

# irreflexive transitive relations

Johan G. F. Belinfante  
2004 April 4

```
In[1]:= << goedel56.01a; << tools.m;

:Package Title: goedel56.01a          2004 April 1 at 9:10 p.m.

It is now: 2004 Apr 5 at 12:59

Loading Simplification Rules

TOOLS.M                               Revised 2004 March 29

weightlimit = 40
```

---

## summary

Any asymmetric relation is irreflexive, and any irreflexive transitive relation is asymmetric. If  $x$  is a transitive relation, then  $\text{dif}[x, \text{inverse}[x]]$  is an irreflexive transitive relation. These facts, and some related results are derived, including applications and variable-free versions that are valid in the case of sets.

---

## definitions

A relation is **irreflexive** if

```
In[2]:= subclass[x, Di]

Out[2]= and[equal[0, fix[x]], subclass[x, cart[V, V]]]
```

A relation is **antisymmetric** if

```
In[3]:= subclass[intersection[x, inverse[x]], Id]

Out[3]= subclass[intersection[x, inverse[x]], Id]
```

A relation is **asymmetric** if the following stronger result is true:

```
In[4]:= disjoint[x, inverse[x]]

Out[4]= equal[0, intersection[x, inverse[x]]]
```

A relation is **transitive** if

```
In[5]:= subclass[composite[x, x], x]

Out[5]= TRANSITIVE[composite[Id, x]]
```

Some temporary abbreviations are convenient:

```
In[6]:= irr[x_] := dif[x, inverse[x]]
```

```
In[7]:= compdup[x_] := composite[x, x]
```

---

## theorem 1: asymmetric => irreflexive

```
In[8]:= SubstTest[implies, subclass[u, v],
  subclass[compdup[u], compdup[v]], {u -> id[fix[x]], v -> x}]
```

```
Out[8]= subclass[fix[x], fix[composite[x, x]]] = True
```

```
In[9]:= subclass[fix[x_], fix[composite[x_, x_]]] := True
```

A generalization:

```
In[10]:= SubstTest[implies, and[subclass[t, u], subclass[v, w]],
  subclass[composite[t, v], composite[u, w]],
  {t -> id[fix[x]], u -> x, v -> id[fix[y]], w -> y}]
```

```
Out[10]= subclass[intersection[fix[x], fix[y]], fix[composite[x, y]]] = True
```

```
In[11]:= subclass[intersection[fix[x_], fix[y_]], fix[composite[x_, y_]]] := True
```

The following theorem is proved along similar lines:

```
In[12]:= SubstTest[implies, subclass[u, v],
  subclass[fix[u], fix[v]], {u -> composite[Id, x], v -> trv[x]}]
```

```
Out[12]= subclass[fix[x], fix[trv[x]]] = True
```

```
In[13]:= subclass[fix[x_], fix[trv[x_]]] := True
```

The presence of rewrite rules makes the following derivation a bit tricky; the outer head for **SubstTest** here must be **and**, not **implies**, as one might naively have guessed. In general, when using **SubstTest**, it is important that it be applied to the innermost head for which there is a rewrite rule.

```
In[14]:= Map[implies[#, equal[0, fix[x]]] &,
  SubstTest[and, subclass[u, v], equal[0, v], {u -> fix[x], v -> fix[compdup[x]}]]]
```

```
Out[14]= or[equal[0, fix[x]], not[equal[0, fix[composite[x, x]]]]] = True
```

```
In[15]:= or[equal[0, fix[x_]], not[equal[0, fix[composite[x_, x_]]]]] := True
```

It follows that an asymmetric relation must be irreflexive.

```
In[16]:= implies[disjoint[x, inverse[x]], equal[0, fix[x]]] // AssertTest
```

```
Out[16]= or[equal[0, fix[x]], not[equal[0, intersection[x, inverse[x]]]]] = True
```

```
In[17]:= or[equal[0, fix[x_]], not[equal[0, intersection[x_, inverse[x_]]]]] := True
```

Corollary.

```
In[18]:= implies[and[subclass[x, cart[V, V]], disjoint[x, inverse[x]]], subclass[x, Di]]
Out[18]= True
```

A related result:

```
In[19]:= Map[implies[#, equal[0, fix[x]]] &,
  SubstTest[and, subclass[u, v], equal[0, v], {u -> fix[x], v -> fix[trv[x]}]]]
Out[19]= or[equal[0, fix[x]], not[equal[0, fix[trv[x]]]]] == True
In[20]:= or[equal[0, fix[x_]], not[equal[0, fix[trv[x_]]]]] := True
```

---

## theorem 2: irreflexive + transitive => asymmetric

A converse assertion holds for transitive relations.

```
In[21]:= Map[implies[subclass[x, cart[V, V]], #] &,
  SubstTest[implies, subclass[y, x], subclass[fix[y], fix[x]], y -> compdup[x]]]
Out[21]= or[not[TRANSITIVE[x]], subclass[fix[composite[x, x]], fix[x]]] == True
In[22]:= (% /. x -> x_) /. Equal -> SetDelayed
In[23]:= Map[not, SubstTest[and, implies[p1, p3], implies[and[p2, p3], p4],
  not[implies[and[p1, p2], p4]], {p1 -> TRANSITIVE[x], p2 -> equal[0, fix[x]],
  p3 -> subclass[fix[compdup[x]], fix[x]], p4 -> equal[0, fix[compdup[x]]}]]]
Out[23]= or[equal[0, fix[composite[x, x]]], not[equal[0, fix[x]]], not[TRANSITIVE[x]]] == True
In[24]:= or[equal[0, fix[composite[x_, x_]]],
  not[equal[0, fix[x_]]], not[TRANSITIVE[x_]]] := True
```

A variant of this:

```
In[25]:= or[equal[0, intersection[x, inverse[x]]],
  not[equal[0, fix[x]]], not[TRANSITIVE[x]]] // AssertTest
Out[25]= or[equal[0, intersection[x, inverse[x]]],
  not[equal[0, fix[x]]], not[TRANSITIVE[x]]] == True
In[26]:= or[equal[0, intersection[x_, inverse[x_]]],
  not[equal[0, fix[x_]]], not[TRANSITIVE[x_]]] := True
```

This corollary does not require an extra rewrite rule:

```
In[27]:= implies[and[TRANSITIVE[x], subclass[x, Di]], disjoint[x, inverse[x]]]
Out[27]= True
```

The following equivalence can be formulated as a rewrite rule.

```
In[28]:= equiv[and[equal[0, intersection[x, inverse[x]]], TRANSITIVE[x]],
  and[equal[0, fix[x]], TRANSITIVE[x]]]
Out[28]= True
```

```

In[29]:= and[equal[0, intersection[x_, inverse[x_]]], TRANSITIVE[x_]] :=
    and[equal[0, fix[x]], TRANSITIVE[x]]

In[30]:= equiv[and[equal[0, fix[compdup[x]]], TRANSITIVE[x]],
    and[equal[0, fix[x]], TRANSITIVE[x]]]

Out[30]= True

In[31]:= Equal[and[equal[0, fix[compdup[x]]], TRANSITIVE[x]],
    and[equal[0, fix[x]], TRANSITIVE[x]]]

Out[31]= and[equal[0, fix[composite[x, x]]], TRANSITIVE[x]] ==
    and[equal[0, fix[x]], TRANSITIVE[x]]

In[32]:= and[equal[0, fix[composite[x_, x_]]], TRANSITIVE[x_]] :=
    and[equal[0, fix[x]], TRANSITIVE[x]]

```

---

## variable-free formulations

The following two classes are related by the first theorem:

```

In[33]:= class[x, and[subclass[x, cart[V, V]], equal[0, fix[x]]]]

Out[33]= P[Di]

In[34]:= class[x, and[subclass[x, cart[V, V]], equal[0, fix[compdup[x]]]]]

Out[34]= fix[composite[DISJOINT, INVERSE]]

```

The latter class can also be described another way:

```

In[35]:= class[x, and[subclass[x, cart[V, V]], disjoint[x, inverse[x]]]]

Out[35]= fix[composite[DISJOINT, INVERSE]]

```

Lemma.

```

In[36]:= Map[equal[0, #] &, dif[fix[composite[DISJOINT, INVERSE]], P[cart[V, V]]] // Renormality]

Out[36]= subclass[U[fix[composite[DISJOINT, INVERSE]]], cart[V, V]] == True

In[37]:= % /. Equal -> SetDelayed

```

Lemma.

```

In[38]:= Map[equal[0, #] &,
    intersection[complement[P[Di]], fix[composite[DISJOINT, INVERSE]]] // Renormality]

Out[38]= equal[0, fix[U[fix[composite[DISJOINT, INVERSE]]]]] == True

In[39]:= fix[U[fix[composite[DISJOINT, INVERSE]]]] := 0

```

To get an equation, and not just an inclusion, one needs a further fact:

```

In[40]:= member[singleton[PAIR[x, y]], fix[composite[DISJOINT, INVERSE]]] // AssertTest
Out[40]= member[cart[singleton[x], singleton[y]], fix[composite[DISJOINT, INVERSE]]] ==
  or[not[equal[x, y]], not[member[x, V]], not[member[y, V]]]

In[41]:= member[cart[singleton[x_], singleton[y_]], fix[composite[DISJOINT, INVERSE]]] :=
  or[not[equal[x, y]], not[member[x, V]], not[member[y, V]]]

In[42]:= Map[equal[0, class[pair[x, y], not[#]]] &,
  SubstTest[implies, and[member[u, v], member[v, w]], member[u, U[w]],
    {u -> PAIR[x, y], v -> singleton[PAIR[x, y]], w -> fix[composite[DISJOINT, INVERSE]]}]
Out[42]= subclass[Di, U[fix[composite[DISJOINT, INVERSE]]]] == True

In[43]:= % /. Equal -> SetDelayed

```

Theorem.

```

In[44]:= SubstTest[and, subclass[u, v], subclass[v, u],
  {u -> U[fix[composite[DISJOINT, INVERSE]]], v -> Di}
Out[44]= True == equal[Di, U[fix[composite[DISJOINT, INVERSE]]]]

In[45]:= U[fix[composite[DISJOINT, INVERSE]]] := Di

```

For the second theorem, one obtains the following variable-free formulation:

```

In[46]:= Map[equal[0, #] &,
  dif[intersection[TRV, P[Di]], fix[composite[DISJOINT, INVERSE]]] // Renormality]
Out[46]= subclass[intersection[TRV, P[Di]], fix[composite[DISJOINT, INVERSE]]] == True

In[47]:= subclass[intersection[TRV, P[Di]], fix[composite[DISJOINT, INVERSE]]] := True

```

This result can be rewritten as an equation by intersecting with TRV.

```

In[48]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
  subclass[u, w], {u -> intersection[TRV, fix[composite[DISJOINT, INVERSE]]],
    v -> fix[composite[DISJOINT, INVERSE]], w -> P[Di]}
Out[48]= and[equal[0, fix[U[intersection[TRV, fix[composite[DISJOINT, INVERSE]]]]],
  subclass[U[intersection[TRV, fix[composite[DISJOINT, INVERSE]]], cart[V, V]]] == True

In[49]:= % /. Equal -> SetDelayed

In[50]:= SubstTest[and, subclass[u, v], subclass[v, u],
  {u -> intersection[TRV, fix[composite[DISJOINT, INVERSE]]],
    v -> intersection[TRV, P[Di]]}
Out[50]= True ==
  equal[intersection[TRV, fix[composite[DISJOINT, INVERSE]]], intersection[TRV, P[Di]]]

In[51]:= intersection[TRV, fix[composite[DISJOINT, INVERSE]]] := intersection[TRV, P[Di]]

```

---

## lemmas

The monotonicity of **compdup** implies:

```
In[52]:= SubstTest[implies, subclass[w, x],
  subclass[compdup[w], compdup[x]], w → intersection[x, y]]

Out[52]= subclass[composite[intersection[x, y], intersection[x, y]], composite[x, x]] == True

In[53]:= subclass[composite[intersection[x_, y_], intersection[x_, y_]],
  composite[x_, x_]] := True
```

In particular:

```
In[54]:= Map[implies[subclass[x, cart[V, V]], #] &,
  SubstTest[implies, and[subclass[u, v], subclass[v, w]],
  subclass[u, w], {u → compdup[irr[x]], v → compdup[x], w → x}]]

Out[54]= or[not[TRANSITIVE[x]], subclass[composite[intersection[x, complement[inverse[x]]],
  intersection[x, complement[inverse[x]]]], x]] == True

In[55]:= (% /. x → x_) /. Equal → SetDelayed
```

This is a form of the Schröder rotation theorem:

```
In[56]:= implies[disjoint[composite[x, y], z], disjoint[composite[z, inverse[y]], x]] //
  AssertTest // InvertFix

Out[56]= or[equal[0, intersection[x, composite[z, inverse[y]]]],
  not[equal[0, intersection[z, composite[x, y]]]] == True

In[57]:= or[equal[0, intersection[x_, composite[z_, inverse[y_]]]],
  not[equal[0, intersection[z_, composite[x_, y_]]]] := True
```

This is a related result, obtained by interchanging  $y$  and its inverse.

```
In[58]:= SubstTest[implies, disjoint[composite[x, w], z],
  disjoint[composite[z, inverse[w]], x], w → inverse[y]]

Out[58]= or[equal[0, intersection[x, composite[z, y]]],
  not[equal[0, intersection[z, composite[x, inverse[y]]]]] == True

In[59]:= or[equal[0, intersection[x_, composite[z_, y_]]],
  not[equal[0, intersection[z_, composite[x_, inverse[y_]]]]] := True
```

---

## derivation

Lemma.

```
In[60]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
  {u → intersection[x, composite[inverse[x], intersection[complement[x], inverse[x]]],
  v → composite[inverse[x], intersection[complement[x], inverse[x]]],
  w → compdup[inverse[x]]}]

Out[60]= subclass[
  intersection[x, composite[inverse[x], intersection[complement[x], inverse[x]]],
  composite[inverse[x], inverse[x]]] == True

In[61]:= (% /. x → x_) /. Equal → SetDelayed
```

```
In[62]:= Map[implies[subclass[x, cart[V, V]], #] &,
  SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
    {u → intersection[x, composite[inverse[x], intersection[complement[x], inverse[x]]],
      v → compdup[inverse[x]], w → inverse[x]}]]
```

```
Out[62]= or[not[TRANSITIVE[x]], subclass[
  intersection[x, composite[inverse[x], intersection[complement[x], inverse[x]]],
  inverse[x]]] = True
```

```
In[63]:= (% /. x → x_) /. Equal → SetDelayed
```

By Schröder rotation, one has:

```
In[64]:= SubstTest[implies, disjoint[composite[u, inverse[v]], w],
  disjoint[composite[w, v], u], {u → inverse[x], v → irr[x], w → irr[x]}]
```

```
Out[64]= or[equal[0, intersection[composite[intersection[x, complement[inverse[x]]],
  intersection[x, complement[inverse[x]]], inverse[x]],
  not[subclass[intersection[x, composite[inverse[x],
  intersection[complement[x], inverse[x]]], inverse[x]]] = True
```

```
In[65]:= (% /. x → x_) /. Equal → SetDelayed
```

```
In[66]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
  not[implies[p1, p3]], {p1 → TRANSITIVE[x], p2 → subclass[
  intersection[x, composite[inverse[x], intersection[complement[x], inverse[x]]],
  inverse[x]], p3 → disjoint[inverse[x], compdup[irr[x]]}]]]
```

```
Out[66]= or[equal[0, intersection[composite[intersection[x, complement[inverse[x]]],
  intersection[x, complement[inverse[x]]], inverse[x]], not[TRANSITIVE[x]]] = True
```

```
In[67]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemmas:

```
In[68]:= Map[implies[TRANSITIVE[x], #] &, SubstTest[subclass, t, intersection[x, y],
  {t → compdup[irr[x]], y → complement[inverse[x]]}] // MapNotNot
```

```
Out[68]= or[not[TRANSITIVE[x]],
  TRANSITIVE[composite[Id, intersection[x, complement[inverse[x]]]]] = True
```

```
In[69]:= (% /. x → x_) /. Equal → SetDelayed
```

```
In[70]:= SubstTest[and, subclass[w, cart[V, V]], TRANSITIVE[composite[Id, w]], w → irr[x]]
```

```
Out[70]= and[subclass[x, cart[V, V]],
  TRANSITIVE[composite[Id, intersection[x, complement[inverse[x]]]]] ==
  TRANSITIVE[intersection[x, complement[inverse[x]]]
```

```
In[71]:= and[subclass[x_, cart[V, V]],
  TRANSITIVE[composite[Id, intersection[x_, complement[inverse[x_]]]]] :=
  TRANSITIVE[intersection[x, complement[inverse[x]]]
```

Final result

```
In[72]:= SubstTest[and, implies[p1, p2], implies[p1, p3], {p1 → TRANSITIVE[x],
  p2 → subclass[irr[x], cart[V, V]], p3 → TRANSITIVE[composite[Id, irr[x]]}] // Reverse
```

```
Out[72]= or[not[TRANSITIVE[x]], TRANSITIVE[intersection[x, complement[inverse[x]]]] = True
```

```
In[73]:= or[not[TRANSITIVE[x_]], TRANSITIVE[intersection[x_, complement[inverse[x_]]]] := True
```

---

## numerical dominance and Cantor's theorem

The following transitive relation in cardinal number theory is not antisymmetric, and therefore fails to be a partial order.

```
In[74]:= SubstTest[subclass, compdup[x], x, x → composite[Q, S]] // Reverse
```

```
Out[74]= TRANSITIVE[composite[Q, S]] == True
```

```
In[75]:= TRANSITIVE[composite[Q, S]] := True
```

The irreflexive relation derived from it is called numerical dominance:

```
In[76]:= SubstTest[implies, TRANSITIVE[x], TRANSITIVE[irr[x]], x → composite[Q, S]]
```

```
Out[76]= TRANSITIVE[intersection[complement[composite[Q, inverse[S]]], composite[Q, S]]] == True
```

```
In[77]:= TRANSITIVE[intersection[complement[composite[Q, inverse[S]]], composite[Q, S]]] := True
```

Cantor's theorem implies that any set is numerically dominated by its power set. No new rewrite rule is needed for this result:

```
In[78]:= subclass[POWER, intersection[complement[composite[Q, inverse[S]]], composite[Q, S]]]
```

```
Out[78]= True
```