iterate and power

Johan G. F. Belinfante
2002 May 22

```
<< goedel52.n92; << tools.m
```

```
:Package Title: GOEDEL52.N92          2002 May 22 at 12:45 midnite

It is now:  2002 May 22 at 8:18

Loading Simplification Rules

TOOLS.M                   Revised 2002 May 22

weightlimit = 40
```

■ iterates and powers

One readily discovers the connection between **iterate** and **power** by looking at a few examples:

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

```
{y, image[x, y], image[x, image[x, y]], image[x, image[x, image[x, y]]]}
```

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

```
{y, image[x, y], image[x, image[x, y]], image[x, image[x, image[x, y]]]}
```

The obvious conjecture is that for any natural number **n** we have:

```
image[image[power[x], singleton[n]], y] == image[iterate[x, y], singleton[n]];
```

We will actually show that something more general is true; we can replace **singleton[n]** by any class. This more general conjecture can be written as

```
image[image[power[x], z], y] == image[iterate[x, y], z];
```

If this is the case, we can use **abstract** to eliminate the variable **z** altogether.

```
Map[abstract[z, #] &, image[image[power[x], z], y] == image[iterate[x, y], z]]
```

```
composite[SECOND, id[cart[y, V]], power[x]] == iterate[x, y]
```

One can go further, and abstract **y** on the left side.

```
abstract[y, composite[SECOND, id[cart[y, V]], power[x]]]
```

```
inverse[rotate[composite[inverse[power[x]], SWAP]]]
```

Thus, we could write **iterate[x,y]** as:

```
image[inverse[rotate[composite[inverse[power[x]], SWAP]]], y] == iterate[x, y]
```

```
composite[SECOND, id[cart[y, V]], power[x]] == iterate[x, y]
```

This may help to explain, for example, why **iterate[x,y]** preserves unions with respect to its second argument, something that had already been proved independently some time ago.

```
iterate[x, union[y, z]] == union[iterate[x, y], iterate[x, z]]
```

```
True
```

■ proof

The proof of the conjecture is surprisingly short. We just use the uniqueness theorem for **iterate**. This is the whole thing:

```
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 -> y, v -> x, w -> composite[SECOND, id[cart[x, V]], power[y]]}]
```

```
equal[composite[SECOND, id[cart[x, V]], power[y]], iterate[y, x]] == True
```

When this rule was discovered yesterday, I was perplexed as to how it should be oriented. One could of course use this to eliminate the concept of **iterate** in favor of **power**. But this would complicate the statement of many properties of **iterate**, and so we choose to keep both concepts.

```
composite[SECOND, id[cart[x_, V]], power[y_]] := iterate[y, x]
```

■ power[cross[x,y]]

As a corollary, we obtain a formula relating **power[cross[Id,x]]** to **power[x]**:

```
composite[SECOND, id[cart[Id, V]], power[cross[Id, x]]]
```

```
power[x]
```

We will later try to improve on this to get a formula for **power[cross[x,y]]**.

■ back to the beginning

The original conjecture can now be proved:

```
ImageComp[composite[SECOND, id[cart[y, V]]], power[x], z] // Reverse
```

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

It is not entirely clear how to orient this equation. The following is tentative.

```
image[image[power[x_], z_], y_] := image[iterate[x, y], z]
```

In particular:

```
SubstTest[image, image[power[x], z], y, z -> V]
```

```
image[range[power[x]], y] == range[iterate[x, y]]
```

Since the quantity **range[power[x]]** can be rewritten in terms of transitive closure, we hold off adding this rule for now.