The function RCF

Johan G. F. Belinfante
2002 February 4

```
<< goedel52.m82; << tests.m

:Package Title: GOEDEL52.M82      2002 February 4 at 1:30 a.m.

It is now:  2002 Feb 4 at 15:47

Loading Simplification Rules

TESTS.M                   Revised 2002 February 3

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

A new function **RCF** has been added to the **GOEDEL** program. The membership rule used to define this function is:

```
member[x, RCF]

and[equal[RC[first[x]], second[x]], member[first[x], V]]
```

All other properties of this function have been deduced from this membership rule, using the **GOEDEL** program itself. In this notebook we summarize a few of these properties.

■ Review: the function RC[x].

The function **RC[x]** introduced earlier performs relative complements with respect to **x**:

```
FUNCTION[RC[x]]

True
```

The domain of **RC[x]** is the power set **P[x]** provided that **x** is a set.

```
domain[RC[x]]

intersection[image[V, singleton[x]], P[x]]
```

A nice characterization of **RC[x]** is:

```
class[pair[u, v], and[disjoint[u, v], equal[union[u, v], x]]]
```

```
RC[x]
```

This function is its own inverse:

```
inverse[RC[x]]
```

```
RC[x]
```

For any subset of x, a second application of relative complementation takes one back.

```
composite[RC[x], RC[x]]
```

```
id[intersection[image[V, singleton[x]], P[x]]]
```

Sine the complement of a set is a proper class, the function **RC[x]** is the empty set when **x** is not a set.

```
RC[V]
```

```
0
```

Many formulas for **RC[x]** involve the class **image[V,singleton[x]]** which is either **V** or **0** depending on whether **x** is a set.

```
equal[V, image[V, singleton[x]]]
```

```
member[x, V]
```

```
equal[0, image[V, singleton[x]]]
```

```
not[member[x, V]]
```

■ Characterizations of RCF

The function **RCF** takes **x** to **RC[x]**.

```
lambda[x, RC[x]]
```

```
RCF
```

For any class **x**, the function **RC[x]** is a set.

```
member[RC[x], V]
```

```
True
```

Consequently, the domain of **RCF** is the class **V** of all sets:

```
domain[RCF]
```

```
V
```

Each **RC[x]** function is a bijection:

```
member[RC[x], BIJ]
```

```
True
```

The range of **RCF** is the class of all **RC[x]** functions, and therefore:

```
subclass[range[RCF], BIJ]
```

```
True
```

A point **pair[u,v]** belongs to some **RC[x]** function provided **u** and **v** are disjoint.

```
U[range[RCF]]
```

```
DISJOINT
```

The relation **DISJOINT** ia:

```
class[pair[x, y], equal[0, intersection[x, y]]]
```

```
DISJOINT
```

■ Some useful observations.

Many of the rules for **RCF** were obtained from this description of it:

```
composite[IMAGE[id[DISJOINT]], VERTSECT[inverse[CUP]]]
```

```
RCF
```

Here **CUP** is the function

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

```
CUP
```

The functions **VERTSECT** and **IMAGE** are in general defined by

```
lambda[y, image[x, singleton[y]]]
```

```
VERTSECT[x]
```

```
lambda[y, image[x, y]]
```

```
IMAGE[x]
```

In particular, the function **IMAGE[id[x]]** is

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

```
IMAGE[id[x]]
```

We call a relation thin if all its vertical sections are sets:

```
thin[x]
```

```
equal[V, domain[VERTSECT[x]]]
```

A key fact is that the inverse of **CUP** is a thin relation:

```
thin[inverse[CUP]]
```

```
True
```

The following indirect description avoids both **VERTSECT** and **IMAGE**, but gives only the composite with the inverse of the membership relation **E**.

```
composite[id[DISJOINT], inverse[CUP]] == composite[inverse[E], RCF]
```

```
True
```

The membership relation **E** is characterized by:

```
member[pair[x, y], E]
```

```
and[member[x, y], member[y, V]]
```