TRANSITIVE

Johan G. F. Belinfante
2003 July 1

```
In[1]:= << goedel52.s28; << tools.m

        :Package Title: goedel52.s28       2003 July 1 at 12:15 noon

        It is now:  2003 Jul 6 at 21:53

        Loading Simplification Rules

        TOOLS.M                      Revised 2003 July 1

        weightlimit = 40
```

■ summary

The definition of **TRANSITIVE** has been wrapped in **class** to prevent it from being immediately expanded. Clauses for the definition of **TRANSITIVE** are derived. It is shown that if **x** is transitive, and **x = y**, then **y** is transitive, and various other properties of transitivity.

■ clauses

The original unwrapped definition of **TRANSITIVE** can be restored by applying **AssertTest**.

```
In[2]:= TRANSITIVE[x] // AssertTest

Out[2]= TRANSITIVE[x] == and[subclass[x, cart[V, V]], subclass[composite[x, x], x]]
```

From this one can derive clauses useful for reasoning about transitive relations.

```
In[3]:= implies[TRANSITIVE[x], subclass[x, cart[V, V]]] // AssertTest

Out[3]= or[not[TRANSITIVE[x]], subclass[x, cart[V, V]]] == True

In[4]:= or[not[TRANSITIVE[x_]], subclass[x_, cart[V, V]]] := True

In[5]:= implies[TRANSITIVE[x], subclass[composite[x, x], x]] // AssertTest

Out[5]= or[not[TRANSITIVE[x]], subclass[composite[x, x], x]] == True

In[6]:= or[not[TRANSITIVE[x_]], subclass[composite[x_, x_], x_]] := True
```

```
In[7]:= implies[and[subclass[x, cart[V, V]], subclass[composite[x, x], x]], TRANSITIVE[x]] //
         AssertTest

Out[7]= or[not[subclass[x, cart[V, V]]], not[subclass[composite[x, x], x]], TRANSITIVE[x]] ==
         True

In[8]:= or[not[subclass[x_, cart[V, V]]],
          not[subclass[composite[x_, x_], x_]], TRANSITIVE[x_]] :=
         True
```

■ special version of the defining clauses

```
In[9]:= implies[TRANSITIVE[x], TRANSITIVE[composite[Id, x]]] // AssertTest

Out[9]= or[not[TRANSITIVE[x]], TRANSITIVE[composite[Id, x]]] == True

In[10]:= or[not[TRANSITIVE[x_]], TRANSITIVE[composite[Id, x_]]] := True
```

The converse is false; the case $\mathbf{x = V}$ would be a counterexample.

```
In[11]:= implies[subclass[composite[x, x], x], TRANSITIVE[composite[Id, x]]] // AssertTest

Out[11]= or[not[subclass[composite[x, x], x]], TRANSITIVE[composite[Id, x]]] == True

In[12]:= or[not[subclass[composite[x_, x_], x_]], TRANSITIVE[composite[Id, x_]]] := True

In[13]:= implies[TRANSITIVE[composite[Id, x]], subclass[composite[x, x], x]] // AssertTest

Out[13]= or[not[TRANSITIVE[composite[Id, x]]], subclass[composite[x, x], x]] == True

In[14]:= or[not[TRANSITIVE[composite[Id, x_]]], subclass[composite[x_, x_], x_]] := True
```

■ relation to TRV and trv[x]

```
In[15]:= class[x, TRANSITIVE[x]]

Out[15]= TRV

In[16]:= equiv[and[member[x, V], TRANSITIVE[x]], member[x, TRV]] // not // not

Out[16]= True

In[17]:= equiv[TRANSITIVE[x], equal[x, trv[x]]] // not // not

Out[17]= True

In[18]:= SubstTest[implies, subclass[z, x], subclass[trv[z], trv[x]], z -> intersection[x, y]]

Out[18]= subclass[trv[intersection[x, y]], trv[x]] == True

In[19]:= subclass[trv[intersection[x_, y_]], trv[x_]] := True
```

```
In[20]:= subclass[trv[intersection[x, y]], intersection[trv[x], trv[y]]]

Out[20]= True
```

■ equality substitution rule

```
In[21]:= SubstTest[implies,
          and[equal[u, v], equal[x, y]], equal[composite[u, x], composite[v, y]],
          {u -> x, v -> y}]

Out[21]= or[equal[composite[x, x], composite[y, y]], not[equal[x, y]]] == True

In[22]:= or[equal[composite[x_, x_], composite[y_, y_]], not[equal[x_, y_]]] := True

In[23]:= implies[and[equal[x, y], subclass[composite[x, x], x]], subclass[composite[y, y], y]]

Out[23]= or[not[equal[x, y]], not[subclass[composite[x, x], x]], subclass[composite[y, y], y]]

In[24]:= Map[not,
          SubstTest[and, implies[p1, p3], implies[and[p2, p3], p4], implies[and[p1, p4], p5],
           not[implies[and[p1, p2], p5]],
           {p1 -> equal[x, y], p2 -> subclass[composite[x, x], x],
            p3 -> equal[composite[x, x], composite[y, y]],
            p4 -> subclass[composite[y, y], x], p5 -> subclass[composite[y, y], y]}]]

Out[24]= or[not[equal[x, y]],
           not[subclass[composite[x, x], x]], subclass[composite[y, y], y]] ==
          True

In[25]:= or[not[equal[x_, y_]],
           not[subclass[composite[x_, x_], x_]], subclass[composite[y_, y_], y_]] :=
          True

In[26]:= implies[and[equal[x, y], TRANSITIVE[x]], TRANSITIVE[y]] // AssertTest // MapNotNot

Out[26]= or[not[equal[x, y]], not[TRANSITIVE[x]], TRANSITIVE[y]] == True

In[27]:= or[not[equal[x_, y_]], not[TRANSITIVE[x_]], TRANSITIVE[y_]] := True
```

■ transitivity of inverses

```
In[28]:= implies[TRANSITIVE[x], TRANSITIVE[inverse[x]]] // AssertTest

Out[28]= or[not[TRANSITIVE[x]], TRANSITIVE[inverse[x]]] == True

In[29]:= or[not[TRANSITIVE[x_]], TRANSITIVE[inverse[x_]]] := True

In[30]:= SubstTest[implies, TRANSITIVE[y], TRANSITIVE[inverse[y]], y -> composite[Id, x]]

Out[30]= or[not[TRANSITIVE[composite[Id, x]]], TRANSITIVE[inverse[x]]] == True

In[31]:= or[not[TRANSITIVE[composite[Id, x_]]], TRANSITIVE[inverse[x_]]] := True
```

```
In[32]:= SubstTest[implies, TRANSITIVE[y], TRANSITIVE[inverse[y]], y -> inverse[x]]

Out[32]= or[not[TRANSITIVE[inverse[x]]], TRANSITIVE[composite[Id, x]]] == True

In[33]:= or[not[TRANSITIVE[inverse[x_]]], TRANSITIVE[composite[Id, x_]]] := True

In[34]:= equiv[TRANSITIVE[inverse[x]], TRANSITIVE[composite[Id, x]]]

Out[34]= True

In[35]:= Equal[TRANSITIVE[inverse[x]], TRANSITIVE[composite[Id, x]]]

Out[35]= TRANSITIVE[inverse[x]] == TRANSITIVE[composite[Id, x]]

In[36]:= TRANSITIVE[inverse[x_]] := TRANSITIVE[composite[Id, x]]
```

■ intersections

The rule derived in this section would look terribly complicated were the definition of **TRANSITIVE** to be expanded.

```
In[37]:= SubstTest[implies, subclass[u, x], subclass[composite[u, u], composite[x, x]],
          u -> intersection[x, y]]

Out[37]= subclass[composite[intersection[x, y], intersection[x, y]], composite[x, x]] == True

In[38]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[39]:= implies[and[subclass[x, cart[V, V]], subclass[composite[x, x], x],
            subclass[y, cart[V, V]], subclass[composite[y, y], y]],
           and[subclass[intersection[x, y], cart[V, V]], subclass[
              composite[intersection[x, y], intersection[x, y]], intersection[x, y]]]] //
          NotNotTest

Out[39]= or[and[subclass[composite[intersection[x, y], intersection[x, y]], x],
             subclass[composite[intersection[x, y], intersection[x, y]], y],
             subclass[intersection[x, y], cart[V, V]]],
            not[subclass[x, cart[V, V]]], not[subclass[y, cart[V, V]]],
            not[subclass[composite[x, x], x]], not[subclass[composite[y, y], y]]] ==
           True

In[40]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[41]:= subclass[composite[intersection[x, y], intersection[x, y]], x] // AssertTest //
          Reverse

Out[41]= equal[0, fix[composite[x, intersection[
               composite[complement[inverse[x]], intersection[x, y]], inverse[y]]]]] ==
           subclass[composite[intersection[x, y], intersection[x, y]], x]

In[42]:= equal[0, fix[composite[x_, intersection[
               composite[complement[inverse[x_]], intersection[x_, y_]], inverse[y_]]]]] :=
           subclass[composite[intersection[x, y], intersection[x, y]], x]
```

```
In[43]:= implies[and[TRANSITIVE[x], TRANSITIVE[y]],
             TRANSITIVE[intersection[x, y]]] // AssertTest // MapNotNot //
          InvertFix

Out[43]= or[not[TRANSITIVE[x]], not[TRANSITIVE[y]], TRANSITIVE[intersection[x, y]]] == True

In[44]:= or[not[TRANSITIVE[x_]], not[TRANSITIVE[y_]], TRANSITIVE[intersection[x_, y_]]] :=
          True
```

■ corollaries

```
In[45]:= TRANSITIVE[cart[x, y]] // AssertTest

Out[45]= TRANSITIVE[cart[x, y]] == True

In[46]:= TRANSITIVE[cart[x_, y_]] := True

In[47]:= SubstTest[implies,
          and[TRANSITIVE[x], TRANSITIVE[w]], TRANSITIVE[intersection[x, w]],
          w -> cart[V, y]]

Out[47]= or[not[TRANSITIVE[x]], TRANSITIVE[composite[id[y], x]]] == True

In[48]:= or[not[TRANSITIVE[x_]], TRANSITIVE[composite[id[y_], x_]]] := True

In[49]:= SubstTest[implies,
          and[TRANSITIVE[x], TRANSITIVE[w]], TRANSITIVE[intersection[x, w]],
          w -> cart[y, V]]

Out[49]= or[not[TRANSITIVE[x]], TRANSITIVE[composite[x, id[y]]]] == True

In[50]:= or[not[TRANSITIVE[x_]], TRANSITIVE[composite[x_, id[y_]]]] := True
```

■ some examples

```
In[51]:= Select[Relns[NamedClasses], subclass[composite[#, #], #]&]

Out[51]= Relns[]

In[52]:= Map[AssertTest[TRANSITIVE[#]]&, %] // TableForm

  Out[52]//TableForm=
       Relns[]

In[53]:= TRANSITIVE[0] := True

In[54]:= TRANSITIVE[ACLOSURE] := True

In[55]:= TRANSITIVE[CARD] := True

In[56]:= TRANSITIVE[DIV] := True
```

```
In[57]:= TRANSITIVE[EQUIDIFF] := True

In[58]:= TRANSITIVE[FUNPART] := True

In[59]:= TRANSITIVE[HC] := True

In[60]:= TRANSITIVE[Id] := True

In[61]:= TRANSITIVE[PS] := True

In[62]:= TRANSITIVE[Q] := True

In[63]:= TRANSITIVE[RANK] := True

In[64]:= TRANSITIVE[RESTRICT] := True

In[65]:= TRANSITIVE[S] := True

In[66]:= TRANSITIVE[TC] := True

In[67]:= TRANSITIVE[UCLOSURE] := True

In[68]:= TRANSITIVE[ZN] := True
```

The following provides an easy counterexample for many assertions.

```
In[69]:= TRANSITIVE[V] // AssertTest

Out[69]= TRANSITIVE[V] == False

In[70]:= TRANSITIVE[V] := False

In[71]:= Select[Relns[Map[#[x]&, UnaryFunctors]], subclass[composite[#, #], #]&]

Out[71]= Relns[]

In[72]:= Map[AssertTest[TRANSITIVE[#]]&,
           {ADJOIN[x], CORE[x], HULL[x], id[x], IMAGE[id[x]], trv[x]}] //
         TableForm

Out[72]//TableForm=
      TRANSITIVE[ADJOIN[x]] == True
      TRANSITIVE[CORE[x]] == True
      TRANSITIVE[HULL[x]] == True
      TRANSITIVE[id[x]] == True
      TRANSITIVE[IMAGE[id[x]]] == True
      TRANSITIVE[trv[x]] == True

In[73]:= TRANSITIVE[ADJOIN[x_]] := True

In[74]:= TRANSITIVE[CORE[x_]] := True

In[75]:= TRANSITIVE[HULL[x_]] := True

In[76]:= TRANSITIVE[id[x_]] := True
```

```
In[77]:= TRANSITIVE[IMAGE[id[x_]]] := True

In[78]:= TRANSITIVE[trv[x_]] := True
```