commuting reflexive relations

Johan G. F. Belinfante
2009 February 20

```
In[1]:= << l:goedel.09feb18a;<< l:tools.m

        :Package Title: goedel.09feb18a        2009 February 18 at 2:35 p.m.

        It is now:  2009 Feb 20 at 15:22

        Loading Simplification Rules

        TOOLS.M                                        Revised 2009 February 18

        weightlimit = 40
```

summary

The composite of commuting reflexive relations is a reflexive relation. In this notebook a variable-free statement of this fact is derived using **reify**.

derivation

Lemma. Deriving an inclusion in one direction is easy.

```
In[2]:= Map[empty[domain[complement[#]]] &,
         SubstTest[class, pair[x, y], implies[member[pair[x, y], u], member[pair[x, y], v]],
          {u → restrict[COMMUTE, RFX, RFX], v -> image[inverse[COMPOSE], RFX]}]]

Out[2]= subclass[image[COMPOSE, composite[id[RFX], COMMUTE, id[RFX]]], RFX] == True

In[3]:= % /. Equal→ SetDelayed
```

The key to deriving an inclusion in the opposite direction is the observation that any reflexive relation **rfx[x]** is the composite of the commuting reflexive relations **id[fix[rfx[x]]]** and **rfx[x]**. To obtain a variable-free formulation of this idea, one can use the function **IMAGE[id[Id]]** which takes **x** to **intersection[Id, x] = id[fix[x]]**.

Lemma. The **rfx[x]** wrapper can be used to derive this result:

```
In[4]:= member[rfx[x], fix[composite[COMMUTE, IMAGE[id[Id]]]]] // AssertTest

Out[4]= member[rfx[x], fix[composite[COMMUTE, IMAGE[id[Id]]]]] == member[fix[x], V]

In[5]:= member[rfx[x_], fix[composite[COMMUTE, IMAGE[id[Id]]]]] := member[fix[x], V]
```

One can eliminate the **rfx** wrapper and the variable **x** at the same time by using **reify**.

```
In[6]:= Map[subclass[RFX, image[CORE[RFX], complement[domain[#]]]] &,
          SubstTest[reify, x, dif[id[set[rfx[x]]], t], t -> composite[COMMUTE, IMAGE[id[Id]]]]]

Out[6]= subclass[RFX, fix[composite[COMMUTE, IMAGE[id[Id]]]]] == True

In[7]:= % /. Equal→ SetDelayed
```

Theorem. This is a better variable-free statement of the observation that **id[fix[rfx[x]]]** commutes with **rfx[x]**.

```
In[8]:= SubstTest[implies, subclass[u, v],
            subclass[composite[t, u], composite[t, v]], {t → IMAGE[id[Id]], u → id[RFX],
             v → inverse[composite[COMMUTE, IMAGE[id[Id]]]]}] // Reverse // InvertFix

Out[8]= subclass[composite[IMAGE[id[Id]], id[RFX]], COMMUTE] == True

In[9]:= subclass[composite[IMAGE[id[Id]], id[RFX]], COMMUTE] := True
```

Theorem. Here **reify** is used to obtain a variable-free statement of the observation that the composite of the reflexive relations **id[fix[rfx[x]]]** and **rfx[x]** is equal to **rfx[x]**.

```
In[10]:= Map[range[VERTSECT[#]] &, SubstTest[reify, x,
           APPLY[composite[COMPOSE, id[IMAGE[id[Id]]], inverse[FIRST]], f[x]], f → rfx]]

Out[10]= image[COMPOSE, composite[IMAGE[id[Id]], id[RFX]]] == RFX

In[11]:= image[COMPOSE, composite[IMAGE[id[Id]], id[RFX]]] := RFX
```

Theorem. The above results are used to derived the desired reverse inclusion.

```
In[12]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → COMPOSE, u→ composite[IMAGE[id[Id]], id[RFX]],
            v -> composite[id[RFX], COMMUTE, id[RFX]]}] // Reverse

Out[12]= subclass[RFX, image[COMPOSE, composite[id[RFX], COMMUTE, id[RFX]]]] == True

In[13]:= % /. Equal→ SetDelayed
```

Theorem. The two inclusions are combined into an equation and made into a rewrite rule.

```
In[14]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> image[COMPOSE, composite[id[RFX], COMMUTE, id[RFX]]], v → RFX}]

Out[14]= equal[RFX, image[COMPOSE, composite[id[RFX], COMMUTE, id[RFX]]]] == True

In[15]:= image[COMPOSE, composite[id[RFX], COMMUTE, id[RFX]]] := RFX
```

comment

A similar argument can be used to derive an analogous result with **RFX** replaced by **TRV**. For that one needs to use the fact that any relation **x** commutes with the identity relation **id[udora[x]]**.