domain of iterate[x,y]

Johan G. F. Belinfante
2002 July 1

```
<< goedel52.o85; << tools.m

:Package Title: goedel52.o85          2002 June 30 at 9:55 p.m.

It is now:  2002 Jul 1 at 11:26

Loading Simplification Rules

TOOLS.M                   Revised 2002 June 12

weightlimit = 40
```

■ Remark

This notebook improves upon a result in the notebook **ITER−DO.NB** dated 2002 May 14, and derives a useful corollary. The result obtained in May is this:

```
implies[equal[V, domain[x]], or[equal[0, y], equal[omega, domain[iterate[x, y]]]]]

True
```

The goal is to weaken the hypothesis **equal[V,domain[x]]** in this result.

■ Introduction

The rquirement that **y** be nonempty is needed to assure that **0** belongs to the domain of **iterate[x,y]**. The induction step depends on the following observation.

```
Map[subclass[image[SUCC, #], domain[iterate[x, y]]] &, IminComp[iterate[x, y], SUCC, V]]

subclass[image[SUCC, image[inverse[iterate[x, y]], domain[x]]],
  domain[iterate[x, y]]] == True

subclass[image[SUCC, image[inverse[iterate[x_, y_]], domain[x_]]],
  domain[iterate[x_, y_]]] := True
```

Observe that if **domain[x]** were **V**, then this would say that **domain[iterate[x,y]]** is inductive, and we get back our old result. But clearly it is not required that **domain[x]** be quite this large. The following termporary rule is useful:

```
SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
    {u -> range[x], v -> y, w -> inverse[x]}]

or[not[subclass[range[x], y]], subclass[domain[x], image[inverse[x], y]]] == True

or[not[subclass[range[x_], y_]], subclass[domain[x_], image[inverse[x_], y_]]] := True
```

We can actually do better:

```
Map[implies[subclass[range[x], y], #] &, SubstTest[and, subclass[u, v], subclass[v, u],
        {u -> image[inverse[x], y], v -> domain[x]}]] // Reverse
```

```
or[equal[domain[x], image[inverse[x], y]], not[subclass[range[x], y]]] == True
```

```
or[equal[domain[x_], image[inverse[x_], y_]], not[subclass[range[x_], y_]]] := True
```

■ the old derivation revisited

We now continue along the same path that we took in May, but generalize a bit, where we can.

```
SubstTest[implies, and[equal[u, v], subclass[v, w]], subclass[u, w],
    {v -> image[SUCC, image[inverse[iterate[x, y]], domain[x]]],
  w -> domain[iterate[x, y]]}]
```

```
or[not[equal[u, image[SUCC, image[inverse[iterate[x, y]], domain[x]]]]],
  subclass[u, domain[iterate[x, y]]]] == True
```

```
or[not[equal[u_, image[SUCC, image[inverse[iterate[x_, y_]], domain[x_]]]]],
  subclass[u_, domain[iterate[x_, y_]]]] := True
```

```
Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p3, p4],
  not[implies[p1, p4]], {p1 -> subclass[range[iterate[x, y]], domain[x]],
        p2 -> equal[image[inverse[iterate[x, y]], domain[x]], domain[iterate[x, y]]],
        p3 -> equal[image[SUCC, image[inverse[iterate[x, y]], domain[x]]],
                    image[SUCC, domain[iterate[x, y]]]],
        p4 -> subclass[image[SUCC, domain[iterate[x, y]]], domain[iterate[x, y]]]}]]
```

```
or[not[subclass[range[iterate[x, y]], domain[x]]],
  subclass[image[SUCC, domain[iterate[x, y]]], domain[iterate[x, y]]]] == True
```

```
or[not[subclass[range[iterate[x_, y_]], domain[x_]]],
  subclass[image[SUCC, domain[iterate[x_, y_]]], domain[iterate[x_, y_]]]] := True
```

```
SubstTest[implies, and[member[0, w], subclass[image[SUCC, w], w]], subclass[omega, w],
    w -> domain[iterate[x, y]]]
```

```
or[equal[0, y],
  not[subclass[image[SUCC, domain[iterate[x, y]]], domain[iterate[x, y]]]],
  subclass[omega, domain[iterate[x, y]]]] == True
```

```
or[equal[0, y_],
  not[subclass[image[SUCC, domain[iterate[x_, y_]]], domain[iterate[x_, y_]]]],
  subclass[omega, domain[iterate[x_, y_]]]] := True
```

```
Map[implies[subclass[omega, domain[iterate[x, y]]], #] &, SubstTest[and, subclass[u, v],
   subclass[v, u], {u -> domain[iterate[x, y]], v -> omega}]] // Reverse
```

```
or[equal[omega, domain[iterate[x, y]]],
  not[subclass[omega, domain[iterate[x, y]]]]] == True
```

```
or[equal[omega, domain[iterate[x_, y_]]],
  not[subclass[omega, domain[iterate[x_, y_]]]]] := True
```

The following result is a slight improvement over what was found in May.

```
Map[not, SubstTest[and, implies[p1, p3], implies[and[p2, p3], p4], implies[p4, p5],
        not[implies[and[p1, p2], p5]],
  {p1 -> subclass[range[iterate[x, y]], domain[x]], p2 -> not[empty[y]],
   p3 -> subclass[image[SUCC, domain[iterate[x, y]]], domain[iterate[x, y]]],
    p4 -> subclass[omega, domain[iterate[x, y]]],
   p5 -> equal[omega, domain[iterate[x, y]]]}]]

or[equal[0, y], equal[omega, domain[iterate[x, y]]],
  not[subclass[range[iterate[x, y]], domain[x]]]] == True

or[equal[0, y_], equal[omega, domain[iterate[x_, y_]]],
  not[subclass[range[iterate[x_, y_]], domain[x_]]]] := True
```

■ a new result

The key to continuing is this result:

```
implies[subclass[image[x, y], y], equal[range[iterate[x, y]], y]]

True
```

This is not surprising; if **y** is invariant under **x** then the successive generations in the iteration get progressively smaller, and so the union of all the generations is just the starting generation **y**. Thus, if the starting generation is contained in **domain[x]**, then the weakened hypothesis about the domain of **x** in our improved theorem will hold, and we wind up with the following corollary.

```
Map[not,
 SubstTest[and, implies[p1, p2], implies[and[p2, p3], p4], implies[and[p4, p5], p6],
        not[implies[and[p1, p3, p5], p6]],
    {p1 -> subclass[image[x, y], y],
        p2 -> equal[range[iterate[x, y]], y],
        p3 -> subclass[y, domain[x]],
        p4 -> subclass[range[iterate[x, y]], domain[x]],
        p5 -> not[equal[0, y]],
        p6 -> equal[omega, domain[iterate[x, y]]]}]]

or[equal[0, y], equal[omega, domain[iterate[x, y]]],
  not[subclass[y, domain[x]]], not[subclass[image[x, y], y]]] == True
```

This says that if **y** is a nonempty invariant subclass of the domain of **x**, then the iteration process never ends.

```
or[equal[0, y_], equal[omega, domain[iterate[x_, y_]]],
  not[subclass[y_, domain[x_]]], not[subclass[image[x_, y_], y_]]] := True
```