reflexive symmetric core

Johan G. F. Belinfante
2006 October 11

```
In[1]:= SetDirectory["l:"]; << goedel86.10a; << tools.m
        :Package Title: goedel86.10a          2006 October 10 at 1:50 p.m.
        It is now:  2006 Oct 11 at 19:23
        Loading Simplification Rules
        TOOLS.M                      Revised 2006 October 10
        weightlimit = 40
```

summary

The reflexive symmetric core of any class is the symmetric core of its reflexive core. The main idea in the derivation is that any reflexive symmetric relation is the union of all the cartesian squares that it contains.

observation

The symmetric core of the reflexive core and the reflexive core of the symmetric core are the same:

```
In[2]:= core[RFX, core[SYM, x]]
Out[2]= intersection[inverse[rfx[x]], rfx[x]]

In[3]:= core[SYM, core[RFX, x]]
Out[3]= intersection[inverse[rfx[x]], rfx[x]]
```

It will be shown in this notebook that this is also a formula for **core[intersection[RFX, SYM], x]**. For the special case that **x** is a set, this formula can be easily deduced from the following formula:

```
In[4]:= composite[CORE[RFX], CORE[SYM]]
Out[4]= CORE[intersection[RFX, SYM]]
```

What is not so obvious is that the formula for **core[intersection[RFX, SYM], x]** also holds when **x** is an arbitrary class.

replacement for an existing rule

The following existing rewrite rule is being removed and will be replaced with a simpler one:

```
In[5]:= composite[inverse[E], id[cliques[x]], E]

Out[5]= composite[id[fix[x]], intersection[x, inverse[x]], id[fix[x]]]

In[6]:= composite[inverse[E], id[cliques[x_]], E] =.
```

Lemma.

```
In[7]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
         {u → intersection[x, inverse[x]], v → x, w → id[cartsq[fix[x]]]}]

Out[7]= subclass[composite[id[fix[x]], intersection[x, inverse[x]], id[fix[x]]], rfx[x]] == True

In[8]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[9]:= SubstTest[and, subclass[u, v], subclass[v, u],
         {u -> composite[id[fix[x]], intersection[x, inverse[x]], id[fix[x]]],
          v -> intersection[rfx[x], inverse[rfx[x]]]}]

Out[9]= True == equal[composite[id[fix[x]], intersection[x, inverse[x]], id[fix[x]]],
          intersection[inverse[rfx[x]], rfx[x]]]

In[10]:= composite[id[fix[x_]], intersection[x_, inverse[x_]], id[fix[x_]]] :=
          intersection[inverse[rfx[x]], rfx[x]]
```

The replacement rule is

```
In[11]:= IminComp[CUP, cross[SINGLETON, SINGLETON], cliques[x]] // Reverse

Out[11]= composite[inverse[E], id[cliques[x]], E] == intersection[inverse[rfx[x]], rfx[x]]

In[12]:= composite[inverse[E], id[cliques[x_]], E] := intersection[inverse[rfx[x]], rfx[x]]
```

IMAGE[SWAP] rules

```
In[13]:= Assoc[IMAGE[SWAP], id[SYM], CORE[SYM]]

Out[13]= composite[IMAGE[SWAP], CORE[SYM]] == CORE[SYM]

In[14]:= composite[IMAGE[SWAP], CORE[SYM]] := CORE[SYM]

In[15]:= Assoc[IMAGE[SWAP], CORE[SYM], CORE[RFX]]

Out[15]= composite[IMAGE[SWAP], CORE[intersection[RFX, SYM]]] == CORE[intersection[RFX, SYM]]
```

```
In[16]:= composite[IMAGE[SWAP], CORE[intersection[RFX, SYM]]] := CORE[intersection[RFX, SYM]]

In[17]:= Assoc[CORE[RFX], CORE[SYM], IMAGE[SWAP]] // Reverse

Out[17]= composite[CORE[intersection[RFX, SYM]], IMAGE[SWAP]] == CORE[intersection[RFX, SYM]]

In[18]:= composite[CORE[intersection[RFX, SYM]], IMAGE[SWAP]] := CORE[intersection[RFX, SYM]]
```

symmetry

```
In[19]:= Map[U, ImageComp[IMAGE[SWAP], CORE[intersection[RFX, SYM]], P[x]]] // Reverse

Out[19]= inverse[core[intersection[RFX, SYM], x]] == core[intersection[RFX, SYM], x]

In[20]:= inverse[core[intersection[RFX, SYM], x_]] := core[intersection[RFX, SYM], x]
```

Corollary.

```
In[21]:= SubstTest[inverse, inverse[y], y -> core[intersection[RFX, SYM], x]] // Reverse

Out[21]= composite[Id, core[intersection[RFX, SYM], x]] == core[intersection[RFX, SYM], x]

In[22]:= composite[Id, core[intersection[RFX, SYM], x_]] := core[intersection[RFX, SYM], x]
```

cliques

The class of all sets whose cartesian squares are contained in **x** is

```
In[23]:= class[w, subclass[cartsq[w], x]]

Out[23]= cliques[x]
```

The following inclusion is immediate:

```
In[24]:= SubstTest[implies, subclass[u, v], subclass[cliques[u], cliques[v]],
          {u → core[intersection[RFX, SYM], x], v → x}]

Out[24]= subclass[cliques[core[intersection[RFX, SYM], x]], cliques[x]] == True

In[25]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[26]:= ImageComp[CLIQUES, CORE[intersection[RFX, SYM]], P[x]] // Reverse

Out[26]= image[CLIQUES, image[CORE[intersection[RFX, SYM]], P[x]]] == image[CLIQUES, P[x]]

In[27]:= image[CLIQUES, image[CORE[intersection[RFX, SYM]], P[x_]]] := image[CLIQUES, P[x]]
```

The reverse inclusion now follows.

```
In[28]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u -> image[CORE[intersection[RFX, SYM]], P[x]],
           v → P[U[image[CORE[intersection[RFX, SYM]], P[x]]]],
           w → composite[inverse[E], CLIQUES]}]

Out[28]= subclass[cliques[x], cliques[core[intersection[RFX, SYM], x]]] == True

In[29]:= (% /. x → x_) /. Equal → SetDelayed
```

Combining the two inclusions yields an equation, which is made into a temporary rewrite rule:

```
In[30]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> cliques[x], v -> cliques[core[intersection[RFX, SYM], x]]}]

Out[30]= True == equal[cliques[x], cliques[core[intersection[RFX, SYM], x]]]

In[31]:= cliques[core[intersection[RFX, SYM], x_]] := cliques[x]
```

IMAGE[DUP] rules

Theorem.

```
In[32]:= Map[VERTSECT, SubstTest[reify, x, rfx[f[x]], f → id]] // Reverse

Out[32]= composite[CORE[RFX], IMAGE[DUP]] == IMAGE[DUP]

In[33]:= composite[CORE[RFX], IMAGE[DUP]] := IMAGE[DUP]
```

Theorem.

```
In[34]:= Map[VERTSECT, SubstTest[reify, x, sym[f[x]], f → id]] // Reverse

Out[34]= composite[CORE[SYM], IMAGE[DUP]] == IMAGE[DUP]

In[35]:= composite[CORE[SYM], IMAGE[DUP]] := IMAGE[DUP]
```

Corollary.

```
In[36]:= Assoc[CORE[RFX], CORE[SYM], IMAGE[DUP]] // Reverse

Out[36]= composite[CORE[intersection[RFX, SYM]], IMAGE[DUP]] == IMAGE[DUP]

In[37]:= composite[CORE[intersection[RFX, SYM]], IMAGE[DUP]] := IMAGE[DUP]
```

fix formula for the reflexive symmetric core

The first lemma is obtained as an application of the **IMAGE[DUP]** rewrite rules. This is made into a temporary rewrite rule.

```
In[38]:= Map[U, ImageComp[CORE[intersection[RFX, SYM]], IMAGE[DUP], P[x]]] // Reverse

Out[38]= core[intersection[RFX, SYM], id[x]] == id[x]

In[39]:= core[intersection[RFX, SYM], id[x_]] := id[x]
```

Corollary.

```
In[40]:= SubstTest[implies, subclass[u, v], subclass[core[w, u], core[w, v]],
          {u → id[fix[x]], v → x, w → intersection[RFX, SYM]}]

Out[40]= subclass[fix[x], fix[core[intersection[RFX, SYM], x]]] == True

In[41]:= (% /. x → x_) /. Equal → SetDelayed
```

The reverse inclusion also holds, yielding an equation, and another temporary rewrite rule.

```
In[42]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> fix[x], v -> fix[core[intersection[RFX, SYM], x]]}]

Out[42]= True == equal[fix[x], fix[core[intersection[RFX, SYM], x]]]

In[43]:= fix[core[intersection[RFX, SYM], x_]] := fix[x]
```

reflexivity

Lemma.

```
In[44]:= SubstTest[implies, subclass[u, v], subclass[core[u, x], core[v, x]],
          {u → intersection[RFX, SYM], v → RFX}]

Out[44]= subclass[core[intersection[RFX, SYM], x], rfx[x]] == True

In[45]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[46]:= SubstTest[implies, subclass[u, v], subclass[domain[u], domain[v]],
          {u -> core[intersection[RFX, SYM], x], v → rfx[x]}]

Out[46]= subclass[domain[core[intersection[RFX, SYM], x]], fix[x]] == True

In[47]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[48]:= SubstTest[subclass, fix[w], domain[w], w -> core[intersection[RFX, SYM], x]]

Out[48]= subclass[fix[x], domain[core[intersection[RFX, SYM], x]]] == True

In[49]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. (Temporary rewrite rule for the domain of the reflexive symmetric core.)

```
In[50]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> fix[x], v -> domain[core[intersection[RFX, SYM], x]]}]

Out[50]= True == equal[domain[core[intersection[RFX, SYM], x]], fix[x]]

In[51]:= domain[core[intersection[RFX, SYM], x_]] := fix[x]
```

Corollary. (Temporary rewrite rule for the range of the reflexive symmetric core.)

```
In[52]:= SubstTest[domain, inverse[w], w → core[intersection[RFX, SYM], x]] // Reverse

Out[52]= range[core[intersection[RFX, SYM], x]] == fix[x]

In[53]:= range[core[intersection[RFX, SYM], x_]] := fix[x]
```

main theorem

Lemma.

```
In[54]:= SubstTest[composite, id[fix[y]], y,
           id[fix[y]], y -> core[intersection[RFX, SYM], x]] // Reverse

Out[54]= rfx[core[intersection[RFX, SYM], x]] == core[intersection[RFX, SYM], x]

In[55]:= rfx[core[intersection[RFX, SYM], x_]] := core[intersection[RFX, SYM], x]
```

Theorem. The reflexive symmetric core of a class is the symmetric core of its reflexive core.

```
In[56]:= SubstTest[composite, inverse[E], id[cliques[u]],
           E, u -> core[intersection[RFX, SYM], x]] // Reverse

Out[56]= core[intersection[RFX, SYM], x] == intersection[inverse[rfx[x]], rfx[x]]

In[57]:= core[intersection[RFX, SYM], x_] := intersection[inverse[rfx[x]], rfx[x]]
```