invar[trv[x]]

Johan G. F. Belinfante
2002 June 23

```
<< goedel52.o68; << tools.m

:Package Title: goedel52.o68        2002 June 22 at 10:00 p.m.

It is now:  2002 Jun 23 at 17:0

Loading Simplification Rules

TOOLS.M                    Revised 2002 June 12

weightlimit = 40
```

■ preliminary observations

```
fix[IMAGE[union[Id, x]]] // Normality

fix[IMAGE[union[Id, x]]] == invar[x]

fix[IMAGE[union[Id, x_]]] := invar[x]
```

The functor **invar** is antitone. As a consequence, one has:

```
SubstTest[implies, subclass[u, v],
 subclass[invar[v], invar[u]], {u -> composite[Id, x], v -> trv[x]}]

subclass[invar[trv[x]], invar[x]] == True

subclass[invar[trv[x_]], invar[x_]] := True
```

In this notebook it is shown, by using the theory of **iterate**, that the reverse inclusion also holds.

■ application of iteration uniqueness

The main tool is the uniqueness of iteration. This yields:

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

or[equal[cart[omega, y], composite[iterate[x, y], inverse[S], id[omega]]],
  not[subclass[image[x, y], y]]] == True

or[equal[cart[omega, y_], composite[iterate[x_, y_], inverse[S], id[omega]]],
  not[subclass[image[x_, y_], y_]]] := True
```

The idea is to look at the ranges of these relations.

■ a technical lemma.

The following technical lemma is useful:

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

```
subclass[range[iterate[x, y]], image[iterate[x, y], image[inverse[S], omega]]] == True
```

```
subclass[range[iterate[x_, y_]],
  image[iterate[x_, y_], image[inverse[S], omega]]] := True
```

It is useful to restate this lemma as an equation so that it can be turned into a rewrite rule.

```
SubstTest[and, subclass[u, v], subclass[v, u],
    {u -> image[iterate[x, y], image[inverse[S], omega]], v -> range[iterate[x, y]]}] //
 Reverse
```

```
equal[image[iterate[x, y], image[inverse[S], omega]], range[iterate[x, y]]] == True
```

```
image[iterate[x_, y_], image[inverse[S], omega]] := range[iterate[x, y]]
```

This equality of relations implies equality of their ranges.

```
SubstTest[implies, equal[u, v], equal[range[u], range[v]],
    {u -> composite[iterate[x, y], inverse[S], id[omega]], v -> cart[omega, y]}]
```

```
or[equal[y, range[iterate[x, y]]],
  not[equal[cart[omega, y], composite[iterate[x, y], inverse[S], id[omega]]]]] == True
```

```
or[equal[y_, range[iterate[x_, y_]]],
  not[equal[cart[omega, y_], composite[iterate[x_, y_], inverse[S], id[omega]]]]] := True
```

The above result can be combined with the fact obtained from uniqueness of iteration as follows:

```
Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
        {p1 -> subclass[image[x, y], y],
   p2 -> equal[cart[omega, y], composite[iterate[x, y], inverse[S], id[omega]]],
         p3 -> equal[y, range[iterate[x, y]]]}]]
```

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

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

■ a corollary

```
Map[equal[#, y] &, SubstTest[image, u, union[v, w],
    {u -> iterate[x, y], v -> complement[singleton[0]], w -> singleton[0]}]] // Reverse
```

```
subclass[image[iterate[x, y], complement[singleton[0]]], y] ==
 equal[y, range[iterate[x, y]]]
```

```
subclass[image[iterate[x_, y_], complement[singleton[0]]], y_] :=
 equal[y, range[iterate[x, y]]]
```

```
Map[equal[V, #] &, union[complement[invar[x]], invar[trv[x]]] // Normality]
```

```
subclass[invar[x], invar[trv[x]]] == True
```

```
subclass[invar[x_], invar[trv[x_]]] := True
```

```
SubstTest[and, subclass[u, v],
  subclass[v, u], {u -> invar[trv[x]], v -> invar[x]}] // Reverse
```

```
equal[invar[x], invar[trv[x]]] == True
```

```
invar[trv[x_]] := invar[x]
```

■ a variable−freeformulation of the theorem.

The result to be proved is suggested by reification theory:

```
SubstTest[reify, x, invar[F[x]], F -> trv]
```

```
INVAR == composite[Id, complement[composite[complement[INVAR], HULL[TRV], inverse[S]]]]
```

Here is a derivation of a better result, using **VSNormality**:

```
composite[complement[INVAR], HULL[TRV], inverse[S]] // VSNormality
```

```
composite[complement[INVAR], HULL[TRV], inverse[S]] == composite[Id, complement[INVAR]]
```

```
composite[complement[INVAR], HULL[TRV], inverse[S]] := composite[Id, complement[INVAR]]
```

Comments:The reification of **invar** is the relation **INVAR**. As usual, the complement of **invar** behaves better than **invar** itself.

```
class[pair[x, y], member[y, invar[x]]]
```

```
INVAR
```