CAP ∘ (S ⊗ S)

Johan G. F. Belinfante
2009 May 12

```
In[1]:= SetDirectory["l:"]; << goedel.09may11a; << tools.m

        :Package Title: goedel.09may11a            2009 May 11 at 4:25 p.m.

        It is now:  2009 May 12 at 15:26

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 April 6

        weightlimit = 40
```

summary

A formula for **composite[CAP, cross[S,S]]** is derived. The key idea is that if a set **w** contains the intersection of two sets **u** and **v**, then **w** is the intersection of a set that contains **u** and a set that contains **v**. This is an immediate consequence of the distributive law:

```
In[2]:= equal[w, intersection[union[w, u], union[w, v]]]

Out[2]= subclass[intersection[u, v], w]
```

A key ingredient of the derivation presented in this notebook is the following variable-free formulation of the distributive law:

```
In[3]:= composite[CAP, cross[CUP, CUP], TWIST, cross[Id, DUP]]

Out[3]= composite[CUP, cross[CAP, Id]]
```

derivation

Lemma. Simplification rule.

```
In[4]:= composite[ADJOIN[y], inverse[LEFT[x]], inverse[CUP]] // DoubleInverse

Out[4]= composite[ADJOIN[y], inverse[LEFT[x]], inverse[CUP]] ==
         composite[ADJOIN[y], inverse[ADJOIN[x]]]

In[5]:= composite[ADJOIN[y_], inverse[LEFT[x_]], inverse[CUP]] :=
         composite[ADJOIN[y], inverse[ADJOIN[x]]]
```

Theorem. A corollary of the distributive law.

```
In[6]:= ImageComp[composite[CAP, cross[CUP, CUP]],
          composite[TWIST, cross[Id, DUP]], cart[cart[set[y], set[x]], V]] // Reverse
```

```
Out[6]= image[CAP, composite[ADJOIN[x], inverse[ADJOIN[y]]]] ==
         intersection[image[S, set[intersection[x, y]]], image[V, set[x]], image[V, set[y]]]
```

```
In[7]:= image[CAP, composite[ADJOIN[x_], inverse[ADJOIN[y_]]]] :=
         intersection[image[S, set[intersection[x, y]]], image[V, set[x]], image[V, set[y]]]
```

Corollary.

```
In[8]:= (SubstTest[implies, subclass[u, v], subclass[image[CAP, u], image[CAP, v]],
           {u -> composite[ADJOIN[t], inverse[ADJOIN[s]]],
            v -> cart[image[S, set[s]], image[S, set[t]]]}] //
          Reverse) /. {s → setpart[x], t → setpart[y]}
```

```
Out[8]= subclass[image[S, set[intersection[setpart[x], setpart[y]]]],
          image[CAP, cart[image[S, set[setpart[x]]], image[S, set[setpart[y]]]]]] == True
```

```
In[9]:= (% /. {x → x_, y → y_}) /. Equal→ SetDelayed
```

The inclusion in this corollary can be sharpened to an equation.

Theorem.

```
In[10]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[S, set[intersection[setpart[x], setpart[y]]]],
           v -> image[CAP, cart[image[S, set[setpart[x]]], image[S, set[setpart[y]]]]]}]
```

```
Out[10]= equal[image[CAP, cart[image[S, set[setpart[x]]], image[S, set[setpart[y]]]],
           image[S, set[intersection[setpart[x], setpart[y]]]]] == True
```

```
In[11]:= image[CAP, cart[image[S, set[setpart[x_]]], image[S, set[setpart[y_]]]]] :=
          image[S, set[intersection[setpart[x], setpart[y]]]]
```

The final step is to eliminate the variables **x** and **y** by using **reify**. Note that both variables can be removed at once by replacing them with **first[x]** and **second[x]**.

Theorem. A formula for **CAP ∘ (S ⊗ S)**.

```
In[12]:= Map[composite[#, id[cart[V, V]]] &,
          SubstTest[reify, x, image[t, cart[image[S, set[setpart[first[x]]]],
             image[S, set[setpart[second[x]]]]], t → CAP]]
```

```
Out[12]= composite[CAP, cross[S, S]] == composite[S, CAP]
```

```
In[13]:= composite[CAP, cross[S, S]] := composite[S, CAP]
```

One of the formulas derived above with **setpart** wrappers can be replaced with a more general wrapper-free rewrite rule.

Corollary.

In[14]:= **ImageComp[CAP, cross[S, S], cart[x, y]] // Reverse**

Out[14]= image[CAP, cart[image[S, x], image[S, y]]] == image[S, image[CAP, cart[x, y]]]

In[15]:= **image[CAP, cart[image[S, x_], image[S, y_]]] := image[S, image[CAP, cart[x, y]]]**