reflexive symmetric relations

Johan G. F. Belinfante
2002 May 3

```
<< goedel52.n46; << tests.m
```

```
:Package Title: GOEDEL52.N46          2002 May 2 at 9:45 p.m.

It is now:  2002 May 3 at 16:1

Loading Simplification Rules

TESTS.M                   Revised 2002 April 16

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.
```

■ Introduction

It is intuitively clear that a relation is reflexive and symmetric if and only if it is a union of cartesian squares. This can be expressed as the following formula, which is derived in this notebook.

```
Uclosure[image[CART, Id]] == intersection[RFX, SYM];
```

An attempted derivation of this formula in a notebook dated 2001 October 28 had failed, but one of its ideas, namely the use of the function called **BCD** is reused below. In addition, we use the following formula which had been derived at that time:

```
image[inverse[PAIRSET], cliques[x]]
```

```
composite[id[fix[x]], intersection[x, inverse[x]], id[fix[x]]]
```

For convenience we introduce the following temporary abbreviation for cartesian squares:

```
cartsq[x_] := cart[x, x]
```

We note that the class of all cartesian squares is:

```
class[y, exists[x, equal[y, cartsq[x]]]]
```

```
image[CART, Id]
```

■ Review

The classes **RFX** and **SYM** of reflexive and symmetric relations, respectively, can be characterized in various ways; the following is especially succinct:

```
class[x, subclass[x, cartsq[fix[x]]]]
```

```
RFX
```

```
class[x, subclass[x, inverse[x]]]
```

```
SYM
```

The set of cliques of a relation **x** is:

```
class[y, subclass[cartsq[y], x]]
```

```
cliques[x]
```

The functions **CART**, **DUP**, and **PAIRSET** are characterized as:

```
lambda[pair[x, y], cart[x, y]]
```

```
CART
```

```
lambda[x, pair[x, x]]
```

```
DUP
```

```
lambda[pair[x, y], pairset[x, y]]
```

```
PAIRSET
```

The function **CORE[x]** is defined by

```
lambda[y, U[intersection[x, P[y]]]]
```

```
CORE[x]
```

The **Uclosure** of a class **x** is the set of all unions of subsets of **x**:

```
class[y, exists[z, and[equal[y, U[z]], subclass[z, x]]]]
```

```
Uclosure[x]
```

These last two concepts have been intensively studied to prepare for topology, but they are also independently useful.

■ Half the battle is over...

The inclusion in one direction is easily disposed of:

```
SubstTest[implies, subclass[u, v], subclass[Uclosure[u], Uclosure[v]],
    {u -> image[CART, Id], v -> intersection[RFX, SYM]}]

and[subclass[Uclosure[image[CART, Id]], RFX],
  subclass[Uclosure[image[CART, Id]], SYM]] == True
```

This justifies adding two temporary rules:

```
subclass[Uclosure[image[CART, Id]], RFX] := True

subclass[Uclosure[image[CART, Id]], SYM] := True
```

■ some new ideas

The following fascinating connection holds between pairsets and cartesian squares:

```
composite[S, PAIRSET] // inverse

composite[inverse[e], CART, DUP]
```

This is what is behind the following new formula

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

subclass[inverse[PAIRSET], composite[inverse[e], CART, DUP]] == True

subclass[inverse[PAIRSET], composite[inverse[E], CART, DUP]] := True
```

We now use the old formula for **image[inverse[PAIRSET],cliques[x]]** mentioned in the introduction to this notebook

```
SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
    {u -> inverse[PAIRSET], v -> composite[inverse[E], CART, DUP],
        w -> cliques[x]}]

subclass[composite[id[fix[x]], intersection[x, inverse[x]], id[fix[x]]],
  composite[inverse[e], id[cliques[x]], e]] == True

subclass[composite[id[fix[x_]], intersection[x_, inverse[x_]], id[fix[x_]]],
  composite[inverse[E], id[cliques[x_]], E]] := True
```

■ The function BCD

The temporary function **BCD** is:

```
lambda[x, U[image[CART, id[x]]]]

composite[BIGCUP, IMAGE[CART], IMAGE[DUP]]

BCD = composite[BIGCUP, IMAGE[CART], IMAGE[DUP]];
```

We derive one property of this function:

```
Map[VERTSECT[inverse[#]] &,
 Assoc[inverse[CLIQUES], E, inverse[composite[inverse[E], CART, DUP]]]]
```

```
composite[BIGCUP, IMAGE[CART], IMAGE[DUP], CLIQUES] == CORE[image[CART, Id]]
```

```
composite[BIGCUP, IMAGE[CART], IMAGE[DUP], CLIQUES] := CORE[image[CART, Id]]
```

This last rule can be rewritten more concisely as follows:

```
composite[BCD, CLIQUES]
```

```
CORE[image[CART, Id]]
```

■ An unused formula

Here is another formula for the function **BCD**:

```
Map[VERTSECT, Assoc[inverse[PAIRSET], inverse[S], inverse[E]]]
```

```
composite[IMAGE[inverse[PAIRSET]], IMAGE[inverse[S]]] ==
 composite[BIGCUP, IMAGE[CART], IMAGE[DUP]]
```

Thia formula is not needed here, nor is it clear how this should be oriented if one would wish to add it as a new simplification rule, so we do not do so for now.

■ A key inclusion

We are now ready to derive a key inclusion.

```
SubstTest[implies, and[subclass[x, y], subclass[y, z]], subclass[x, z],
    {y -> composite[id[fix[x]], intersection[x, inverse[x]], id[fix[x]]],
  z -> composite[inverse[E], id[cliques[x]], E]}]
```

```
or[not[subclass[x, cart[fix[x], fix[x]]]], not[subclass[x, inverse[x]]],
  subclass[x, composite[inverse[e], id[cliques[x]], e]]] == True
```

```
Map[class[x, #] &, %] // InvertFix
```

```
union[complement[RFX], complement[SYM], fix[CORE[image[CART, Id]]]] == V
```

```
Map[equal[V, #] &, %]
```

```
subclass[intersection[RFX, SYM], fix[CORE[image[CART, Id]]]] == True
```

```
subclass[intersection[RFX, SYM], fix[CORE[image[CART, Id]]]] := True
```

■ final steps

For the three quantities **Uclosure[image[CART,Id]]**, **fix[CORE[image[CART,Id]]]** and **intersection[RFX,SYM]**, we now have inclusions going around in a circle, and so all three are equal.

```
SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
    {u -> Uclosure[image[CART, Id]], v -> intersection[RFX, SYM],
        w -> fix[CORE[image[CART, Id]]]}]
```

```
subclass[Uclosure[image[CART, Id]], fix[CORE[image[CART, Id]]]] == True
```

```
subclass[Uclosure[image[CART, Id]], fix[CORE[image[CART, Id]]]] := True
```

```
SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
    {u -> fix[CORE[image[CART, Id]]],
  v -> Uclosure[image[CART, Id]], w -> intersection[RFX, SYM]}]
```

```
and[subclass[fix[CORE[image[CART, Id]]], RFX],
  subclass[fix[CORE[image[CART, Id]]], SYM]] == True
```

```
subclass[fix[CORE[image[CART, Id]]], RFX] := True
```

```
subclass[fix[CORE[image[CART, Id]]], SYM] := True
```

```
SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
    {v -> fix[CORE[image[CART, Id]]],
  w -> Uclosure[image[CART, Id]], u -> intersection[RFX, SYM]}]
```

```
subclass[intersection[RFX, SYM], Uclosure[image[CART, Id]]] == True
```

```
subclass[intersection[RFX, SYM], Uclosure[image[CART, Id]]] := True
```

■ The promised formula, and another.

At this point we are ready to derive the formula promised in the introduction.

```
SubstTest[and, subclass[u, v], subclass[v, u],
    {u -> Uclosure[image[CART, Id]], v -> intersection[RFX, SYM]}] // Reverse
```

```
equal[intersection[RFX, SYM], Uclosure[image[CART, Id]]] == True
```

```
Uclosure[image[CART, Id]] := intersection[RFX, SYM]
```

As a bonus, we found a second interesting formula:

```
SubstTest[and, subclass[u, v], subclass[v, u],
    {u -> fix[CORE[image[CART, Id]]], v -> intersection[RFX, SYM]}] // Reverse
```

```
equal[fix[CORE[image[CART, Id]]], intersection[RFX, SYM]] == True
```

```
fix[CORE[image[CART, Id]]] := intersection[RFX, SYM]
```

■ Generalizations...

Some formulas obtained in this notebook can be generalized. For example:

```
ImageComp[BCD, inverse[BCD], P[x]] // Reverse
```

```
Uclosure[image[CART, id[cliques[x]]]] == intersection[RFX, SYM, P[x]]
```

```
ImageComp[BCD, inverse[BCD], P[cart[x, x]]] // Reverse
```

```
Uclosure[image[CART, id[P[x]]]] == intersection[RFX, SYM, P[cart[x, x]]]
```

```
Uclosure[image[CART, id[P[x_]]]] := intersection[RFX, SYM, P[cart[x, x]]]
```