Uclosure[Uclosure[x]]

Johan G. F. Belinfante
2002 January 22

```
<< goedel52.M63; << tests.m
```

```
:Package Title: GOEDEL52.M63      2002 January 22 at 12:20 midnite

It is now:  2002 Jan 22 at 9:39

Loading Simplification Rules

TESTS.M                   Revised 2002 January 12

weightlimit = 40


Context switch to ‘Goedel‘Private is needed for ReplaceTest

Just ignore the error message about Unterminated use of BeginPackage

Get::bebal : Unterminated uses of BeginPackage or Begin in << tests.m.
```

■ The Problem

The function **UCLOSURE** is idempotent, but until now, we had not been able to show that the function symbol **Uclosure[x]** shares this property:

```
composite[UCLOSURE, UCLOSURE]
```

```
UCLOSURE
```

```
Uclosure[Uclosure[x]]
```

```
Uclosure[Uclosure[x]]
```

The inclusion in one direction is no problem:

```
subclass[Uclosure[x], Uclosure[Uclosure[x]]]
```

```
True
```

To prove equality, one needs to show that the inclusion holds in the opposite direction. Any element of **Uclosure[Uclosure[x]]** can be written in the form **U[y]**, where **y** is a subset of **Uclosure[x]**. What one needs to show is that one can write **U[y]** also as **U[z]**, where **z** is a subset of **x**. Each element of **y** can be written as the sum class of a subset of **x**, and although this in general can be done in many ways, one can avoid the axiom of choice by chosing the largest subset of **x** that does the trick. The function

```
f[x_] := composite[BIGCUP, IMAGE[id[P[x]]], VERTSECT[inverse[BIGCUP]]]
```

does the job of selecting for each element of **Uclosure[x]** the largest subset of **x** whose sum class is that element. The idea is to construct the set **z** from the set **y** as **z = image[f[x],y]**.

```
Map[U, z == image[f[x], y]]
```

```
U[z] == U[intersection[image[inverse[BIGCUP], y], P[x]]]
```

■ construction of the choice function f[x]

For each element **w** of **Uclosure[x]**, there exists a subset of **x** whose sum class is **w**. One can select one such subset of **x** without using the axiom of choice. First, note that the class of all such subsets is the intersection of **P[x]** with

```
class[u, member[U[u], z]] /. z -> singleton[w]
```

```
image[inverse[BIGCUP], singleton[w]]
```

The function which takes **w** to this set of candidates is

```
lambda[w, intersection[P[x], image[inverse[BIGCUP], singleton[w]]]]
```

```
composite[IMAGE[id[P[x]]], VERTSECT[inverse[BIGCUP]]]
```

If every element of a class has the same sum class, then their union also has that same sum class. Thus, the largest subset of **x** whose sum class is **w** is **U[intersection[image[inverse[BIGCUP],singleton[w]],P[x]]]**. The function which takes **w** to this largest subset of **x** with sum class **w** is

```
f[x] == composite[BIGCUP, IMAGE[id[P[x]]], VERTSECT[inverse[BIGCUP]]]
```

```
True
```

■ properties of the choice function f[x]

Two special cases are:

```
f[V]
```

```
POWER
```

```
f[0]
```

```
cart[V, singleton[0]]
```

The domain of the choice function **f[x]** is

```
domain[f[x]]
```

```
V
```

There are several connections with **Uclosure[x]**. Consider:

```
subclass[image[f[x], Uclosure[x]], P[x]]
```

```
True
```

The following formula seems to be a key result:

```
composite[BIGCUP, f[x]] // VSNormality
```

```
composite[BIGCUP, BIGCUP, IMAGE[id[P[x]]], VERTSECT[inverse[BIGCUP]]] == union[
  cart[union[complement[Uclosure[x]], singleton[0]], singleton[0]], id[Uclosure[x]]]
```

```
composite[BIGCUP, BIGCUP, IMAGE[id[P[x_]]], VERTSECT[inverse[BIGCUP]]] := union[
  cart[union[complement[Uclosure[x]], singleton[0]], singleton[0]], id[Uclosure[x]]]
```

■ Normality results

```
VERTSECT[inverse[BIGCUP]] // VSNormality // Reverse
```

```
intersection[complement[composite[complement[E], inverse[BIGCUP]]],
  composite[inverse[IMAGE[BIGCUP]], SINGLETON]] == VERTSECT[inverse[BIGCUP]]
```

```
intersection[complement[composite[complement[E], inverse[BIGCUP]]],
  composite[inverse[IMAGE[BIGCUP]], SINGLETON]] := VERTSECT[inverse[BIGCUP]]
```

```
equal[union[singleton[0], Uclosure[x]], Uclosure[x]]
```

```
True
```

```
union[singleton[0], Uclosure[x_]] := Uclosure[x]
```

```
fix[composite[BIGCUP, f[x]]]
```

```
Uclosure[x]
```

```
ImageComp[BIGCUP, f[x], V]
```

```
Uclosure[x] ==
 image[BIGCUP, image[BIGCUP, image[IMAGE[id[P[x]]], range[VERTSECT[inverse[BIGCUP]]]]]]
```

```
ImageComp[BIGCUP, VERTSECT[inverse[BIGCUP]], V] // Reverse
```

```
image[BIGCUP, range[VERTSECT[inverse[BIGCUP]]]] == range[POWER]
```

```
image[BIGCUP, range[VERTSECT[inverse[BIGCUP]]]] := range[POWER]
```

```
Map[implies[member[y, V], #] &, SubstTest[member, y, union[u, v],
   {u -> complement[P[x]], v -> image[inverse[BIGCUP], Uclosure[x]]}]] // Reverse
```

```
or[member[U[y], Uclosure[x]], not[member[y, V]], not[subclass[y, x]]] == True
```

```
or[member[U[y_], Uclosure[x_]], not[member[y_, V]], not[subclass[y_, x_]]] := True
```

```
equal[intersection[image[inverse[BIGCUP], Uclosure[x]], P[x]], P[x]]
```

```
True
```

```
intersection[image[inverse[BIGCUP], Uclosure[x_]], P[x_]] := P[x]
```

```
SubstTest[implies, and[FUNCTION[u], member[v, V]],
    member[image[u, v], V], {u -> f[x], v -> y}]
```

```
or[member[intersection[image[inverse[BIGCUP], y], P[x]], V], not[member[y, V]]] == True
```

```
or[member[intersection[image[inverse[BIGCUP], y_], P[x_]], V],
  not[member[y_, V]]] := True
```

```
ImageComp[BIGCUP, inverse[BIGCUP], w] // Reverse
```

```
image[BIGCUP, image[inverse[BIGCUP], w]] == w
```

```
image[BIGCUP, image[inverse[BIGCUP], w_]] := w
```

```
ImageComp[funpart[u], inverse[funpart[u]], y] /.
  u -> composite[BIGCUP, id[P[x]]] // Reverse
```

```
image[BIGCUP, intersection[image[inverse[BIGCUP], y], P[x]]] ==
 intersection[y, Uclosure[x]]
```

```
image[BIGCUP, intersection[image[inverse[BIGCUP], y_], P[x_]]] :=
 intersection[y, Uclosure[x]]
```

```
SubstTest[implies, equal[y, w], equal[U[y], U[w]],
w -> intersection[y, Uclosure[x]]]
```

```
or[equal[U[y], U[intersection[y, Uclosure[x]]]], not[subclass[y, Uclosure[x]]]] == True
```

```
or[equal[U[y_], U[intersection[y_, Uclosure[x_]]]],
  not[subclass[y_, Uclosure[x_]]]] := True
```

```
SubstTest[U, image[BIGCUP, w],
  w -> intersection[image[inverse[BIGCUP], y], P[x]]] // Reverse
```

```
U[U[intersection[image[inverse[BIGCUP], y], P[x]]]] == U[intersection[y, Uclosure[x]]]
```

```
U[U[intersection[image[inverse[BIGCUP], y_], P[x_]]]] := U[intersection[y, Uclosure[x]]]
```

■ The final steps

```
Map[not, SubstTest[and, implies[p1, p2], implies[and[p2, p4], p3],
        not[implies[and[p1, p4], p3]],
        {p1 -> member[t, P[x]],
p2 -> member[U[t], Uclosure[x]],
        p3 -> member[u, Uclosure[x]],
        p4 -> equal[u, U[t]]}]]
```

```
or[member[u, Uclosure[x]], not[equal[u, U[t]]],
  not[member[t, V]], not[subclass[t, x]]] == True
```

```
or[member[u_, Uclosure[x_]], not[equal[u_, U[t_]]],
  not[member[t_, V]], not[subclass[t_, x_]]] := True
```

```
SubstTest[implies, and[member[t, P[x]], equal[U[t], u]], member[u, Uclosure[x]],
{t -> U[image[f[x], y]], u -> U[y]}]
```

```
or[member[U[y], Uclosure[x]], not[equal[U[y], U[intersection[y, Uclosure[x]]]]],
  not[member[intersection[image[inverse[BIGCUP], y], P[x]], V]]] == True
```

```
or[member[U[y_], Uclosure[x_]], not[equal[U[y_], U[intersection[y_, Uclosure[x_]]]]],
  not[member[intersection[image[inverse[BIGCUP], y_], P[x_]], V]]] := True
```

```
Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
        not[implies[p1, p4]], {p1 -> member[y, P[Uclosure[x]]],
        p2 -> equal[U[U[image[f[x], y]]], U[y]],
p3 -> member[U[image[f[x], y]], P[x]],
        p4 -> member[U[y], Uclosure[x]]}]]
```

```
or[member[U[y], Uclosure[x]], not[member[y, V]], not[subclass[y, Uclosure[x]]]] == True
```

```
or[member[U[y_], Uclosure[x_]],
  not[member[y_, V]], not[subclass[y_, Uclosure[x_]]]] := True
```

```
SubstTest[class, y, or[member[U[y], z], not[member[y, V]], not[subclass[y, z]]],
        z -> Uclosure[x]] // Reverse
```

```
union[complement[P[Uclosure[x]]], image[inverse[BIGCUP], Uclosure[x]]] == V
```

```
union[complement[P[Uclosure[x_]]], image[inverse[BIGCUP], Uclosure[x_]]] := V
```

```
SubstTest[equal, V, union[complement[u], v],
    {u -> P[Uclosure[x]], v -> image[inverse[BIGCUP], Uclosure[x]]}] // Reverse
```

```
subclass[Uclosure[Uclosure[x]], Uclosure[x]] == True
```

```
subclass[Uclosure[Uclosure[x_]], Uclosure[x_]] := True
```

```
SubstTest[and, subclass[u, v], subclass[v, u],
    {u -> Uclosure[Uclosure[x]], v -> Uclosure[x]}]
```

```
True == equal[Uclosure[x], Uclosure[Uclosure[x]]]
```

```
Uclosure[Uclosure[x_]] := Uclosure[x]
```

■ Further thoughts

The following formula may be useful for deriving further results about **Uclosure[x]**.

```
composite[BIGCUP, id[P[x]], inverse[BIGCUP]]
```

```
id[Uclosure[x]]
```