composite[CARD,Q]

Johan G. F. Belinfante
2001 December 14

```
<< goedel52.L81; << tests.m;

:Package Title: GOEDEL52.L81       2001 December 12 at 11:50 a.m.

It is now:  2001 Dec 17 at 14:22

Loading Simplification Rules

TESTS.M                   Revised 2001 December 1

weightlimit = 30


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.
```

■ Definitions and Summary of Results

In this notebook the **GOEDEL** program is used to derive a formula that says roughly that equipollent sets has the same cardinal number. More precisely, the composite of the function **CARD** which assigns cardinality, and the equipollence relation **Q** is the function **CARD**. Since the axiom of choice is not assumed here, not every set has a cardinal number. The cardinal number of a class **x** is the least ordinal equipollent to **x**.

```
A[intersection[OMEGA, image[Q, singleton[x]]]]

card[x]
```

In general, there may not be any ordinal that is equipollent to **x**, in which case the above reduces to the universal class **V**, which of course is not an ordinal number.

```
A[0]

V
```

The function **CARD** is (this takes a while)

```
class[pair[x, y], equal[y, card[x]]]

CARD
```

The domain of **CARD** would be the universal class **V** if the axiom of choice were assumed. Here it is the class of sets which are equipollent to some ordinal number:

```
domain[CARD]
```

```
image[Q, OMEGA]
```

■ Review of some prerequisites

In this section we recall some basic definitions. The function symbol **A[ ]** takes a class of sets to their intersection:

```
class[y, forall[z, implies[member[z, x], member[y, z]]]]
```

```
A[x]
```

The class **OMEGA** is the class of all ordinal numbers (as defined by Isbell):

```
class[x, equal[intersection[FULL, P[x]], succ[x]]]
```

```
OMEGA
```

The equipollence relation is: (this takes quite a while!)

```
class[pair[x, y], exists[z, and[ONEONE[z], equal[domain[z], x], equal[range[z], y]]]]
```

```
Q
```

■ Caveats

The results of this notebook have been incorporated in the most recent version of the **GOEDEL** program. To reproduce the calculations in this notebook one would need to use the version **52.L81** which is no longer posted on this website, but it is still available upon request.

Some of the rules added in this notebook are considered to have only temporary intterest, and may not be added to future versions of the **GOEDEL** program.

■ SubstTest results

A basic tool used to derive new results is the substitution test, which compares the results of evaluating an expression in two different orders:

```
?? SubstTest
```

```
Goedel‘Private‘SubstTest

SubstTest[f_, x__, r_] := f @@ ({x} /. r) == (f @@ {x} /. r)
```

Several applications of this test are used here. First:

```
SubstTest[implies, member[u, v], subclass[A[v], u],
    {u -> y, v -> intersection[OMEGA, image[Q, singleton[x]]]}]

or[not[member[x, V]], not[member[y, OMEGA]],
  not[member[pair[x, y], Q]], subclass[card[x], y]] == True

or[not[member[x_, V]], not[member[y_, OMEGA]],
  not[member[pair[x_, y_], Q]], subclass[card[x_], y_]] := True
```

Second:

```
SubstTest[class, pair[x, y], or[not[member[x, V]], not[member[y, u]],
  not[member[pair[x, y], v]], subclass[card[x], y]], {u -> OMEGA, v -> Q}]

cart[V, V] ==
 union[cart[V, complement[OMEGA]], composite[Id, complement[Q]], composite[S, CARD]]
```

Comment: this calculation takes quite a while.

■ Restatements

The final result of the last section can be reformulated in various ways. First:

```
Map[composite[id[OMEGA], complement[#]] &, cart[V, V] ==
  union[cart[V, complement[OMEGA]], composite[Id, complement[Q]], composite[S, CARD]]]

0 == composite[id[OMEGA], intersection[Q, composite[complement[S], CARD]]]

composite[id[OMEGA], intersection[Q, composite[complement[S], CARD]]] := 0
```

Second:

```
SubstTest[equal, 0, composite[id[OMEGA], x],
    x -> intersection[Q, composite[complement[S], CARD]]] // Reverse

subclass[OMEGA,
  complement[fix[composite[Q, inverse[CARD], complement[inverse[S]]]]]] == True

subclass[OMEGA,
  complement[fix[composite[Q, inverse[CARD], complement[inverse[S]]]]]] := True
```

Third:

```
SubstTest[equal, 0, intersection[x, y],
    {x -> composite[id[OMEGA], Q], y -> composite[complement[S], CARD]}] // Reverse

subclass[composite[id[OMEGA], Q], composite[S, CARD]] == True

subclass[composite[id[OMEGA], Q], composite[S, CARD]] := True
```

■ Corollaries

The **GOEDEL** program verifies the truth of the following:

```
equal[intersection[OMEGA, fix[CARD]], fix[CARD]]
```

```
True
```

But it does not have a rule to reduce the left side to the right side:

```
Equal[intersection[OMEGA, fix[CARD]], fix[CARD]]
```

```
intersection[OMEGA, fix[CARD]] == fix[CARD]
```

We may feel free to add such a rule:

```
intersection[OMEGA, fix[CARD]] := fix[CARD]
```

The **Assoc** test used next is an application of the associative law for composition:

```
?? Assoc
```

```
Goedel‘Private‘Assoc
```

```
Assoc[x_, y_, z_] := composite[x, composite[y, z]] == composite[composite[x, y], z]
```

This is one of the more useful tests. Here we use it to deduce:

```
Assoc[id[OMEGA], id[fix[CARD]], CARD]
```

```
composite[id[OMEGA], CARD] == CARD
```

```
composite[id[OMEGA], CARD] := CARD
```

Two more applications of **SubstTest**:

```
SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
    {u -> CARD, v -> Q, w -> Q}]
```

```
subclass[composite[CARD, Q], Q] == True
```

```
subclass[composite[CARD, Q], Q] := True
```

```
SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
{u -> composite[CARD, Q], v -> composite[id[OMEGA], Q], w -> composite[S, CARD]}]
```

```
subclass[composite[CARD, Q], composite[S, CARD]] == True
```

```
subclass[composite[CARD, Q], composite[S, CARD]] := True
```

■ Eliminating S

We now want to eliminate the subclass relation **S** that occurs in the above results.

```
SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
    {u -> composite[CARD, Q], v -> composite[S, CARD], w -> inverse[CARD]}]
```

```
subclass[composite[CARD, Q, inverse[CARD]], S] == True
```

```
subclass[composite[CARD, Q, inverse[CARD]], S] := True
```

The intersection of the subclass relation and its inverse is the identity relation **Id**.

```
SubstTest[subclass, u, intersection[v, w],
    {u -> composite[CARD, Q, inverse[CARD]], v -> S, w -> inverse[S]}]
```

```
subclass[composite[CARD, Q, inverse[CARD]], Id] == True
```

```
subclass[composite[CARD, Q, inverse[CARD]], Id] := True
```

Sharper result:

```
SubstTest[subclass, u, intersection[v, w], {u -> composite[CARD, Q, inverse[CARD]],
v -> Id, w -> cart[fix[CARD], fix[CARD]]}]
```

```
subclass[composite[CARD, Q, inverse[CARD]], id[fix[CARD]]] == True
```

```
subclass[composite[CARD, Q, inverse[CARD]], id[fix[CARD]]] := True
```

■ Equational formulation

The inclusion found above can be sharpened to an equation.

```
SubstTest[implies, subclass[u, v], subclass[composite[x, u, y], composite[x, v, y]],
    {u -> id[fix[CARD]], v -> Q, x -> CARD, y -> inverse[CARD]}]
```

```
subclass[fix[CARD], fix[composite[CARD, Q, inverse[CARD]]]] == True
```

```
subclass[fix[CARD], fix[composite[CARD, Q, inverse[CARD]]]] := True
```

Since the inclusion goes in both directions, equality follows:

```
SubstTest[and, subclass[u, v], subclass[v, u],
    {u -> composite[CARD, Q, inverse[CARD]], v -> id[fix[CARD]]}] // Reverse
```

```
equal[composite[CARD, Q, inverse[CARD]], id[fix[CARD]]] == True
```

```
composite[CARD, Q, inverse[CARD]] := id[fix[CARD]]
```

■ The composite[CARD,Q] formula

The final steps of our derivation just put it all together.

```
SubstTest[implies, subclass[u, v], subclass[composite[x, u], composite[x, v]],
    {u -> id[image[Q, OMEGA]], v -> composite[inverse[CARD], CARD],
        x -> composite[CARD, Q]}]
```

```
subclass[composite[CARD, Q], CARD] == True
```

```
subclass[composite[CARD, Q], CARD] := True
```

Inclusion in the opposite direction is easy because every set is equipollent to itself.

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

```
subclass[CARD, composite[CARD, Q]] == True
```

```
subclass[CARD, composite[CARD, Q]] := True
```

Since the inclusion goes in both directions, equality follows:

```
SubstTest[and, subclass[u, v],
  subclass[v, u], {v -> composite[CARD, Q], u -> CARD}] // Reverse
```

```
equal[CARD, composite[CARD, Q]] == True
```

```
composite[CARD, Q] := CARD
```

This final result will be added to the newer versions of the **GOEDEL** program. Many of the other rules in this notebook are regarded as of lesser interest and will not be added at this time.