Aclosures of classes of cartesian squares

Johan G. F. Belinfante
2003 May 18

```
In[1]:= << goedel52.r75; << tools.m

        :Package Title: goedel52.r75         2003 May 18 at 1:05 p.m.

        It is now:  2003 May 22 at 12:34

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 May 21

        weightlimit = 40
```

■ Aclosures of classes of squares

The simplest case is the class of all squares:

```
In[2]:= ImageComp[BIGCAP, IMAGE[CART], P[Id]] // Reverse

Out[2]= Aclosure[image[CART, Id]] == image[CART, Id]

In[3]:= Aclosure[image[CART, Id]] := image[CART, Id]
```

In this notebook a generalization of this formula is derived which holds for any class of squares.

■ the derivation

The basic fact needed is this:

```
In[4]:= image[IMAGE[DUP], P[x]]

Out[4]= P[id[x]]
```

This implies a similar formula involving **DORA**, the function that takes **x** to **pair[domain[x],range[x]]**.

```
In[5]:= ImageComp[DORA, IMAGE[DUP], P[x]] // Reverse

Out[5]= image[DORA, P[id[x]]] == id[P[x]]

In[6]:= image[DORA, P[id[x_]]] := id[P[x]]
```

From this result one immediately obtains the promised **Aclosure** formula:

```
In[7]:= ImageComp[BIGCAP, IMAGE[CART], P[id[x]]] // Reverse

Out[7]= Aclosure[image[CART, id[x]]] == image[CART, id[Aclosure[x]]]
```

```
In[8]:= Aclosure[image[CART, id[x_]]] := image[CART, id[Aclosure[x]]]
```

■ a version without variables

A variable−free version can be obtained using **reify**:

```
In[9]:= Map[VERTSECT, SubstTest[reify, x, Aclosure[image[CART, F[x]]], F -> id]] // Reverse

Out[9]= composite[ACLOSURE, IMAGE[CART], IMAGE[DUP]] ==
         composite[IMAGE[CART], IMAGE[DUP], ACLOSURE]

In[10]:= composite[ACLOSURE, IMAGE[CART], IMAGE[DUP]] :=
          composite[IMAGE[CART], IMAGE[DUP], ACLOSURE]
```

This result says that **ACLOSURE** commutes with the function **composite[IMAGE[CART],IMAGE[DUP]]**.

```
In[11]:= commute[ACLOSURE, composite[IMAGE[CART], IMAGE[DUP]]]

Out[11]= True
```

■ some related results

The **GOEDEL** program already contains the following rule:

```
In[12]:= image[IMAGE[inverse[DUP]], P[id[x]]]

Out[12]= P[x]
```

There are analogs of this with **inverse[DUP]** replaced by **FIRST** and **SECOND**:

```
In[13]:= ImageComp[IMAGE[FIRST], IMAGE[DUP], P[x]] // Reverse

Out[13]= image[IMAGE[FIRST], P[id[x]]] == P[x]

In[14]:= image[IMAGE[FIRST], P[id[x_]]] := P[x]

In[15]:= ImageComp[IMAGE[SECOND], IMAGE[DUP], P[x]] // Reverse

Out[15]= image[IMAGE[SECOND], P[id[x]]] == P[x]

In[16]:= image[IMAGE[SECOND], P[id[x_]]] := P[x]
```