characterizing transitivity on a class

Johan G. F. Belinfante
2009 April 26

```
In[1]:= SetDirectory["l:"]; << goedel.09apr25a; << tools.m

        :Package Title: goedel.09apr25a            2009 April 25 at 11:20 a.m.

        It is now:  2009 Apr 26 at 13:38

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 April 6

        weightlimit = 40
```

summary

A relation **r** is said to be **transitive** on a class **s** if **r ⋂ cart[s,s]** is a transitive relation. In this notebook it is shown that this condition can be rewritten as **P[s] ⊂ fix[image[inverse[CART], image[inverse[IMAGE[id[r]]], TRV]]]**, where **TRV** is the class of all (small) transitive relations.

derivation

Lemma.

```
In[2]:= Assoc[IMAGE[id[y]], IMAGE[id[cart[z, x]]], CART] // Reverse

Out[2]= composite[IMAGE[id[composite[id[x], y, id[z]]]], CART] ==
         composite[IMAGE[id[y]], CART, cross[IMAGE[id[z]], IMAGE[id[x]]]]

In[3]:= composite[IMAGE[id[composite[id[x_], y_, id[z_]]]], CART] :=
         composite[IMAGE[id[y]], CART, cross[IMAGE[id[z]], IMAGE[id[x]]]]
```

Theorem. A characterization of the condition that a relation be transitive on a class.

```
In[4]:= Map[equal[V, #] &,
         IminComp[IMAGE[id[composite[id[x], y, id[x]]]], composite[CART, DUP], TRV]]

Out[4]= subclass[P[x], fix[image[inverse[CART], image[inverse[IMAGE[id[y]]], TRV]]]] ==
         TRANSITIVE[composite[id[x], y, id[x]]]

In[5]:= subclass[P[x_], fix[image[inverse[CART], image[inverse[IMAGE[id[y_]]], TRV]]]] :=
         TRANSITIVE[composite[id[x], y, id[x]]]
```

related results and comments

Theorem. A special case of a simplification rule.

```
In[6]:= SubstTest[composite, IMAGE[id[composite[id[t], x, id[t]]]], CART, t → V] // Reverse

Out[6]= composite[IMAGE[id[composite[Id, x]]], CART] == composite[IMAGE[id[x]], CART]

In[7]:= composite[IMAGE[id[composite[Id, x_]]], CART] := composite[IMAGE[id[x]], CART]
```

The main theorem of the preceding section was first derived for the special case that **s** is a set using the following result:

Theorem.

```
In[8]:= equiv[member[t, image[inverse[S], x]],
           and[member[t, V], subclass[P[t], image[inverse[S], x]]]] // not // not

Out[8]= True

In[9]:= and[member[t_, V], subclass[P[t_], image[inverse[S], x_]]] :=
         member[t, image[inverse[S], x]]
```

A class **x** is said to be **hereditary** if every subset of a member is a member. A class **x** is hereditary if and only if it satisfies **image[inverse[S], x] = x**. Although the class **TRV** of transitive relations is not itself hereditary, the class of sets on which a given relation **r** is transitive. This class of all sets **s** on which **r** is transitive is precisely the class that figures in the main theorem of the preceding section:

```
In[10]:= member[s, fix[image[inverse[CART], image[inverse[IMAGE[id[r]]], TRV]]]]

Out[10]= and[member[s, V], TRANSITIVE[composite[id[s], r, id[s]]]]
```

The following corollary of the hereditary property of this class is sometimes useful as a simplification rule.

Theorem.

```
In[11]:= SubstTest[image, S, complement[image[inverse[S], t]],
            t -> fix[image[inverse[CART], image[inverse[IMAGE[id[x]]], TRV]]] // Reverse

Out[11]= image[S, complement[fix[image[inverse[CART], image[inverse[IMAGE[id[x]]], TRV]]]]] ==
          complement[fix[image[inverse[CART], image[inverse[IMAGE[id[x]]], TRV]]]]

In[12]:= image[S, complement[fix[image[inverse[CART], image[inverse[IMAGE[id[x_]]], TRV]]]]] :=
          complement[fix[image[inverse[CART], image[inverse[IMAGE[id[x]]], TRV]]]]
```

an application

If a relation **r** is asymmetric on a class **s**, then it is irreflexive. This can be written as:

```
In[13]:= implies[subclass[P[s], chains[complement[r]]], disjoint[s, fix[r]]]

Out[13]= True
```

The converse holds when **r** is transitive on **s**.

```
In[14]:= implies[and[TRANSITIVE[restrict[r, s, s]], disjoint[s, fix[r]]],
          subclass[P[s], chains[complement[r]]]]

Out[14]= True
```

These observations allow one to eliminate the variable **s** as follows:

```
In[15]:= Map[equal[V, #] &,
          SubstTest[class, s, implies[and[subclass[P[s], u], disjoint[s, v]], subclass[P[s], w]],
           {u -> fix[image[inverse[CART], image[inverse[IMAGE[id[r]]], TRV]]],
            v → fix[r], w → chains[complement[r]]}]]

Out[15]= subclass[intersection[fix[image[inverse[CART], image[inverse[IMAGE[id[r]]], TRV]]],
            P[complement[fix[r]]]], chains[complement[r]]] == True

In[16]:= subclass[intersection[fix[image[inverse[CART], image[inverse[IMAGE[id[x_]]], TRV]]],
            P[complement[fix[x_]]]], chains[complement[x_]]] := True
```

One can also express this as an equation that can be made into a rewrite rule. A lemma is needed for this.

Lemma.

```
In[23]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
           subclass[u, w], {u → intersection[chains[complement[x]],
              fix[image[inverse[CART], image[inverse[IMAGE[id[x]]], TRV]]]],
            v -> chains[complement[x]], w -> P[complement[fix[x]]]}] // Reverse

Out[23]= equal[0, intersection[fix[x], U[intersection[chains[complement[x]],
              fix[image[inverse[CART], image[inverse[IMAGE[id[x]]], TRV]]]]]]] == True

In[24]:= (% /. x → x_) /. Equal→ SetDelayed
```

The following rewrite rule says that the class of sets on which a relation is asymmetric and transitive is the same as the class of sets on which that relation is irreflexive and transitive.

Corollary.

```
In[25]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> intersection[fix[image[inverse[CART], image[inverse[IMAGE[id[x]]], TRV]]],
             P[complement[fix[x]]]],
           v -> intersection[fix[image[inverse[CART], image[inverse[IMAGE[id[x]]], TRV]]],
             chains[complement[x]]]}]

Out[25]= equal[intersection[chains[complement[x]],
            fix[image[inverse[CART], image[inverse[IMAGE[id[x]]], TRV]]]],
           intersection[fix[image[inverse[CART], image[inverse[IMAGE[id[x]]], TRV]]],
            P[complement[fix[x]]]]] == True
```

```
In[27]:= intersection[chains[complement[x_]],
           fix[image[inverse[CART], image[inverse[IMAGE[id[x_]]], TRV]]]] := intersection[
           fix[image[inverse[CART], image[inverse[IMAGE[id[x]]], TRV]]], P[complement[fix[x]]]]
```