subcommutants

Johan G. F. Belinfante and Tiffany D. Goble
2004 June 24

```
In[1]:= SetDirectory["p:"]; << goedel58.24a; << tools.m

        :Package Title: goedel58.24a    2004 June 24 at midnite

        It is now:  2004 Jun 24 at 16:19

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 June 22

        weightlimit = 40
```

summary

The general theory of transvarance is applied to the special case of subcommutants in this notebook. Recall the definitions:

```
In[2]:= transvariant[x, y, z]

Out[2]= subclass[image[x, z], image[y, z]]

In[3]:= class[z, transvariant[x, y, z]]

Out[3]= transvar[x, y]
```

For the special case of subcommutants, one has these definitions:

```
In[4]:= subcommute[x, y]

Out[4]= subclass[composite[x, y], composite[y, x]]

In[5]:= class[y, subcommute[x, y]]

Out[5]= subcommutant[x]
```

The connections between these concepts is:

```
In[6]:= transvariant[cross[Id, x], cross[inverse[x], Id], y] == subcommute[x, y]

Out[6]= True

In[7]:= transvar[cross[Id, x], cross[inverse[x], Id]]

Out[7]= subcommutant[x]
```

subcommutant is closed under composition

Lemma.

```
In[8]:= or[not[member[pair[x, y], cart[subcommutant[z], subcommutant[z]]]],
          member[composite[x, y], subcommutant[z]]] // NotNotTest

Out[8]= or[and[member[composite[x, y], V],
            subclass[composite[z, x, y], composite[x, y, z]]], not[member[x, V]],
          not[member[y, V]], not[subclass[composite[z, x], composite[x, z]]],
          not[subclass[composite[z, y], composite[y, z]]]] == True

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

This result will not be made permanent, due to its complexity, but instead a version of it with fewer variables will be derived. The set variables in the above expression are eliminated in a standard fashion:

```
In[10]:= Map[equal[0, composite[Id, complement[#]]] &,
           SubstTest[class, pair[u, v], or[not[member[pair[u, v], cart[w, w]]],
             member[composite[u, v], w]], w → subcommutant[x]]] // Reverse

Out[10]= subclass[image[COMPOSE, cart[subcommutant[x], subcommutant[x]]],
           subcommutant[x]] == True
```

This corollary is worth making permanent:

```
In[11]:= subclass[image[COMPOSE, cart[subcommutant[x_], subcommutant[x_]]],
           subcommutant[x_]] := True
```

The following corollary is obtained from this:

```
In[12]:= SubstTest[implies, subclass[u, v], subclass[U[u], U[v]],
          {u → image[COMPOSE, cart[subcommutant[x], subcommutant[x]]],
           v → subcommutant[x]}]

Out[12]= TRANSITIVE[composite[Id, U[subcommutant[x]]]] == True

In[13]:= TRANSITIVE[composite[Id, U[subcommutant[x_]]]] == True

Out[13]= TRANSITIVE[composite[Id, U[subcommutant[x_]]]] == True
```

subcommutant properties

The statement that **x** subcommutes with **y** is invariant under replacing **y** with **composite[Id, y]**. And similarly for **x**.

```
In[14]:= subcommute[x, composite[Id, y]] == subcommute[x, y]

Out[14]= True
```

A formula for **subcommutant[x]** can be derived from this by eliminating the variable **y**. A quick way to do this uses **Normality**.

```
In[15]:= image[inverse[IMAGE[id[cart[V, V]]]], subcommutant[x]] // Normality

Out[15]= image[inverse[IMAGE[id[cart[V, V]]]], subcommutant[x]] == subcommutant[x]

In[16]:= image[inverse[IMAGE[id[cart[V, V]]]], subcommutant[x_]] := subcommutant[x]
```

Corollary:

```
In[17]:= ImageComp[IMAGE[id[cart[V, V]]],
           inverse[IMAGE[id[cart[V, V]]]], subcommutant[x]] // Reverse

Out[17]= image[IMAGE[id[cart[V, V]]], subcommutant[x]] ==
          intersection[P[cart[V, V]], subcommutant[x]]

In[18]:= image[IMAGE[id[cart[V, V]]], subcommutant[x_]] :=
          intersection[P[cart[V, V]], subcommutant[x]]
```

some observations regarding set-hood

Since any non-relation subcommutes with any **x**, it follows that **subcommutant[x]** is always a proper class.

```
In[19]:= subclass[P[complement[cart[V, V]]], subcommutant[x]]

Out[19]= True

In[20]:= Map[not, SubstTest[implies, and[subclass[u, v], member[v, V]], member[u, V],
           {u -> P[complement[cart[V, V]]], v -> subcommutant[x]}]]

Out[20]= member[subcommutant[x], V] == False

In[21]:= (% /. x → x_) /. Equal → SetDelayed
```

```
In[22]:= member[subcommutant[x], y] // AssertTest

Out[22]= member[subcommutant[x], y] == False

In[23]:= member[subcommutant[x_], y_] := False
```

The same holds for the sum class:

```
In[24]:= member[U[subcommutant[x]], y] // AssertTest

Out[24]= member[U[subcommutant[x]], y] == False

In[25]:= member[U[subcommutant[x_]], y_] := False
```

Here is a fairly typical example:

```
In[26]:= U[subcommutant[cart[V, singleton[0]]]] // Normality

Out[26]= U[subcommutant[cart[V, singleton[0]]]] == V
```

---

fix[U[subcommutant[x]]]

In this section a special formula is derived from which it follows that **composite[Id, U[subcommutant[x]]]** is a proper class when **x** is thin. For that one needs results about the **domain** and **range**.

```
In[27]:= member[composite[Id, z], V]

Out[27]= and[member[domain[z], V], member[range[z], V]]
```

It will suffice to look at the fixed point set because of the following obsevation:

```
In[28]:= Map[implies[equal[V, fix[x]], #] &, SubstTest[and,
            equal[V, u], subclass[u, v], {u → fix[x], v → domain[x]}]] // Reverse

Out[28]= or[equal[V, domain[x]], not[equal[V, fix[x]]]] == True

In[29]:= or[equal[V, domain[x_]], not[equal[V, fix[x_]]]] := True
```

A similar result holds for the **range**.

```
In[30]:= SubstTest[implies, equal[V, fix[y]], equal[V, domain[y]], y → inverse[x]]

Out[30]= or[equal[V, range[x]], not[equal[V, fix[x]]]] == True

In[31]:= or[equal[V, range[x_]], not[equal[V, fix[x_]]]] := True
```

The basic obervation is that **x** subcommutes with **id[y]** if and only if **y** is invariant under **x**.

```
In[32]:= subcommute[x, id[y]] == invariant[x, y]

Out[32]= True
```

Eliminating the variable **y** yields:

```
In[33]:= image[inverse[IDP], subcommutant[x]] // Normality

Out[33]= image[inverse[IMAGE[DUP]], subcommutant[x]] == invar[x]

In[34]:= image[inverse[IMAGE[DUP]], subcommutant[x_]] := invar[x]

In[35]:= Map[subclass[U[#], U[subcommutant[x]]] &,
           ImageComp[IDP, inverse[IDP], subcommutant[x]]] // Reverse

Out[35]= subclass[domain[VERTSECT[trv[x]]], fix[U[subcommutant[x]]]] == True

In[36]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[37]:= Map[implies[thin[x], #] &, SubstTest[and, equal[V, u], subclass[u, v],
            {u -> domain[VERTSECT[trv[x]]], v -> fix[U[subcommutant[x]]]}]] // Reverse

Out[37]= or[equal[V, fix[U[subcommutant[x]]]],
           not[equal[V, domain[VERTSECT[x]]]]] == True

In[38]:= or[equal[V, fix[U[subcommutant[x_]]]],
           not[equal[V, domain[VERTSECT[x_]]]]] := True
```

This looks simpler when the thin-ness hypothesis is replaced with a **thinpart** wrapper:

```
In[39]:= SubstTest[implies, thin[y],
          equal[V, fix[U[subcommutant[y]]]], y → thinpart[x]]

Out[39]= equal[V, fix[U[subcommutant[thinpart[x]]]]] == True

In[40]:= fix[U[subcommutant[thinpart[x_]]]] := V
```

Corollary:

```
In[41]:= SubstTest[implies, equal[V, fix[y]],
          equal[V, domain[y]], y -> U[subcommutant[thinpart[x]]]]

Out[41]= equal[V, domain[U[subcommutant[thinpart[x]]]]] == True

In[42]:= domain[U[subcommutant[thinpart[x_]]]] := V
```

It follows that if **x** is a thin relation, the relation **composite[Id, U[subcommutant[x]]]** is a proper class.

some examples

An important example is the relation **ZN** whose vertical sections at ordinals are the levels of the Zermelo-von Neumann cumulative hierarchy.

```
In[43]:= composite[Id, U[subcommutant[inverse[E]]]]

Out[43]= ZN
```

Since **inverse[E]** is thin, the relation **ZN** is a proper class.

```
In[44]:= member[ZN, x]

Out[44]= False
```

When x is not thin, the relation **composite[Id, U[subcommutant[x]]]** need not be a proper class. For example:

```
In[45]:= composite[Id, U[subcommutant[cart[V, V]]]]

Out[45]= 0
```

closure under arbitrary unions

The class **subcommutant[x]** is closed under arbitrary unions.

```
In[46]:= Uclosure[subcommutant[x]]

Out[46]= subcommutant[x]
```

A slightly stronger result holds; this is derived a special case of a more general result about **transvar**:

```
In[47]:= SubstTest[implies, subclass[x, transvar[u, v]], transvariant[u, v, U[x]],
          {u → cross[Id, y], v → cross[inverse[y], Id]}]

Out[47]= or[not[subclass[x, subcommutant[y]]],
           subclass[composite[y, U[x]], composite[U[x], y]]] == True

In[48]:= or[not[subclass[x_, subcommutant[y_]]],
           subclass[composite[y_, U[x_]], composite[U[x_], y_]]] := True
```