intersection[RFX,TRV]

Johan G. F. Belinfante
2004 December 18

```
In[1]:= SetDirectory["i:"]; << goedel64.17b; << tools.m

        :Package Title: goedel64.17b              2004 December 17 at 9:30 p.m.

        It is now:  2004 Dec 18 at 6:32

        Loading Simplification Rules

        TOOLS.M                        Revised 2004 December 16

        weightlimit = 40
```

summary

Reflexive transitive relations, also known as quasi-orders, satisfy some properties analogous to those of equivalence relations. In this notebook, variable-free formulations of facts about the class **intersection[RFX,TRV]** of small reflexive symmetric relations are derived, culminating in a formula for a canonical factorization of such relations.

reflexive core of a transitive relation

If **x** is transitive, so is its reflexive core.

```
In[2]:= SubstTest[implies, TRANSITIVE[x], TRANSITIVE[restrict[x, y, y]], y → fix[x]]

Out[2]= or[not[TRANSITIVE[x]], TRANSITIVE[rfx[x]]] == True

In[3]:= or[not[TRANSITIVE[x_]], TRANSITIVE[rfx[x_]]] := True
```

The class **TRV** is invariant under **CORE[RFX]**.

```
In[4]:= Map[equal[0, composite[Id, complement[#]]] &, SubstTest[class, pair[x, z],
            implies[member[z, image[u, singleton[setpart[x]]]], member[z, v]],
            {u → composite[HULL[TRV], IMAGE[id[cart[V, V]]]],
             v -> image[inverse[CORE[RFX]], TRV]}]] // Reverse

Out[4]= subclass[image[CORE[RFX], TRV], TRV] == True

In[5]:= % /. Equal → SetDelayed
```

The image is also contained in **RFX**.

```
In[6]:= Map[subclass[#, RFX] &, ImageComp[id[RFX], CORE[RFX], TRV]]

Out[6]= subclass[image[CORE[RFX], TRV], RFX] == True

In[7]:= % /. Equal → SetDelayed
```

The reverse inclusion also holds:

```
In[8]:= SubstTest[implies, subclass[u, v],
         subclass[image[u, w], image[v, w]], {u → id[RFX], v → CORE[RFX], w → TRV}]

Out[8]= subclass[intersection[RFX, TRV], image[CORE[RFX], TRV]] == True

In[9]:= % /. Equal → SetDelayed
```

The following equation follows:

```
In[10]:= equal[image[CORE[RFX], TRV], intersection[RFX, TRV]] // AssertTest

Out[10]= equal[image[CORE[RFX], TRV], intersection[RFX, TRV]] == True

In[11]:= image[CORE[RFX], TRV] := intersection[RFX, TRV]
```

---

an antitone property of vertical sections

The truth of the following statement is currently recognized by the **GOEDEL** program. It says that the vertical sections of a transitive relation satisfy an antitone property with respect to inclusions: if **u** is related to **v** by a transitive relation **x**, then the vertical section of **v** is contained in that of **u**.

```
In[12]:= or[not[member[pair[u, v], x]], not[TRANSITIVE[x]],
          subclass[image[x, singleton[v]], image[x, singleton[u]]]]

Out[12]= True
```

A version of this fact which eliminates the variables **u** and **v** is readily derived in the case that **x** is thin:

```
In[13]:= Map[subclass[x, #] &,
          composite[inverse[VERTSECT[x]], inverse[S], VERTSECT[x]] // RelnNormality]

Out[13]= subclass[x, composite[inverse[VERTSECT[x]], inverse[S], VERTSECT[x]]] ==
          and[equal[V, domain[VERTSECT[x]]], TRANSITIVE[x]]
```

```
In[14]:= subclass[x_, composite[inverse[VERTSECT[x_]], inverse[S], VERTSECT[x_]]] :=
          and[equal[V, domain[VERTSECT[x]]], TRANSITIVE[x]]
```

The following corollary requires nothing further:

```
In[15]:= subclass[x, composite[id[fix[x]],
           inverse[VERTSECT[x]], inverse[S], VERTSECT[x], id[fix[x]]]]

Out[15]= and[equal[V, domain[VERTSECT[x]]], REFLEXIVE[x], TRANSITIVE[x]]
```

It will be shown below that this inclusion can be replaced with an equation.

---

a factorization result

Recall that **composite[inverse[funpart[x]], inverse[S], funpart[x]]** is both reflexive and transitive. Variable-free restatements of these facts can be obtained as follows. For the reflexive property, one obtains:

```
In[16]:= Map[equal[V, #] &,
           SubstTest[class, x, subclass[image[u, singleton[setpart[x]]], v],
            {u -> composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
                composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]]],
              FUNPART], v → RFX}]] // Reverse

Out[16]= subclass[FUNS, fix[composite[INVERSE, inverse[image[inverse[COMPOSE], RFX]],
           IMAGE[cross[Id, inverse[S]]]]]] == True

In[17]:= % /. Equal → SetDelayed
```

A similar result holds for the transitive property:

```
In[18]:= Map[equal[V, #] &,
           SubstTest[class, x, subclass[image[u, singleton[setpart[x]]], v],
            {u -> composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
                composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]]],
              FUNPART], v → TRV}]] // Reverse

Out[18]= subclass[FUNS, fix[composite[INVERSE, inverse[image[inverse[COMPOSE], TRV]],
           IMAGE[cross[Id, inverse[S]]]]]] == True

In[19]:= % /. Equal → SetDelayed
```

These two results will now be solved to derive an lower bound for **intersection[RFX,-TRV]**. Note that the following is a function:

```
In[20]:= abstract[x,
          image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]], id[x], INVERSE]]]

Out[20]= composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
           composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]]]

In[21]:= composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
            composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]]] // FUNCTION

Out[21]= True
```

Consequently, one has:

```
In[22]:= SubstTest[composite, funpart[x], inverse[funpart[x]],
          x -> composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
             composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]]]]

Out[22]= composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
            composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]],
           intersection[composite[INVERSE, FIRST], composite[
             inverse[IMAGE[cross[Id, inverse[S]]]], SECOND]], inverse[COMPOSE]] ==
          id[image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]], INVERSE]]]

In[23]:= % /. Equal → SetDelayed
```

Corollary:

```
In[24]:= ImageComp[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
             composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]]],
           inverse[composite[COMPOSE, intersection[
              composite[inverse[FIRST], INVERSE], composite[inverse[SECOND],
               IMAGE[cross[Id, inverse[S]]]]]]], x] // Reverse

Out[24]= image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]],
            id[fix[composite[INVERSE, inverse[image[inverse[COMPOSE], x]],
               IMAGE[cross[Id, inverse[S]]]]]], INVERSE]] == intersection[x,
           image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]], INVERSE]]]
```

This is made into a temporary rewrite rule:

```
In[25]:= image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]],
            id[fix[composite[INVERSE, inverse[image[inverse[COMPOSE], x_]],
               IMAGE[cross[Id, inverse[S]]]]]], INVERSE]] := intersection[x,
           image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]], INVERSE]]]
```

The statement obtained for the reflexive property is then transformed as follows:

```
In[26]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → FUNS, v -> fix[composite[INVERSE,
              inverse[image[inverse[COMPOSE], RFX]], IMAGE[cross[Id, inverse[S]]]]]],
           w -> composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
              composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]]]}]

Out[26]= subclass[image[COMPOSE,
            composite[IMAGE[cross[Id, inverse[S]]], id[FUNS], INVERSE]], RFX] == True

In[27]:= % /. Equal → SetDelayed
```

A similar restatement is obtained for the transitive property:

```
In[28]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → FUNS, v -> fix[composite[INVERSE,
              inverse[image[inverse[COMPOSE], TRV]], IMAGE[cross[Id, inverse[S]]]]]],
           w -> composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
              composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]]]}]

Out[28]= subclass[image[COMPOSE,
            composite[IMAGE[cross[Id, inverse[S]]], id[FUNS], INVERSE]], TRV] == True

In[29]:= % /. Equal → SetDelayed
```

---

the antitone property

Conversely, for any thin reflexive transitive relation **x** there is a canonical factorization of the form **x = composite[inverse[funpart[y]], inverse[S], funpart[y]]** where **y** is the restriction of the vertical section function to **fix[x]**. For sets, this result is:

```
In[30]:= (equal[z, composite[inverse[funpart[y]], inverse[S], funpart[y]]] /.
            y -> composite[VERTSECT[z], id[fix[z]]]) /. z -> rfx[trv[setpart[x]]]

Out[30]= True
```

The variable **x** can be eliminated as follows.

```
In[31]:= SubstTest[class, x,
           equal[image[u, singleton[setpart[x]]], image[v, singleton[setpart[x]]]],
           {u -> composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
               composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]]],
              IMAGE[id[cart[V, V]]], VS, CORE[RFX], HULL[TRV], IMAGE[id[cart[V, V]]]],
            v → composite[CORE[RFX], HULL[TRV], IMAGE[id[cart[V, V]]]]}] // Reverse

Out[31]= image[inverse[IMAGE[id[cart[V, V]]]],
           image[inverse[HULL[TRV]], image[inverse[CORE[RFX]],
             fix[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
                composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]], VS]]]]] == V

In[32]:= % /. Equal → SetDelayed
```

The wrapper-related functions **CORE[RFX]** and **composite[HULL[TRV], IMAGE[id[cart[V,V]]]]** can be eliminated using **ImageComp**:

```
In[33]:= Map[subclass[intersection[RFX, TRV], #] &,
          ImageComp[composite[CORE[RFX], HULL[TRV], IMAGE[id[cart[V, V]]]],
           inverse[composite[CORE[RFX], HULL[TRV], IMAGE[id[cart[V, V]]]]],
           fix[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
              composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]], VS]]]]

Out[33]= subclass[intersection[RFX, TRV],
           fix[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
              composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]], VS]]] == True

In[34]:= % /. Equal → SetDelayed
```

The next step uses the fact that **VS** is contained in **cart[V, FUNS]**. One needs this relation:

```
In[35]:= abstract[x,
          fix[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
             composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]], x]]]

Out[35]= composite[FIRST,
          id[composite[intersection[composite[INVERSE, FIRST], composite[
              inverse[IMAGE[cross[Id, inverse[S]]]], SECOND]], inverse[COMPOSE]]]]
```

The result is this inclusion:

```
In[36]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → VS, v → cart[V, FUNS], w -> composite[FIRST,
             id[composite[intersection[composite[INVERSE, FIRST], composite[inverse[
                  IMAGE[cross[Id, inverse[S]]]], SECOND]], inverse[COMPOSE]]]]}]

Out[36]= subclass[
           fix[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
              composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]]], VS]],
           image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]],
             id[FUNS], INVERSE]]] == True

In[37]:= % /. Equal → SetDelayed
```

Combining this inclusion with the one derived for **intersection[RFX, TRV]** yields:

```
In[38]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u -> intersection[RFX, TRV],
           v -> fix[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
               composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]]], VS]],
           w -> image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]],
              id[FUNS], INVERSE]]}]

Out[38]= subclass[intersection[RFX, TRV], image[COMPOSE,
            composite[IMAGE[cross[Id, inverse[S]]], id[FUNS], INVERSE]]] == True

In[39]:= % /. Equal → SetDelayed
```

The reverse inclusion was proved in the preceding section. Combining these, one obtains a variable-free equation which says that (small) reflexive transitive relations can be characterized by the existence of a factorization of the form **composite[inverse[f], inverse[S], f]** where **f** is a function.

```
In[40]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> intersection[RFX, TRV], v -> image[COMPOSE,
             composite[IMAGE[cross[Id, inverse[S]]], id[FUNS], INVERSE]]}]

Out[40]= True == equal[
           image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]], id[FUNS], INVERSE]],
           intersection[RFX, TRV]]

In[41]:= image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]], id[FUNS], INVERSE]] :=
          intersection[RFX, TRV]
```

Note the resemblance of this formula with a corresponding one for the class of equivalence relations:

```
In[42]:= image[COMPOSE, composite[id[FUNS], INVERSE]]

Out[42]= EQV
```