate, cross[Id, x], Id, x -> 0] // Reverse
```

```
power[0] == cart[singleton[0], Id]
```

```
power[0] := cart[singleton[0], Id]
```

```
domain[power[0]]
```

```
singleton[0]
```

```
SubstTest[iterate, cross[Id, x], Id, x -> Id]
```

```
iterate[id[cart[V, V]], Id] == power[Id]
```

```
iterate[id[cart[V, V]], Id] := power[Id]
```

```
SubstTest[implies, and[equal[image[w, singleton[0]], v],
        equal[composite[u, w], composite[w, SUCC]]],
    equal[composite[w, id[omega]], iterate[u, v]],
    {u -> cross[Id, Id], v -> Id, w -> cart[omega, Id]}]
```

```
equal[cart[omega, Id], power[Id]] == True
```

```
power[Id] := cart[omega, Id]
```

```
domain[power[Id]]
```

```
omega
```

■ particular powers

```
SubstTest[image, iterate[cross[Id, x], z], singleton[0], z -> Id]
```

```
image[power[x], singleton[0]] == Id
```

```
image[power[x_], singleton[0]] := Id

SubstTest[image, iterate[cross[Id, x], z], singleton[singleton[0]], z -> Id]

image[power[x], singleton[singleton[0]]] == composite[Id, x]

image[power[x_], singleton[singleton[0]]] := composite[Id, x]

SubstTest[image, iterate[cross[Id, x], z], singleton[succ[y]], z -> Id]

image[power[x], singleton[succ[y]]] == composite[x, image[power[x], singleton[y]]]

image[power[x_], singleton[succ[y_]]] := composite[x, image[power[x], singleton[y]]]
```

■ a general formula

```
Map[image[iterate[id[x], y], singleton[#]] &, NestList[succ, 0, 3]]

{y, intersection[x, y], intersection[x, y], intersection[x, y]}

SubstTest[implies, and[equal[image[w, singleton[0]], v],
        equal[composite[u, w], composite[w, SUCC]]],
    equal[composite[w, id[omega]], iterate[u, v]],
    {u -> id[x], v -> y,
  w -> union[cart[singleton[0], y], cart[omega, intersection[x, y]]]}]

equal[iterate[id[x], y],
  union[cart[omega, intersection[x, y]], cart[singleton[0], y]]] == True

iterate[id[x_], y_] := union[cart[omega, intersection[x, y]], cart[singleton[0], y]]

SubstTest[iterate, cross[Id, w], Id, w -> id[x]] // Reverse

power[id[x]] == union[cart[omega, id[x]], cart[singleton[0], Id]]

power[id[x_]] := union[cart[omega, id[x]], cart[singleton[0], Id]]
```