CORE[RFX] and CORE[SYM] commute

Johan G. F. Belinfante
2004 October 4

```
In[1]:= SetDirectory["i:"]; << goedel62.01b; << tools.m

        :Package Title: goedel62.01b          2004 October 2 at 8:50 p.m.

        It is now:  2004 Oct 4 at 20:58

        Loading Simplification Rules

        TOOLS.M                      Revised 2004 September 25

        weightlimit = 40
```

---

summary

The composite of **CORE[RFX]** and **CORE[SYM]** in either order is equal to **CORE[intersection[RFX,SYM]]**.

---

commute

The reflexive core of the symmetric core is equal to the symmetric core of the reflexive core.

```
In[2]:= core[RFX, core[SYM, x]] == core[SYM, core[RFX, x]]

Out[2]= True
```

From this one can deduce that the functions **CORE[RFX]** and **CORE[SYM]** commute.

```
In[3]:= symdif[composite[CORE[RFX], CORE[SYM]],
          composite[CORE[SYM], CORE[RFX]]] // VSNormality

Out[3]= union[intersection[composite[complement[CORE[RFX]], CORE[SYM]],
           composite[CORE[SYM], CORE[RFX]]],
          intersection[composite[complement[CORE[SYM]], CORE[RFX]],
           composite[CORE[RFX], CORE[SYM]]]] == 0

In[4]:= % /. Equal → SetDelayed
```

```
In[5]:= SubstTest[equal, 0, symdif[u, v], {u -> composite[CORE[RFX], CORE[SYM]],
           v -> composite[CORE[SYM], CORE[RFX]]}] // Reverse

Out[5]= equal[composite[CORE[RFX], CORE[SYM]], composite[CORE[SYM], CORE[RFX]]] == True

In[6]:= % /. Equal → SetDelayed
```

Corollary. The composite is idempotent.

```
In[7]:= SubstTest[implies, and[idempotent[x], idempotent[y], commute[x, y]],
         idempotent[composite[x, y]], {x → CORE[RFX], y → CORE[SYM]}]

Out[7]= equal[composite[CORE[RFX], CORE[SYM]],
          composite[CORE[RFX], CORE[SYM], CORE[RFX], CORE[SYM]]] == True

In[8]:= % /. Equal → SetDelayed
```

---

other temporary lemmas

The following temporary lemmas will all be subsumed by later rules.

```
In[9]:= fix[composite[CORE[RFX], CORE[SYM]]] // Normality

Out[9]= fix[composite[CORE[RFX], CORE[SYM]]] == intersection[RFX, SYM]

In[10]:= % /. Equal → SetDelayed
```

Lemma

```
In[11]:= SubstTest[implies, and[subcommute[x, z], subcommute[y, z]],
          subcommute[composite[x, y], z], {x → CORE[RFX], y → CORE[SYM], z → S}]

Out[11]= subclass[composite[CORE[RFX], CORE[SYM], S],
           composite[S, CORE[RFX], CORE[SYM]]] == True

In[12]:= % /. Equal → SetDelayed
```

Lemma.

```
In[13]:= SubstTest[implies, and[subclass[u, x], subclass[v, y]],
          subclass[composite[u, v], composite[x, y]],
          {u → CORE[RFX], v → CORE[SYM], x → inverse[S], y → inverse[S]}]

Out[13]= subclass[composite[inverse[CORE[SYM]], inverse[CORE[RFX]]], S] == True

In[14]:= % /. Equal → SetDelayed
```

The following will be needed later.

```
In[15]:= (composite[CORE[RFX], CORE[SYM]] // VSNormality // Reverse) /.
          Equal → SetDelayed
```

uniqueness theorem for CORE

The uniiqueness theorem for **CORE** now implies:

```
In[16]:= SubstTest[implies, and[FUNCTION[x], idempotent[x],
           subcommute[x, S], subclass[x, inverse[S]], equal[V, domain[x]]],
          equal[x, CORE[fix[x]]], x → composite[CORE[RFX], CORE[SYM]]]

Out[16]= equal[composite[CORE[RFX], CORE[SYM]], CORE[intersection[RFX, SYM]]] == True

In[17]:= composite[CORE[RFX], CORE[SYM]] := CORE[intersection[RFX, SYM]]

In[18]:= composite[CORE[SYM], CORE[RFX]] // VSNormality

Out[18]= composite[CORE[SYM], CORE[RFX]] == CORE[intersection[RFX, SYM]]

In[19]:= composite[CORE[SYM], CORE[RFX]] := CORE[intersection[RFX, SYM]]
```

image rules

The following general result holds:

```
In[20]:= ImageComp[CORE[x], CORE[x], V] // Reverse

Out[20]= image[CORE[x], Uclosure[x]] == Uclosure[x]

In[21]:= image[CORE[x_], Uclosure[x_]] := Uclosure[x]
```

Corollary.   (Comment: The corresponding result for **SYM** is already in the **GOEDEL** program.)

```
In[22]:= SubstTest[image, CORE[x], Uclosure[x], x → RFX]

Out[22]= image[CORE[RFX], RFX] == RFX

In[23]:= image[CORE[RFX], RFX] := RFX
```

Other such rules:

```
In[24]:= ImageComp[CORE[RFX], CORE[SYM], V] // Reverse

Out[24]= image[CORE[RFX], SYM] == intersection[RFX, SYM]
```

```
In[25]:= image[CORE[RFX], SYM] := intersection[RFX, SYM]

In[26]:= ImageComp[CORE[SYM], CORE[RFX], V] // Reverse

Out[26]= image[CORE[SYM], RFX] == intersection[RFX, SYM]

In[27]:= image[CORE[SYM], RFX] := intersection[RFX, SYM]
```