Cartesian Squares

Johan G. F. Belinfante
2001 October 20

```
<< goedel52.k87; << tests.m
```

```
:Package Title: GOEDEL52.K87      2001 October 20 at 7:55 a.m.

It is now:  2001 Oct 20 at 11:14

Loading Simplification Rules

TESTS.M                  Revised 2001 October 18

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.
```

■ A temporary rule

```
subclass[cart[singleton[0], singleton[0]], Id] // AssertTest
```

```
subclass[cart[singleton[0], singleton[0]], Id] == True
```

```
subclass[cart[singleton[0], singleton[0]], Id] := True
```

```
SubstTest[implies, subclass[u, v], subclass[image[CART, u], image[CART, v]],
    {u -> id[singleton[0]], v -> Id}]
```

```
member[0, image[CART, Id]] == True
```

We add this as a temporary rule...

```
member[0, image[CART, Id]] := True
```

```
equal[union[image[CART, Id], singleton[0]], image[CART, Id]]
```

```
True
```

This too:

```
union[image[CART, Id], singleton[0]] := image[CART, Id]
```

■ A Normality Test

```
fix[composite[CART, DUP, IMAGE[inverse[DUP]]]] // Normality // Reverse
```

```
intersection[RFX, fix[composite[S, CART, DUP, IMAGE[inverse[DUP]]]]] ==
 fix[composite[CART, DUP, IMAGE[inverse[DUP]]]]
```

```
intersection[RFX, fix[composite[S, CART, DUP, IMAGE[inverse[DUP]]]]] :=
 fix[composite[CART, DUP, IMAGE[inverse[DUP]]]]
```

■ Corollary of the axiom of replacement

We derive a corollary of the axiom of replacement. First we consider the case that **x** is a set:

```
forall[x, implies[member[x, V], member[image[funpart[y], x], V]]] // assert
```

```
True
```

The case that **x** is a proper class is done separately:

```
member[p, V] := False
```

```
implies[member[x, V], member[image[funpart[y], x], V]] /. x -> p
```

```
True
```

This proves:

```
implies[member[x, V], member[image[funpart[y], x], V]] == True
```

```
or[member[image[funpart[y], x], V], not[member[x, V]]] == True
```

This is a permanent new rule:

```
or[member[image[funpart[y_], x_], V], not[member[x_, V]]] := True
```

From this new rule we deduce a corollary:

```
SubstTest[implies, member[x, V], member[image[funpart[y], x], V], y -> inverse[DUP]]
```

```
or[member[fix[x], V], not[member[x, V]]] == True
```

This is another candidate for a permanent rule:

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

```
equiv[and[member[x, V], member[fix[x], V]], member[x, V]] // assert
```

```
True
```

```
and[member[x_, V], member[fix[x_], V]] := member[x, V]
```

■ Two descriptions of the class of cartesian squares

The following inverse image formula is the key to finding a connection between two descriptions of the class of cartesian squares.

```
image[inverse[CART], fix[composite[CART, DUP, IMAGE[inverse[DUP]]]]] // VSNormality
```

```
image[inverse[CART], fix[composite[CART, DUP, IMAGE[inverse[DUP]]]]] ==
 union[Id, cart[V, singleton[0]], cart[singleton[0], V]]
```

We add this as a temporary rule:

```
image[inverse[CART], fix[composite[CART, DUP, IMAGE[inverse[DUP]]]]] :=
 union[Id, cart[V, singleton[0]], cart[singleton[0], V]]
```

The following formula is needed to go further:

```
keyformula :=
 ImageComp[CART, inverse[CART], fix[composite[CART, DUP, IMAGE[inverse[DUP]]]]]
```

First we derive a key membership rule:

```
Map[member[x, #] &, keyformula] // Reverse
```

```
member[x, image[CART, Id]] == and[equal[x, cart[fix[x], fix[x]]], member[x, V]]
```

This membership rule can be made permanent:

```
member[x_, image[CART, Id]] := and[equal[x, cart[fix[x], fix[x]]], member[x, V]]
```

The key formula could be derived from this rule!

```
image[CART, Id] // Normality // Reverse
```

```
fix[composite[CART, DUP, IMAGE[inverse[DUP]]]] == image[CART, Id]
```

```
fix[composite[CART, DUP, IMAGE[inverse[DUP]]]] := image[CART, Id]
```

■ What about our temporary rules?

Let’s remove the temporary rule about the empty set being a cartesian square:

```
member[0, image[CART, Id]] =.
```

It is not needed anymore because of the more general rule that we added.

```
member[0, image[CART, Id]]
```

```
True
```

Another temporary rule can be replaced with a more general result:

```
subclass[cart[singleton[x], singleton[y]], Id] // AssertTest

subclass[cart[singleton[x], singleton[y]], Id] ==
 or[equal[x, y], not[member[x, V]], not[member[y, V]]]

subclass[cart[singleton[x_], singleton[y_]], Id] :=
 or[equal[x, y], not[member[x, V]], not[member[y, V]]]
```