RC[x] as a binary homomorphism

Johan G. F. Belinfante
2011 November 17

In[1]:= SetDirectory["1:"]; << goedel.11nov16a

:Package Title: goedel.11nov16a 2011 November 16 at 8:30 p.m.

Loading takes about thirteen minutes, half that time due to builtin pauses.

It is now: 2011 Nov 17 at 12:35

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40

Loading completed.

It is now: 2011 Nov 17 at 12:48

summary

If x is a set, then the function RC[x] is an isomorphism between the restrictions of CAP and CUP to the cartesian square of the power set P[x]. Each of these restrictions is a monoid.

In[2]:= member[composite[id[P[x]], CUP], MONOIDS]

Out[2]= member[x, V]

In[3]:= member[composite[CAP, id[cart[P[x], P[x]]]], MONOIDS]

Out[3]= member[x, V]

Both if these are groups only in the trivial case that x is empty.

In[4]:= member[composite[id[P[x]], CUP], GROUPS]

Out[4]= equal[0, x]

In[5]:= member[composite[CAP, id[cart[P[x], P[x]]]], GROUPS]

Out[5]= equal[0, x]
derivation

In one direction, the use of AssertTest suffices.

Theorem.

```
In[6]:=  member[RC[x], binhom[composite[id[P[x]], CUP], CAP]] // AssertTest
Out[6]=  member[RC[x], binhom[composite[id[P[x]], CUP], CAP]] := member[x, V]
```

```
In[7]:=  member[RC[x_], binhom[composite[id[P[x_]], CUP], CAP]] := member[x, V]
```

Theorem.

```
In[8]:=  member[RC[x], binhom[composite[id[P[x]], CUP], composite[CAP, id[cartsq[P[x]]]]]] // AssertTest
Out[8]=  member[RC[x], binhom[composite[id[P[x]], CUP], composite[CAP, id[cart[P[x], P[x]]]]]] := member[x, V]
```

```
In[9]:=  member[RC[x_], binhom[composite[id[P[x_]], CUP],
composite[CAP, id[cart[P[x_], P[x_]]]]]] := member[x, V]
```

In the opposite direction, some lemmas will be required.

Lemma. Simplification rule.

```
In[10]:=  equal[composite[id[P[x]], CAP, cross[RC[x], RC[x]]],
composite[CAP, cross[RC[x], RC[x]]]]
Out[10]=  True
```

```
In[11]:=  composite[id[P[x_]], CAP, cross[RC[x_], RC[x_]]] := composite[CAP, cross[RC[x], RC[x]]]
```

Theorem. Simplification rule.

```
In[12]:=  member[RC[x], map[P[x], V]] // AssertTest
Out[12]=  member[RC[x], map[P[x], V]] := member[x, V]
```

```
In[13]:=  member[RC[x_], map[P[x_], V]] := member[x, V]
```

Lemma. Simplification rule.

```
In[14]:=  composite[CUP, cross[RC[x], RC[x]]] // FastReifTriNormality// Reverse
Out[14]=  composite[id[image[V, set[x_]]], CUP, cross[RC[x], RC[x]]] :=
composite[CUP, cross[RC[x], RC[x]]]
```

```
In[15]:=  composite[id[image[V, set[x_]]], CUP, cross[RC[x], RC[x]]] :=
composite[CUP, cross[RC[x], RC[x]]]
```
Theorem.

\[ \text{In[16]} : = \text{Map}[\text{composite}[\#, \text{id}[\text{cart}[P[x], P[x]]]] \&, \text{composite}[RC[x], \text{CAP}]] \text{\textbar FastReifTriNormality} \]

\[ \text{Out[16]} = \text{composite}[RC[x], \text{CAP}, \text{id}[\text{cart}[P[x], P[x]]]] = \text{composite}[\text{CUP}, \text{cross}[RC[x], RC[x]]] \]

\[ \text{In[17]} : = \text{composite}[RC[x_], \text{CAP}, \text{id}[\text{cart}[P[x_], P[x_]]]] = \text{composite}[\text{CUP}, \text{cross}[RC[x], RC[x]]] \]

Theorem.

\[ \text{In[18]} : = \text{member}[RC[x], \text{binhom}[\text{composite}[\text{CAP}, \text{id}[\text{cart}[P[x], P[x]]]], \text{CUP}]] \text{\textbar AssertTest} \]

\[ \text{Out[18]} = \text{member}[RC[x], \text{binhom}[\text{composite}[\text{CAP}, \text{id}[\text{cart}[P[x], P[x]]]], \text{CUP}]] = \text{member}[x, V] \]

\[ \text{In[19]} : = \text{member}[RC[x_], \text{binhom}[\text{composite}[\text{CAP}, \text{id}[\text{cart}[P[x_], P[x_]]]], \text{CUP}]] = \text{member}[x, V] \]

Theorem.

\[ \text{In[20]} : = \text{member}[RC[x], \text{binhom}[\text{composite}[\text{CAP}, \text{id}[\text{cart}[P[x], P[x]]]], \text{composite}[\text{id}[P[x]], \text{CUP}]]] \text{\textbar AssertTest} \]

\[ \text{Out[20]} = \text{member}[RC[x], \text{binhom}[\text{composite}[\text{CAP}, \text{id}[\text{cart}[P[x], P[x]]]], \text{composite}[\text{id}[P[x]], \text{CUP}]]] = \text{member}[x, V] \]

\[ \text{In[21]} : = \text{member}[RC[x_], \text{binhom}[\text{composite}[\text{CAP}, \text{id}[\text{cart}[P[x_], P[x_]]]], \text{composite}[\text{id}[P[x_]], \text{CUP}]]] = \text{member}[x, V] \]

functor rules

Theorem.

\[ \text{In[23]} : = \text{functor}[RC[x], \text{composite}[\text{id}[P[x]], \text{CUP}], \text{composite}[\text{CAP}, \text{id}[\text{cart}[P[x], P[x]]]]] \text{\textbar AssertTest} \]

\[ \text{Out[23]} = \text{functor}[RC[x], \text{composite}[\text{id}[P[x]], \text{CUP}], \text{composite}[\text{CAP}, \text{id}[\text{cart}[P[x], P[x]]]]] = \text{member}[x, V] \]

\[ \text{In[24]} : = \text{functor}[RC[x_], \text{composite}[\text{id}[P[x_]], \text{CUP}], \text{composite}[\text{CAP}, \text{id}[\text{cart}[P[x_], P[x_]]]]] = \text{member}[x, V] \]

Theorem.

\[ \text{In[25]} : = \text{functor}[RC[x], \text{composite}[\text{CAP}, \text{id}[\text{cart}[P[x], P[x]]]], \text{composite}[\text{id}[P[x]], \text{CUP}]] \text{\textbar AssertTest} \]

\[ \text{Out[25]} = \text{functor}[RC[x], \text{composite}[\text{CAP}, \text{id}[\text{cart}[P[x], P[x]]]], \text{composite}[\text{id}[P[x]], \text{CUP}]] = \text{member}[x, V] \]

\[ \text{In[26]} : = \text{functor}[RC[x_], \text{composite}[\text{CAP}, \text{id}[\text{cart}[P[x_], P[x_]]]], \text{composite}[\text{id}[P[x_]], \text{CUP}]] = \text{member}[x, V] \]