

IMAGE[inverse[S]] commutes with UCLOSURE

Johan G. F. Belinfante
2003 March 1

```
<< goedel52.r25; << tools.m

:Package Title: goedel52.r25          2003 February 26 at 11:00 p.m.

It is now: 2003 Mar 1 at 4:18

Loading Simplification Rules

TOOLS.M                               Revised 2002 December 27

weightlimit = 40
```

■ summary

The hereditary closure function **IMAGE[inverse[S]]** commutes with the function **UCLOSURE**. The derivation of this fact in this notebook illustrates the use of reification and the use of **image[V,x]** to convert a conditional equality into an unconditional one.

■ derivation

If x is a set, then one of the members of **Uclosure[x]** is the union of all members of x .

```
SubstTest[implies, member[y, P[x]], member[U[y], Uclosure[x]], y -> x]
or[member[U[x], Uclosure[x]], not[member[x, V]]] == True

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

The hereditary closure **image[inverse[S],x]** of a class x is the class of all subsets of members of x . If y is a subset of a member of x , then so are all subsets of y .

```
SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
  {u -> singleton[x], v -> y, w -> inverse[S]}]
or[not[member[x, y]], subclass[P[x], image[inverse[S], y]]] == True

or[not[member[x_, y_]], subclass[P[x_], image[inverse[S], y_]]] := True
```

Putting together these two observations yields:

```
Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
  {p1 -> member[x, V], p2 -> member[U[x], Uclosure[x]],
  p3 -> subclass[P[U[x]], image[inverse[S], Uclosure[x]]}]]]
or[not[member[x, V]], subclass[P[U[x]], image[inverse[S], Uclosure[x]]] == True
```

```
or[not[member[x_, V]], subclass[P[U[x_]], image[inverse[S], Uclosure[x_]]] := True
```

The opposite inclusion holds (even without assuming that x is a set), so the above statement can be strengthened to a conditional equality:

```
SubstTest[and, implies[p1, subclass[u, v]], implies[p1, subclass[v, u]],
  {p1 -> member[x, V], u -> P[U[x]], v -> image[inverse[S], Uclosure[x]]} // Reverse
or[equal[image[inverse[S], Uclosure[x]], P[U[x]], not[member[x, V]]] == True
or[equal[image[inverse[S], Uclosure[x_]], P[U[x_]], not[member[x_, V]]] := True
```

This conditional equality can be replaced by a simple equation. The idea is to build the condition that x be a set into the classes on both sides of the equation by intersecting with `image[V, singleton[x]]`.

```
equal[intersection[image[V, singleton[x]], image[inverse[S], Uclosure[x]]],
  intersection[image[V, singleton[x]], P[U[x]]]
True
```

The final step is to replace `equal` with `Equal` and apply reification to this equation:

```
Map[VERTSECT[reify[x, #]] &,
  Equal[intersection[image[V, singleton[x]], image[inverse[S], Uclosure[x]]],
    intersection[image[V, singleton[x]], P[U[x]]]]]
composite[IMAGE[inverse[S]], UCLOSURE] == composite[POWER, BIGCUP]
composite[IMAGE[inverse[S]], UCLOSURE] := composite[POWER, BIGCUP]
```

The composite in the opposite order is the same. Hence:

```
commute[IMAGE[inverse[S]], UCLOSURE]
True
```